Title
GIST: An Interactive, GPU-Based Level-Set Segmentation

Journal
Medical Image Analysis, 8

Authors
Cates, Joshua E.
Lefohn, Aaron
Whitaker, Ross T.

Publication Date
2004

Peer reviewed

GIST: An Interactive, GPU-Based Level Set Segmentation Tool for 3D Medical Images

Joshua E. Cates, Aaron E. Lefohn, Ross T. Whitaker
Scientific Computing and Imaging Institute,
University of Utah, Salt Lake City, UT 84112-9205
Contact: cates@cs.utah.edu

February 4, 2004

Abstract

While level sets have demonstrated a great potential for 3D medical image segmentation, their usefulness has been limited by two problems. First, 3D level sets are relatively slow to compute. Second, their formulation usually entails several free parameters which can be very difficult to correctly tune for specific applications. The second problem is compounded by the first. This paper describes a new tool for 3D segmentation that addresses these problems by computing level-set surface models at interactive rates. This tool employs two important, novel technologies. First is the mapping of a 3D level-set solver onto a commodity graphics card (GPU). This mapping relies on a novel mechanism for GPU memory management. The interactive rates level-set PDE solver give the user immediate feedback on the parameter settings, and thus users can tune free parameters and control the shape of the model in real time. The second technology is the use of region-based speed functions, which allow a user to quickly and intuitively specify the behavior of the deformable model. We have found that the combination of these interactive tools enables users to produce good, reliable segmentations. To support this observation, this paper presents qualitative results from several different datasets as well as a quantitative evaluation from a study of brain tumor segmentations.

1 Introduction

Image segmentation is arguably the most widely studied problem in image processing, and the literature shows a plethora of image segmentation algorithms that rely on a diverse range of strategies such as statistics, differential geometry, heuristics, graph theory, and algebra. No one segmentation technique has emerged as being superior to all others in all circumstances, and thus it seems that the field of medical image processing will evolve to a state where researchers and clinicians have access to a set of segmentation tools, i.e. a toolbox, from which they can choose the particular tool that is best suited for their particular application.

A complete segmentation toolbox will include a set of *general purpose* tools as well as various *specialized* segmentation tools. General purpose tools are those that can be quickly launched and used as the need arises in a wide range of applications. Specialized tools rely on stronger assumptions about a specific modality, anatomy, or application. When properly trained, tuned, and applied we would expect specialized tools to perform better than general purpose tools—when all other factors, such as operator time and compute time, are equal. Among general tools, the most popular example, and the goal standard for many applications, is hand contouring, which entails a knowledgeable user (e.g. a medical doctor) creating a 2D curve, drawn by manipulating a mouse, on a sequence of slices to delineate the object of interest.

This paper describes a new, general-purpose segmentation tool that relies on interactive deformable models implemented as level sets. While level sets have demonstrated a great potential for 3D medical image segmentation, their usefulness has been limited by two problems. First, 3D level sets are relatively slow to compute. Second, their formulation usually entails several free parameters, which can be very difficult to correctly tune for specific applications. The second problem is compounded by the first. That is, users find it impractical to explore the space of possible parameter settings when an example result from a point in that space requires minutes or hours to generate.

The software application described in this paper is called GIST (GPU-based Interactive Segmentation Tool). GIST updates a level-set surface model at interactive rates on commodity

graphics cards (GPUs), such as those that are commonly found on consumer-level personal computers. It can be applied to a general set of medical and biological applications by tuning several free parameters. Despite its general nature, we demonstrate the effectiveness of GIST by a quantitative comparison to a specialized tool and the associated gold standard for a specific problem: brain tumor segmentation [1, 2]. This paper make the following contributions:

- A 3D segmentation tool that uses a new level-set deformation solver to achieve interactive rates (approximately 10-15 times faster than previous solutions).
- A interactive mechanism for defining a level-set speed function that works on both scalar and multivalued (i.e. spectral) data.
- Quantitative and qualitative evidence that this interactive level-set approach is effective for brain tumor segmentation.

The remainder of the paper is organized as follows. The next section gives some technical background and related work on level sets, GPUs, and segmentation evaluation methods. Section 3 describes the formulation of the level-set equations and the solution on the GPU. Section 5.2 presents qualitative results on various datasets and a quantitative analysis of the performance of the method for brain tumor segmentation. Section 6 summarizes this work.

2 Background and Related Work

2.1 Level Sets

This paper relies on an implicit representation of deformable surface models called the method of *level sets*, proposed by Osher and Sethian [3]. The level-set method (See also Sect. 3) computes the motion of a moving interface by solving a partial differential equation (PDE) on a volume. The use of level sets has been widely documented in the medical imaging literature, and several works give more comprehensive reviews of the method and the associated numerical techniques [4, 5].

For certain classes of applications level sets have several advantages over parametric models. Because they are implicit, level sets can change topology. This means that during a deformation a user need not worry about surfaces colliding or pinching off. Also, level sets do not require reparameterization as they deform far from their initial conditions—e.g. deformable meshes typically require the insertion or deletion of triangles under such circumstances [6]. Finally, level sets allow for *geometric* surface deformations, which means that the results of a deformation process depend on the shape of the surface and the input data and *not* on some underlying parameterization. The level-set method is a general framework that must be tuned to specific applications.

As with the original work on image segmentation by parametric deformable models [7], the level-set approach to segmentation typically combines a data-fitting term with a smoothing term. However, there are alternatives. For instance, Whitaker [8] proposes a formulation that mimics parametric deformable models, in which level surfaces move toward edges (high gradient magnitude) in volumes. In that formulation the model must be within a somewhat narrow band of attraction (defined by the second derivative) in order to lock onto such edges, and therefore the author proposes a multiscale computational method to improve convergence. Malladi et al. [9] describe a formulation in which level curves/surfaces expand (or contract) with a motion that slows at image edges. Because of the monotonic expansion/contraction, convergence to local minima is less of a problem, but the results tend to be biased either inward or outward. Caselles et al. [10] propose an alternative that minimizes an edge-weighted area metric. In that case the data term is weighted more heavily as the model approaches its target. These methods (and many others) focus on image edges, but the the literature documents several other strategies for fitting level sets to image data. For instance, several authors have propose using the statistics of the greyscale interior of the model to control the motion [11, 12]. Alternatively, the motion of the level set can depend on a variational formulation that positions the interface to create discontinuities that best model the discontinuities in the input data [13, 14, 15]. In this paper we use a supervised, statistical classifier to drive the motion of the level-set model.

Virtually all of these methods include a form of mean curvature to keep the level-set smooth

as it converges on a solution. Whitaker [8] proposes a weighted sum of principle curvatures to preserve cylindrical structures. Lorigio et al. [16] proposes the minimum curvature, in the context of segmenting blood vessels; this is equivalent to a space-curvature shortening for very thin objects. Recently, Tasdizen et al. propose the diffusion of normals in order to approximate higher-order geometric flows [17, 18].

Solving level-set PDEs on a volume requires proper numerical schemes [19] and entails a significant computational burden. Stability requires that the surface can progress at most a distance of one voxel at each iteration, and thus a large number of iterations are required to compute significant deformations. There is a special case of the level-set PDEs in which the surface motion is strictly inward or outward. Such equations can be solved somewhat efficiently using the *fast marching method* [4] and variations thereof [20]. However, this case covers only a very small subset of interesting speed functions, and such speed functions are inconsistent with interactive parameter tuning. In general we are concerned with problems that include a surface curvature term and simultaneously require the model to expand and contract to match the data.

Efficient algorithms for solving the more general level-set problem rely on the observation that at any one time step the only parts of the solution that are important are those adjacent to the moving surface. In light of this several authors have proposed numerical schemes that compute solutions for only those voxels that lie in a small number of layers adjacent to the surface as shown in Fig. 1b. Adalsteinsson and Sethian [21] have proposed the *narrow band method*, which updates the embedding on a band of 10-20 pixels around the model, and reinitializes that band whenever the model approaches the edge. Whitaker [22] proposed the *sparse-field* method, which introduces a scheme in which updates are calculated only on the wavefront, and several layers around that wavefront are updated via a distance transform at each iteration. Peng et al. [23] present a similar local method. Even with this very narrow band of computation, update rates using conventional processors on typical medical data sets (e.g. 256^3 voxels) are not interactive. This is the motivation behind the GPU-based solver in GIST.

2.2 Graphics Processing Units for Scientific Computation

Graphics processing units have been developed primarily for the computer gaming industry, but over the last several years researchers have come to recognize them as low cost, high performance computing platforms. Two important trends in GPU development, increased programmability and higher precision arithmetic processing, have helped to foster new non-gaming applications.

Graphics processors outperform central processing units (CPUs)—often by more than an order of magnitude—because of their *streaming* architecture[24] and dedicated high-speed memory. In the streaming model of computation, arrays of input data are processed identically by the same computation *kernel* to produce output data streams. The GPU takes advantage of the data-level parallelism inherent in this model by having many identical processors execute the computation in parallel. This computation model has been used by a number of researchers to map a wide variety of computationally demanding problems to GPUs. Examples include matrix multiplication, finite element methods, and multi-grid solvers [25, 26, 27]. All of these examples demonstrate a homogeneous sequence of operations over a densely populated grid structure.

Strzodka et al. [28] were the first to show that the level-set equations could be solved using a graphics processor. Their solver implements the two-dimensional level-set method using a time-invariant speed function for flood-fill-like image segmentation, without the associated curvature. Their solver does not take advantage of the sparse nature of the level-set PDEs and therefore performs only marginally better than a highly-optimized sparse-field CPU implementation. The work in this paper relies on a three-dimensional generalization of [28], which includes a second-order curvature computation, and a significantly improved GPU solver that implements a narrow-band strategy. Also related is the work of Sherbondy et al. [29], in which they identify regions of interest to solve a diffusion equation for volume segmentation.

This paper describes a GPU computational model that supports *time-dependent, sparse grid* problems. These problems are difficult to solve efficiently with GPUs for two reasons. The first is that in order to take advantage of the GPU's parallelism, the streams being processed must be large, contiguous blocks of data, and thus grid points near the level-set surface model must be

packed into a small number of textures. The second difficulty is that the level set moves with each time step, and thus the packed representation must readily adapt to the changing position of the model. This requirement is in contrast to the recent sparse matrix solvers [30, 31] and previous work on rendering with compressed data [32, 33]. In the two sparse-matrix solvers[30, 31], a packed texture scheme is used to efficiently compute sparse matrix-vector multiplications as well as compute values of the sparse matrix elements on the GPU. The scheme is static, however, in the sense that the nonzero matrix elements must be identified before the computation begins.

2.3 Segmentation Evaluation

This paper includes a systematic evaluation of the performance of GIST. The role of segmentation evaluation is to understand the strengths, limitations, and potential applications of a particular segmentation algorithm. There are two strategies for evaluating segmentation algorithms. One strategy is to study segmentation performance in the context of a particular clinical or scientific question [34, 35]. For instance, the effectiveness of the algorithm within a study that monitors the volumes or sizes of tumors. The second approach is to study to evaluate segmentation in the absence of a specific clinical application by quantifying the general behavior of the algorithm relative to an *ideal* solution. This paper takes the second approach, and uses general shape metrics to compare watershed segmentation results with the defacto gold standard for clinical applications, which is *hand contouring* one slice at a time (which we will also call *manual* segmentation) by expert observers.

Segmentation evaluation is difficult because of the lack of standard metrics and the difficulty of establishing ground truth in clinical data. Our evaluation methodology is derived from ideas developed by [36], and others [37, 38, 39], who emphasize the importance of quantitative evaluation and statistical metrics. The study in this paper concerns a user-assisted segmentation technique, which requires a user-based evaluation to capture variations in the individual decision-making process. Experimental trials across a number of users and images[40, 41] can generate data appropriate for

statistical analysis that account for user variability.

A combination of different factors determines the effectiveness of a segmentation. For instance Udupa et. al[37] propose a quantification of performance based on validity of the results (accuracy), reproducibility of the results (precision), and efficiency of the segmentation method (time). Other researchers have studied the sensitivity of the technique to various disruptive factors such as data artifacts, pathology, or individual anatomical variation (robustness) [42].

Accuracy metrics typically rely on a *ground truth* segmentation—segmentations that are somehow close to this ground truth are considered better than those that are not. Studies with digital or physical phantoms provide a ready definition of ground truth. However, for biological or clinical data sets, ground truth is usually unknown. In such a case, researchers typically rely on experts to delineate the ground truth by hand [42, 43]. Experts seldom all agree, but a statistical combination (averaging) of several expert segmentations can account for expert variability. Averaging of multiple nonparametric shapes, however, is itself a difficult problem. One technique for combining multiple segmentations is *Simultaneous Truth and Performance Level Estimation* (STAPLE), [44]. This treats segmentation as a pixelwise classification, which leads to an averaging scheme that accounts for systematic biases in the behavior of experts

The accuracy of an individual experimental segmentation is usually given through some measure of a region's overlap and its distance from the ground truth. Common distance metrics include the Hausdorff distance [45] and the root mean squared distance between selected boundary points [38, 39]. Often overlap is characterized by a *similarity* measure between experimental and ground truth volumes. One common similarity measure is the cardinality of the intersection (in pixels or voxels) of positive classifications in two volumes volumes over the union of the positive classifications [40, 46], denoted s. Another overlap metric is the *total correct fraction*, c, which is simply the percentage of correctly classified pixels in the image volume (negative and positive) [1].

Another strategy for evaluating a single-object segmentation is to view each pixel as an instance of a detection task, which gives rise to metrics for sensitivity and specificity. Sensitivity, p, is the true positive fraction of the segmentation, the percentage of pixels in an image correctly classified

as lying inside the object boundary. Specificity, q, is the true negative fraction, the percentage of pixels in a segmentation correctly classified as lying outside the object boundary. Because there is an explicit tradeoff between sensitivity and specificity, researchers have proposed using receiver operator characterizations (ROC), which monitor the behavior of this tradeoff for different segmentation algorithms or parameter settings [47, 37].

The precision of a segmentation method is an indicator of how repeatable the results are using that technique. Alternatively, precision is an indicator or the degree of randomness inherent to the method. Precision does not rely on knowing ground truth and can be estimated by applying the similarity measure s within a set of experimental segmentations [37]. The mean s value from these comparisons gives a characterization of the precision of the method.

The efficiency of a segmentation technique is a measure of the time involved in achieving a segmentation. This can include user interaction and compute times. These two characteristics are usually considered individually, because each has a separate cost and will affect the practicability of the method in a way that depends on the specific application.

3 Level-Set Formulation and Algorithms

We begin this section with a brief review of the notation and mathematics of level-set methods and describe the particular formulation that is relevant to this paper. Comprehensive reviews of level-set methods are given the literature [4, 5].

An implicit model is a surface representation in which the surface consists of all points $\mathcal{S} = \{\bar{x}|\phi(\bar{x}) = 0\}$, where $\phi : \Re^3 \mapsto \Re$. Level-set methods relate the motion of that surface to a PDE on the volume, i.e. $\partial\phi/\partial t = -\nabla\phi \cdot \bar{v}(t)$, where describes the motion $\bar{v}(t)$ of the surface. Within this framework one can implement a wide range of deformations by defining an appropriate $\bar{v}$. For segmentation, the velocity often consists of a combination of two terms [8, 9]

$$\frac{\partial\phi}{\partial t} = |\nabla\phi| \left[\alpha D(\bar{x}) + (1-\alpha)\nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}\right], \tag{1}$$

where D is a data term that forces the model toward desirable features in the input data, the term $\nabla \cdot (\nabla\phi/|\nabla\phi|)$ is the mean curvature of the surface, which forces the surface to have less area (and remain smooth), and $\alpha \in [0,1]$ is a free parameter that controls the degree of smoothness in the solution. There are several variations on this framework in the literature, e.g. [10].

The behavior of the model is mostly characterized by the data term and how it relates to the image. Invariably, the data term introduces free parameters, and the proper tuning of those parameters, along with α, is critical to making the model behave in a desirable manner.

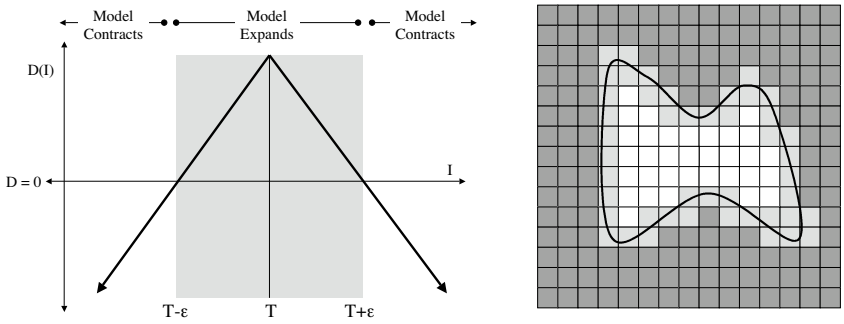
3.1 An Intensity-Based Speed Function

Our strategy is to construct a speed function D that causes the model to grow in regions where the data is consistent with the desired segmentation and to contract in regions where it is not. We can achieve this by letting D have positive or negative values depending on whether or not the model is within a specified range of data values. In this case the speed function at any one point is based solely on input value value I at the point $\vec{x}$ in the image, i.e $D(\vec{x}) = D(I(\vec{x}))$.

Such a simple scalar speed function is given by

$$D(I) = \varepsilon - |I - T|, \tag{2}$$

where T controls the dominant intensity of the region to be segmented and ε controls the range of greyscale values around T that could be considered inside the object. Thus when the model lies on a voxel with a greyscale level between $T-\varepsilon$ and $T+\varepsilon$, the model expands and otherwise it contracts. The speed term is gradual, and thus the effects of the D diminish as the model approaches the boundaries of regions whose greyscale levels lie within the $T \pm \varepsilon$ range. Even with this simple scheme a user would have to specify three free parameters, T, ε, and α, *as well as* an initialization. Figure 1 shows a graph of D defined in this manner.



(a) (b)

Figure 1: (a) A speed function based on image intensity causes the model to expand over regions with greyscale values within the specified range and contract otherwise. (b) Efficient implementations of level sets entail computing the solution only near the moving wavefront.

3.2 A Statistical Classifier

The speed term in (2) represents a simple one-dimensional, two-class statistical classifier. If we let $P(A|I)$ be the probability that a pixel lies in the object conditional on the pixel/voxel value I and $P(B|I)$ be the probability that the pixel is not in the object, then the Bayesian decision variable is

$$\mathcal{D} = \frac{P(A|I)}{P(B|I)} = \frac{P(I|A)P(A)}{P(I|B)P(B)}, \tag{3}$$

which should be compared to unity in order to decide on either A or B. If all intensities in the background, B, are equally likely (the goal here is *simplicity*), the denominator is constant, and the log of $\mathcal{D}$ is

$$\log \mathcal{D} = \log P(I|A) + \log P(A) - \log P(I|B) - \log P(B). \tag{4}$$

If we let the statistics of the inside of the object be Gaussian $P(A|I) = \exp[-(I-\mu)^2/2\sigma^2]$, we have the following decision rule for a pixel $\bar{x}$ with intensity I:

$$\bar{x} \in \begin{cases} A & |I-T| \leq \varepsilon \\ B & \text{otherwise} \end{cases}, \tag{5}$$

where $\varepsilon = \left[\left|2\sigma^2(\log P(A) - \log P(I|B) - \log P(B))\right|\right]^{\frac{1}{2}}$, and $T = \mu$. Thus, we see that this simple three parameter model allows a user to explore the possibilities of this simple statistical classifier and combine it with the geometric information embodied in the curvature of the level set.

This analysis sheds some light on the proper interface for these parameters. For instance, we can help the user choose these parameters by providing a interactive tool that allows a user to select a set of points (e.g. by holding down a mouse button and moving the cursor over the area of interest) that generate a mean and a variance, and use these values initialize T and ε.

3.3 A Speed Function for Spectral Volumes

This statistical classifier also extends to image/volumes with multiple values, i.e. *spectral images*, with values denoted $\bar{I}(\bar{x}) \in \Re^m$. In this case the in-object condition probability is

$$P(A|\bar{I}) = \exp\left[-\frac{1}{2}(\bar{I} - \bar{\mu})^T \Sigma^{-1}(\bar{I} - \bar{\mu})\right], \tag{6}$$

where Σ is the covariance. We can express the classifier in terms of Mahalanobis distance. That is:

$$\bar{x} \in \begin{cases} A & D \leq \varepsilon \\ B & \text{otherwise} \end{cases}, \tag{7}$$

and $D = \left[(\bar{I} - \bar{\mu})^T \Sigma^{-1}(\bar{I} - \bar{\mu})\right]^{\frac{1}{2}}$.

The graph of the speed function given in (7) is an ellipsoidal hypercone that crosses the zero axis of the independent variable to form an ellipsoid, centered at $\bar{\mu}$, with a shape and orientation given by the covariance. Figure 2 depicts this for $m = 2$. The free parameter ε defines the *w*idth of the resulting ellipsoidal classifier in units of standard deviation. The model expands when it lies on a pixel whose value is within the classifier range and contracts elsewhere. This Mahalanobis classifier is a natural extension of the scalar speed function that accounts for the correlation between different features of $\bar{I}$ and allows for curved decision boundaries. In the case where all features are

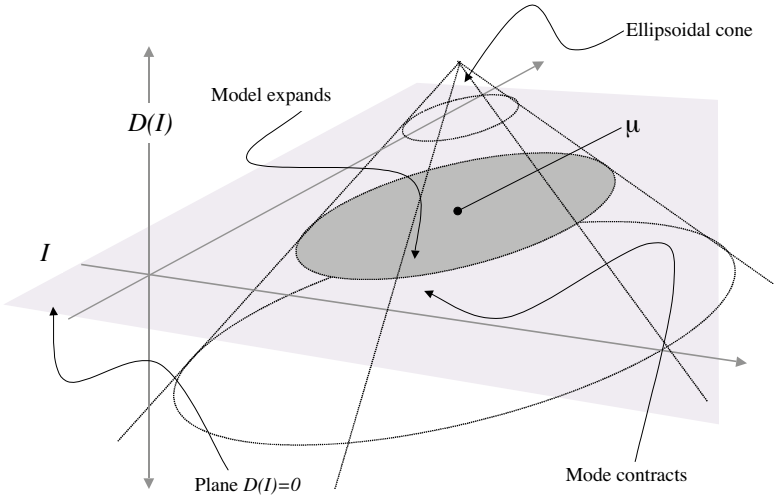


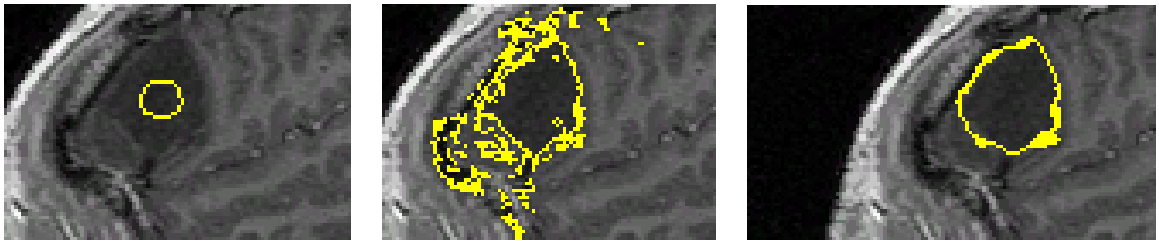
Figure 2: (a)A Mahalanobis distance classifier describes a mean and standard deviation of feature values that is normalized with respect to the covariance of the features. (b) Image values within the classifier range are mapped to a positive speed. Values outside the classifier range are mapped to a negative speed.

uncorrelated Mahalanobis distance is equivalent to Euclidean distance.

For three-channel data such as color RGB images, the covariance matrix and mean vector entail a total of *t*welve free parameters. Thus, it is not feasible to provide a user interface to interactively control all of the parameters associated with the multivariate form of this speed function. Instead, we allow the user to extract the mean and variance from a region of interest and provide the user with a single free parameter ε, which controls the size of the ellipsoidal region in the feature space and corresponds to the relative prior probabilities between the object and background.

3.4 The Role of Surface Curvature

If a user were to initialize a model in a volume and use the speed terms in Eqs. (2–7) without any curvature the results would be virtually the same as a simple flood fill over the region bounded by the upper and lower thresholds (or the ellipsoid in the *m*D case). However, the inclusion of the curvature term alleviates the critical *leaking* problem that arises when using flood filling as a segmentation technique. The leaking effect is particularly acute in 3D segmentations and is easily demonstrated on a brain tumor data set, as shown in Fig. 3. Furthermore in cases where the noise in the data corrupts the shapes of object boundaries, the curvature term can provide smoother results.



(a) (b) (c)

Figure 3: Showing one slice of a MRI volume: (a) The spherical initialization. (b) A model expands to fill the tumor but leaks through gaps and expands into other anatomy. (c) The same scenario with a degree of curvature prevents unwanted leaking. The level set isosurface is shown in yellow.

However, oversmoothing with curvature can significantly distort the shapes of segmented objects, and if the weight of the curvature is too large the model will pull away from the data and collapse to a point.

3.5 Rescaling the Distance Function

When solving the PDE associated with Eq. (1), the different level sets of the function ϕ will tend to spread out in some regions of the volume (due to the curvature term and numerical diffusion) and aggregate in other areas (due to the speed term). These phenomena are characterized by a decreasing or increasing of $|\nabla\phi|$ over time, respectively. Both of these tendencies will undermine the effectiveness of narrow-band algorithms, and therefore the literature describes mechanisms for maintaining ϕ with a relatively constant gradient magnitude. For instance, in [21], the authors stop the evolution of ϕ at regular intervals and establish a new ϕ that corresponds to the signed distance transform of the zero level set. In [22], the author updates the values of grid points around the zero-set of ϕ in layers that maintain an approximation to the signed distance.

For the GPU-based solver, neither of this strategies is appropriate, because they entail dynamic data structures that cannot be easily implemented in the streaming architecture. Instead we maintain $|\nabla\phi|$, by the addition of an extra term to the update equation (PDE) that governs the evolution of ϕ. This term will force the level sets of ϕ to spread out if the gradient is too large and to move

together if the gradient is too low. This *rescaling* term, G_r, is of the form

$$G_r = \phi \left(g_\phi - |\nabla \phi| \right), \tag{8}$$

where g_ϕ, the target magnitude for the gradient.

This rescaling term has several properties that are important to the implementation of GIST. First, the distance transform of the level set, scaled by g_ϕ, is formally (i.e. ignore points where $\nabla\phi$ is undefined) a fixed point of (8). Second, because G_r is proportional to ϕ, it does not affect the values of ϕ near the zero set, and therefore should not impact the evolution of the surface model. Finally, when G_r is implemented with the upwind scheme, it will maintain monotonicity, and therefore the fixed point of this term applied to updates of grid representing ϕ will be a clamped distance transform with extreme values limited by those in the initial conditions. Thus, G_r will maintain the narrow-band property, which is to say that ϕ will have $|\nabla\phi| \approx g_\phi$ within a narrow band around the zero set and $|\nabla\phi| \approx 0$ elsewhere.

4 Software Application Design

This section describes GIST, an interactive level-set segmentation tool, and the GPU implementation that makes it possible. It begins with a brief review of the GPU-based level-set solver (for a more complete description see [48]), describes the visualization of the volume data and surface models, and then describes the user interface to GIST.

4.1 GPU Level-Set Solver

The efficient solution of the level-set PDEs relies on only updating voxels that are on or near the isosurface. The sparse GPU level-set solver achieves this by decomposing the volume into a set of small 2D tiles (e.g. 16 x 16 pixels each). Only those tiles with non-zero derivatives are stored on the GPU (Fig. 4b). These *active* tiles are packed, in an arbitrary order, into a large 2D texture on the GPU. The 3D level-set PDE is computed directly on this compressed format. Because active

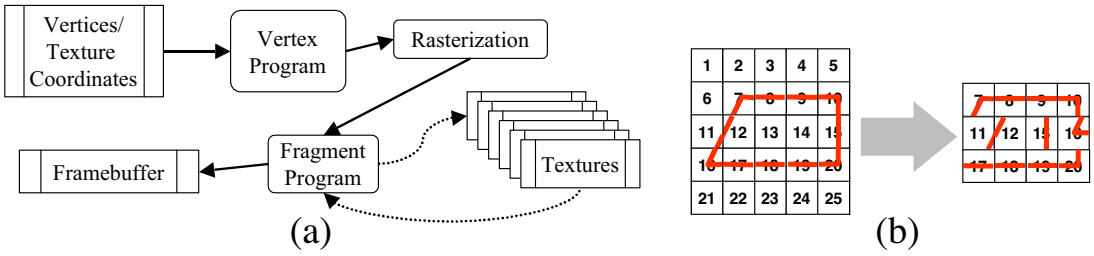


Figure 4: (a) The modern graphics processor computation pipeline. (b) The proposed method relies on packing active tiles into 2D texture—a compressed format.

tiles are identified by non-zero gradients, it is crucial that the volume in which the level-set surface is embedded, ϕ, resemble a clamped distance transform. In this way regions on or near the model will have finite derivatives, while tiles outside this narrow band will be flat, with derivative values of zero. Thus, the rescaling term given in Eq. (8) is particularly important.

For each PDE time step update, the 3D neighborhoods of all pixels in the active tiles must be sampled from the compressed 2D compressed format. For each active tile, the CPU sends texture coordinates, i.e. memory addresses, to the GPU for each of the tiles that share a side or an edge in the 3D volume. These texture coordinates are generated and maintained on the CPU. Using these texture coordinates, the GPU can perform neighborhood lookups to produce the complete set of partial derivatives (finite differences) used for the gradient and curvature calculations, which are in turn used to update values of ϕ.

After the level-set embedding is updated, the GPU uses built-in, hardware accelerated, *mipmapping* capabilities to create a bit vector image that summarizes the status of each tile. Each pixel in this coarse texture contains a bit code that identifies if that tile, as well as any of its six cardinal neighbors, need to be active for the next time step. This small image ($<=$ 64KB) is read back by the CPU and used to update the data structures that track the active volume regions. The texture coordinates are updated based on these structures and the next time step is computed.

This GPU-based level-set solver achieves a speedup of ten to fifteen times over a highly-optimized, sparse-field, CPU-based solver. All benchmarks were run on an Intel Xeon 1.7 GHz processor with 1 GB of RAM and an ATI Radeon 9700 Pro GPU. For the tumor segmentations performed in the user study, the GPU-based solver ran at 60-70 time steps per second while the

CPU version ran at 7-8 steps per second. The final steps of the cerebral cortex segmentation shown in Fig. 10 ran at 4 steps per second on the GPU and 0.25 steps per second on the CPU.

4.2 Interactive Visualization

An important aspect of GIST is the interactive visualization of the level-set surface, the volume data, and the speed function. This interactivity includes 2D slice-by-slice visualization of data, model, and speed, as well as 3D volume/surface rendering with user-controlled clipping planes to visualize and query the volume data.

GIST provides a simultaneous volume visualization of the input data with the evolving level-set model. The volume renderer associated with GIST performs a full 3D (transfer-function based) volume rendering of the greyscale data. For rendering the original volume, the input data and its gradient vectors are kept on the GPU as 3D textures. This GPU-based volume rendering incorporates multidimensional transfer functions as described in Kniss et al. [49]. The current implementation of GIST renders only scalar volume data, and thus for spectral data it renders only a derived scalar quantity (e.g. one component or magnitude). Future work will include the use of multidimensional transfer functions to directly render spectral data.

For rendering the evolving level-set model, we use a modification of the conventional 2D sliced approach to texture-based volume rendering [50]. The modification to the conventional approach is the rendering of the level-set solution directly from the packed tiles, which are stored as a single 2D texture. The level-set data and tile configuration is dynamic, and therefore does not require separate precomputed versions of the data (e.g. sliced along cardinal views) as is typically done with 2D texture approaches. Instead the renderer reconstructs these views, as needed, each time the volume is rendered. For efficiency, the renderer reuses data wherever possible. For instance, lighting for the level-set surface uses gradient vectors computed during the level-set update stage. The rendering of the source data relies on precomputed gradient data—the gradient magnitude is used by the transfer function and the gradient direction is used in the lighting model. More details on this design are given in [48].

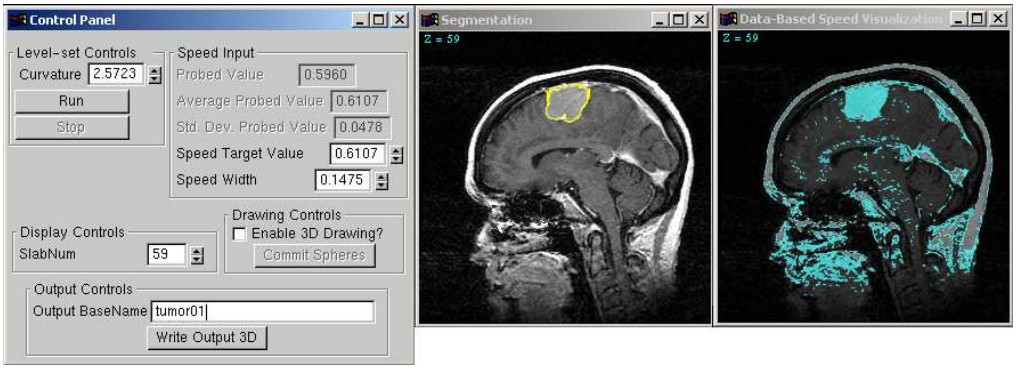


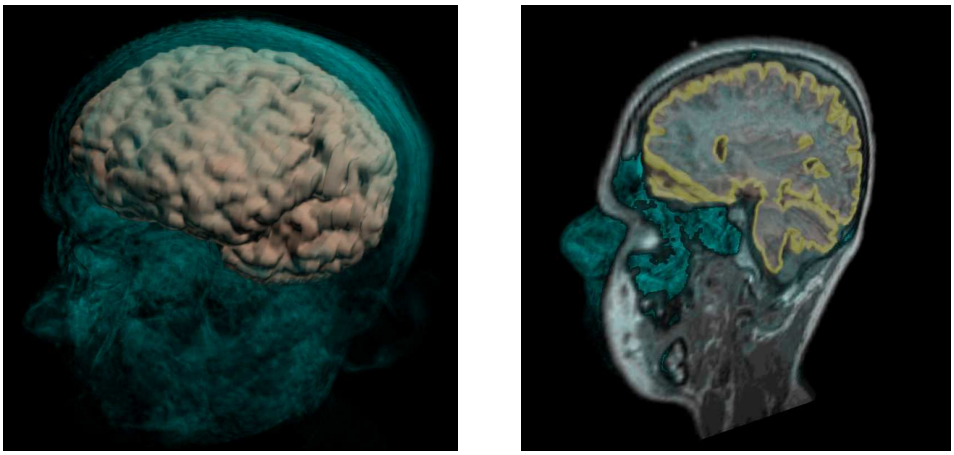
Figure 5: The main user interface of software application, called GIST. The center window shows a slice of an MRI volume overlaid by a brain tumor segmentation in progress. The right window displays the sign of the speed function.

4.3 Interface and Usage

GIST combines a graphical user interface (GUI), which controls the underlying GPU-based level-set solver, with a volume renderer. The GUI presents the user with two volume slices, a 3D rendering window, and a control panel. The first slice window displays the current segmentation as a yellow line overlaid on top of the target data. The second slice viewing window displays a visualization of the speed function that use color to clearly delineate the positive and negative regions. The GUI has controls for scrolling through image slices, starting and stopping the solver, and saving the 3D segmentation to file. The user can also query data values in the slice viewer and create spherical surface models to use as initializations to the level-set solver. A screen capture of the slice-based interface is shown in Fig. 5.

To set the free parameters of the speed function, the user samples image values by clicking and dragging the mouse in regions of interest through the 2D slice view window (center window of 5). As the user gathers statistical samples, GIST simultaneously updates the mean value and the variance or covariance that defines the shape of the classifier. A user will typically probe the object across a range of slices for a better representative sampling than can be obtained in just one image slice. The remaining speed function parameter ε is set manually in the GUI. The speed function is updated and displayed in real time as parameters are modified to guide the process.

The volume renderer window displays a 3D rendering of the source data and a surface rendering



(a) (b)

Figure 6: Two views of the volume rendering window from GIST. A brain cortex segmentation is shown at left with a cutting plane applied to the rendering on the right. The intersection of the level-set surface with the cutting plane is shown as a yellow band.

of the evolving level-set model. The opacity of each rendering can be controlled by the user. A clipping plane with the original data can also be applied to the rendering in any orientation and position. All of the interactions available in the 2D slice view are also available on the clipping plane, e.g. the user can probe data to set the speed term parameters and draw spheres for initializing the model directly into the 3D view. The intersection of the level-set solution with the clipping plane is shown as a yellow band. Figure 6 shows two views from the volume rendering window.

For a typical segmentation GIST, a user scrolls through slices until they find the location of the target object and then queries values with the mouse to set the speed function parameters. Next, the user creates an initial model by drawing one or more spheres within the object and then starts the solver. The user scrolls through slices as the model begins to deform, observing its behavior and modifying curvature (model smoothness) and classifier width as needed. The user may also stop the solver and resample the data to either refine or replace the current statistical speed function parameters. Using the immediate feedback they get on the behavior of the model, they continue modifying parameters until the model boundaries appear to align with those of the tumor. In a typical 5 minute session, a user may modify the model parameters between 10 and 30 times.

5 Results

This section presents results from the application of our GPU-based level-set segmentation tool to a range of scalar and spectral data. The evaluations in this section include qualitative and quantitative comparisons with hand contouring as well as two other user-assisted methods. We choose hand contouring as the main focus of the comparison for several reasons. First, it is (like the proposed method) a general purpose segmentation method. Second, the field at large considers hand contouring (by experts) to be the de facto gold standard. Third, hand contouring is, in many cases, the state of the art. That is, a large number of clinical applications that require image segmentation still rely on hand contouring as their primary segmentation technique.

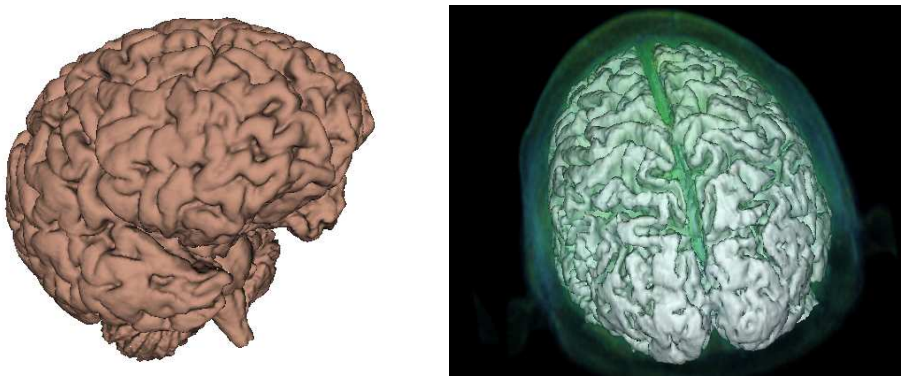
Section 5.1 gives a qualitative analysis of several anatomical segmentations from MRI and color cryosection data. A more rigorous, quantitative evaluation is presented in Sect. 5.2, which describes a user study of our software and compares results of brain tumor segmentations with ground truth obtained from experts.

5.1 Qualitative Evaluation

As a preliminary evaluation of our segmentation tool, we segment a variety of anatomical structures in several imaging modalities: scalar and spectral MRI, and color cryosection data from the Visible Human Female (VHF) [51]. This section presents results and discussion of those segmentations.

Figure 7(a) is a rendering of a cortical brain surface segmentation from a 256 x 256 x 175 MRI volume. The complete segmentation required no preprocessing (e.g. no filtering) of the data and required five minutes using the one-dimensional classifier speed function with a small, spherical surface (placed by the user) as the initial model. This type of segmentation is impractical to compute on ordinary, CPU-based solvers because of the size and complexity of the solution. In our experience with state-of-the art CPU-based solvers (e.g. see the Insight Toolkit, *www.itk.org*) the same cortical segmentation typically takes more than an hour.

For MRI spectral data, we use volumes consisting of co-registered T1, T2, and proton density



(a) (b)

Figure 7: Renderings of (a) brain cortical surface from scalar MRI and (b) white matter from spectral MRI show qualitatively good results on large, complex structures computed at interactive rates.

data. This combination of image modalities requires the three-dimensional classifier, given in Eq. (7), to take full advantage of the wider spectrum of information. Figure 7(b) shows a rendering of a segmentation of the white matter of the brain. As with the corticle segmentation, the results are encouraging because they can be obtained in only a few minutes (no preprocessing) with a simple spherical *seed point* initialization. We have seen similarly promising results with our tool segmenting skin and skull tissue from spectral MRI data.

For the VHF color cryosection data, we use a region of interest (cropped volume) from the head that contains two interesting structures: the right lateral rectus muscles, and anterior portions of the right and left optic nerves. The texture information in this data set posed a significant challenge, and therefore we preprocessed the data by smoothing with 10 iterations of modified-curvature diffusion [52]. This diffusion step blurs the more homogeneous regions of the data while preserving object boundaries. This nonlinear diffusion is relatively computationally expensive, especially on spectral data, and it required approximately 20 minutes of computation on a two-processor Pentium IV desktop machine.

Figure 8 presents the results of our VHF anatomical segmentations and compares them with results obtained using other general-purpose segmentation methods. Column (a) shows a single slice of the original data with the target object for segmentation highlighted. Column (b) is a surface rendering of the results using our level-set tool. The results from (b) are overlaid on the slice

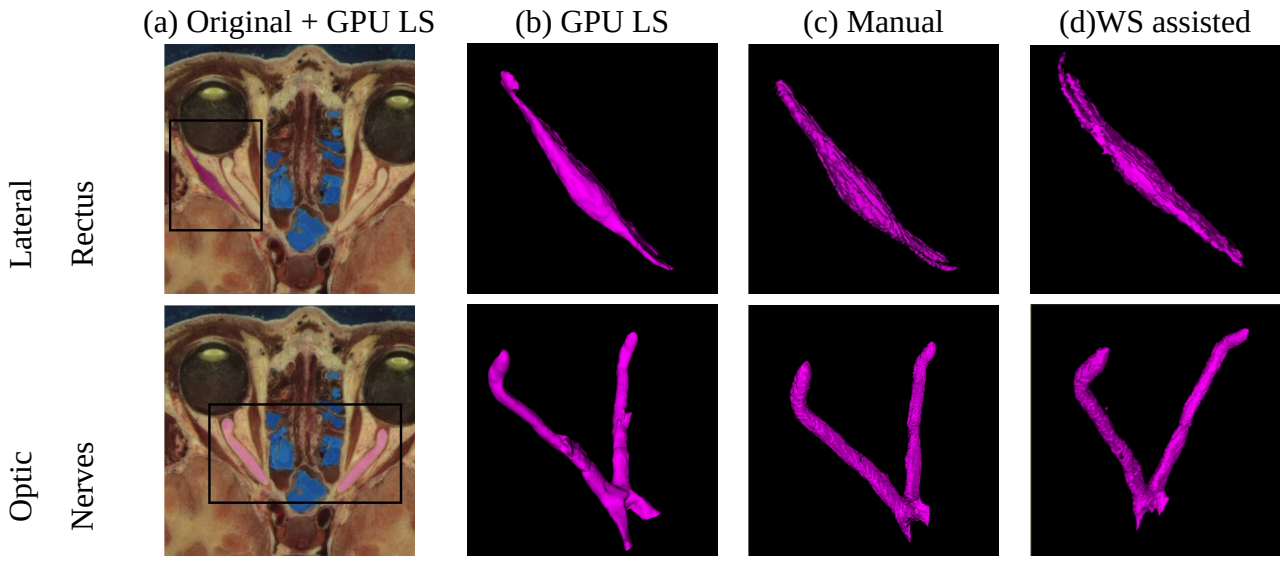


Figure 8: Visual comparison of surface renderings of GPU level-set (b) and manual (c) segmentations of the Visible Human Female color cryosection anatomy. The targeted anatomical structure is highlighted in column (a), which shows the segmentation from (b) superimposed over a transverse slice through the original color data. Column (d) is a comparison with the user-assisted watershed technique.

in column (a). Expert segmentations of the same structures are shown in 8(b). The expert segmentations were obtained from multiple operators at Harvard Brigham and Women's Hospital and at the University of Utah using the Slicer Tool [53]. The renderings shown in (b) are of composite ground-truth volumes created with the STAPLE method described in Sect. 5.2. Column (c) shows results obtained using another general, interactive segmentation method based on morphological watersheds segmentation method (for details see [54]).

Visual inspection of the GPU level-set results show them to be of similar quality as the hand-contour and watershed results. Anterior and posterior sections optic nerves in the area of the optic chiasm are segmented separately in this example and combined prior to rendering. The current speed function implementation in GIST is limited to a single statistical feature profile, and therefore distinct structures in color space such as the optic chiasm must be segmented separately.

Because of the curvature term in (1), segmentations created using our tool are naturally anti-aliased. The level-set technique also tends to produce a smoother boundary in the axial direction

than the hand-contour and watershed methods, which tend to more resemble a stack of 2D slices with poor continuity of the boundary in the axial direction. The greatest advantage to the GPU level-set segmentation is its relative efficiency. The time taken for the VHF segmentations are up to 20 times faster than hand-contouring (several minutes versus up to several hours) and were up to 6 times faster than using the watersheds method (full processing time). As the following section will demonstrate, the level-set segmentation tool can generally produce acceptable results on the raw image data, which is not possible with many other algorithms, such watershed segmentation; therefore the level-set segmentation tool is particularly useful for fast, impromptu segmentations of 3D data sets.

5.2 User Study

5.2.1 Motivation

The purpose of this study is to determine if our level-set tool can produce volumetric delineations of brain tumor boundaries comparable to those done by experts (e.g. radiologists or neurosurgeons) using traditional hand-contouring. We apply our method to the problem of brain tumor segmentation using data from the *Brain Tumor Segmentation Database*, which is made available by the Harvard Medical School at the Brigham and Women's Hospital (HBW) [1, 2]. The HBW database consists of ten 3D 1.5T MRI brain tumor patient datasets selected by a neurosurgeon as a representative sampling of a larger clinical database. For each of the ten cases, there are also four independent expert hand segmentations of one randomly selected 2D *slice* in the region of the tumor.

We use nine cases for our study: three meningioma (cases 1-3) and 6 low grade glioma (4-6, 8-10). One case, number 7, is omitted because a quick inspection shows that its intensity structure is too complicated to be segmented by the proposed tool—such a problem remains as future work, as we will discuss in Sect. 6. For this study, there is *no preprocessing on the data* and there are no hidden parameters in this study—all parameters in our system are set by the users in real time, as

they interact with the data and the models.

The subjects consist of five people from among the staff and students in our group who have each been given a brief introduction on how to use the application. During the study, each user is asked to delineate the full, 3D boundaries of the tumor in each of the nine selected cases. We set no time limit on the users and record their time to complete each tumor. None of our users are experts in reading radiological data. It is not our intention to test for tumor recognition (tissue classification), but rather to test whether parameters can be selected for our algorithm to produce a segmentation which mimics those done by the experts. To control for tumor recognition, we allow each user to refer to a single slice from an expert segmentation. Users are told to treat this hand segmentation slice as a guide for understanding the difference between tumor and non-tumor tissue. Our assumption is that an expert would not need such an example.

5.2.2 Aggregation of Expert Segmentation Data

The expert data serves two purposes in the this study. First, it provides a mechanism for establishing a ground truth, against which we can compare the level-set segmentation. Second, the set of expert segmentations establish a performance benchmark for the accuracy, precision, and efficiency of hand contouring.

Ground truth is established from manual segmentations by the experts using the STAPLE algorithm [2], an iterative EM algorithm that accounts for systematic biases in the behavior of experts. The STAPLE algorithm generates a fuzzy ground truth as well as sensitivity and specificity parameters for each expert and each case.

We denote a single subject within a population with the subscript j and the pixels within the image/volume as i. An image of binary values D_{ij} represents a segmentation for a particular subject. Given sensitivities p_j and specificities q_j for each subject, the degree of confidence that a particular pixel is in the target object is

$$W_i = \frac{g_i \alpha_i}{g_i \alpha_i + (1 - g_i)\beta_i}, \tag{9}$$

where g_i is the prior probability that any pixel would be classified as inside the target object (usually taken to be the fraction of the image that is filled by the object). The values of α and β are

$$\alpha = \left[\Pi_j p_j D_{ij}\right]\left[\Pi_j (1-p_j)(1-D_{ij})\right] \text{ and } \beta = \left[\Pi_j q_j (1-D_{ij})\right]\left[\Pi_j (1-q_j) D_{ij}\right]. \quad (10)$$

Given a probability image W_i, the sensitivity/specificity for each subject can be updated as

$$p_j = \frac{\sum_i W_i D_{ij}}{\sum_i W_i} \text{ and } q_j = \frac{\sum_i (1-W_i)(1-D_{ij})}{\sum_i (1-W_i)}. \quad (11)$$

The full STAPLE algorithm entails iterating on these updates, back and forth between (p,q) and W, until the process converges.

Accuracy is evaluated against aggregate volumes created for each segmented object by applying the STAPLE algorithm to the expert hand-contours. These aggregate (STAPLE) volumes consist of a graded membership function (zero to one). We analyze the accuracy of the experimental, level-set results by evaluating the sensitivity and specificity of each experimental subject, using Eq. (11), relative to these aggregate volumes. We can then make comparisons by computing average sensitivity and specificity for the two groups—subjects using hand contouring and subjects using the level-set GUI. Additionally, we can combine values of p_j and q_j to compute a total correct fraction for a subject:

$$c_j = \frac{\sum_i W_i D_{ij} + \sum_i (1-W_i)(1-D_{ij})}{\sum_i 1}. \quad (12)$$

Ideally we would compute accuracy of hand-contour segmentations using aggregate data from an *independent* group of expert segmenters. A characterization of the accuracy of a small group of manual segmentations using ground truth generated as a complete aggregate of *those same segmentations* contains an clear bias that over estimates the accuracy of the expert segmentations. A second, less conservative measurement that produces a more unbiased estimate of the manual segmentation accuracy is a round-robin *leave-one-out* strategy, [55], where p, q, and c values for

each D_{ij} are computed using W_k generated by all segmentations $k \neq j$.

Accuracy metrics must be interpreted carefully. Note that where a segmentation technique shows high sensitivity, there is a high confidence level in the results it produces for *negatively* classified pixels, and where a technique shows high specificity, there is a high confidence level for *positively* classified pixels. The magnitudes of p and q are incommensurate because they are percentages of different populations of pixels. Total correct fraction is particularly difficult to interpret because it is biased by the ratio of the size of the image volume to the size of the target object. Where this ratio is high, c approaches q. Where the ratio is low, c approaches p. Total correct fraction is used in this study only as a way to compare our results with other published results on the same data.

We quantify precision in this study using the *similarity* s_{jk} of results from subjects j and k,

$$s_{jk} = \frac{2\sum_i D_{ij} D_{ik}}{\sum_i D_{ij} + D_{ik}}, \tag{13}$$

and average similarity across all pairs of subjects $j \neq k$. Accuracy, precision, and efficiency metrics were also applied *across* subjects. Given the limited resources for this study and the scarcity of manually segmented data, we were not able to make *intra*-subject comparisons, which require multiple segmentations from the same subject.

5.2.3 Discussion and Analysis

Figure 9 shows graphs of average p, q, and c values for the experts and the users in our study. Error bars represent the standard deviations of the associated values. This figure shows the average accuracy across all experts using round-robin ground truth.

The performance of the experts and our users varies case by case, but in almost all cases the performance of our users was within the range of performances of the experts. A comparison with expert-biased ground truth shows similar results. The average correct fraction of our users was better than the experts in 6 out of 9 cases. A general trend is that our users tended to underestimate

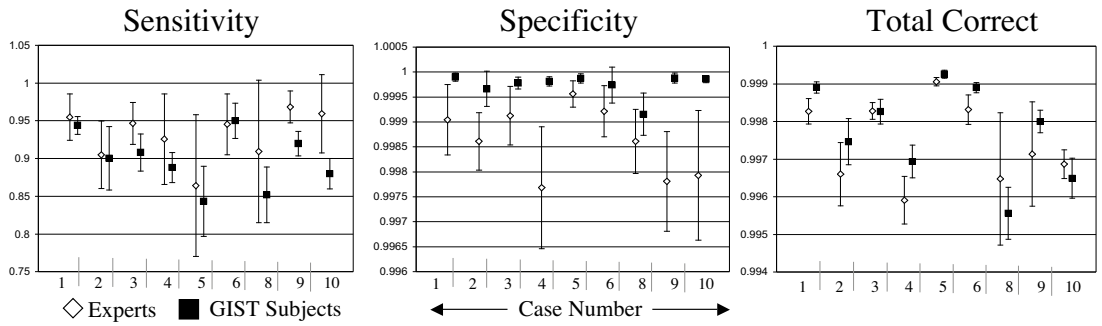
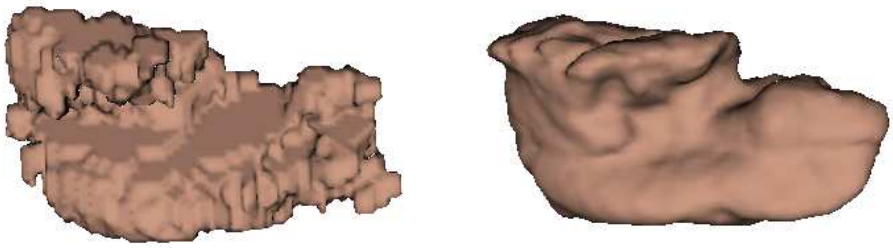


Figure 9: A comparison of user study results (GPU LS) with expert and expert round-robin (RR) results reveal an overall comparable performance with a tendency to underestimate the region of tumor.



(a) (b)

Figure 10: (a) An expert hand segmentation of a tumor from the HBW database shows significant interslice artifacts. (b) A 3D segmentation of the same tumor from one of the subjects in our study.

the tumor relative to the experts, as indicated by lower values of p and higher values of q, especially when compared to the round-robin expert averages. This is consistent with our experiences with hand segmentations and level set models— with hand contouring users tend to overestimate structures, and with level sets the curvature term tends to reduce the size of convex structures.

The segmentations in our study show a much higher degree of precision than the expert hand segmentations. Mean precision [37] across all users and cases was $94.04\% \pm 0.04\%$ while the mean precision across all experts and cases was $82.65\% \pm 0.07\%$. Regarding efficiency, the average time to complete a segmentation (all users, all cases) was 6 ± 3minutes. Only $5\% - 10\%$ of this time is spent processing the level-set surface. This compares favorably with the 3-5 hours required for a typical 3D segmentation done by hand.

The accuracy and precision of subjects using our tool also compares well with the automated brain tumor segmentation results of Kaus, et al. [1], who use a superset of the same data used in our study. They report an average correct volume fraction of $99.68\% \pm 0.29\%$ (using the expert-biased

ground truth), while the average correct volume fraction of our users was $99.78\% \pm 0.13\%$. Their method required similar average operator times (5-10 minutes), but unlike the proposed method their classification approach required subsequent processing times of approximately 75 minutes. That method, like many other segmentation methods discussed in the literature, includes a number of hidden parameters, which were not part of their analysis of timing or performance.

These quantitative comparisons with experts pertain to a only single 2D slice that was extracted from the 3D segmentations. This is a limitation due to the scarcity of expert data. Our experience is that computer-aided segmentation tools perform relatively better for 3D segmentations because the hand contours typically show signs of interslice inconsistencies and fatigue. Figures 10a–b show a segmentation by an expert with hand contouring compared with a segmentation done by one of our subjects.

6 Summary and Conclusions

A careful implementation of real-time visualization and a sparse level-set solver on a GPU provides a new tool, called GIST, for interactive 3D segmentation. Users can manipulate several parameters simultaneously in order to find a set of values that are appropriate for a particular segmentation task. The quantitative results of using this tool for brain tumor segmentation suggest that it is compares well with hand contouring and state-of-the-art automated methods. However, the tool as built and tested is quite general, and it has no hidden parameters. Thus, the same tool can be used to segment a variety of anatomy as was show in Sect. 5.1.

The current limitations are mostly in the speed function and the interface. The speed function used in this paper is quite simple and easily extended, within the current framework, to include image edges and more complicated statistical classifiers. Future work will include development of a more intuitive 3D interface that could potentially improve user interaction times and accuracy.

References

[1] Kaus, M., Warfield, S.K., Nabavi, A., Black, P.M., Jolesz, F.A., Kikinis, R.: Automated segmentation of mri of brain tumors. Radiology **218** (2001) 586–591

[2] Warfield, S.K., Nabavi, A., Butz, T., Tuncali, K., Silverman, S.G.: Intraoperative segmentation and nonrigid registration for image guided therapy. In: International Conference on Medical Image Computing and Computer-Assisted Intervention MICCAI'2000. Lecture Notes in Computer Science, Cambridge, England, Springer Verlag (2000) 176–185

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

[4] Sethian, J.A.: Level Set Methods and Fast Marching Methods Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science. Cambridge University Press (1999)

[5] Fedkiw, R., Osher, S.: Level Set Methods and Dynamic Implicit Surfaces. Springer (2002)

[6] Miller, J., Breen, D., Lorensen, W., O'Bara, R., Wozny, M.: Geometrically deformed models: A method for extracting closed geometric models from volume data. Computer Graphics (SIGGRAPH '91 Proceedings) **25** (1991) 217–226

[7] Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. International Journal of Computer Vision **1** (1987) 321–323

[8] Whitaker, R.T.: Volumetric deformable models: Active blobs. In Robb, R.A., ed.: Visualization In Biomedical Computing 1994, Mayo Clinic, Rochester, Minnesota, SPIE (1994) 122–134

[9] Malladi, R., Sethian, J.A., Vemuri, B.C.: Shape modeling with front propogation: A level set approach. IEEE Trans. on Pattern Analysis and Machine Intelligence **17** (1995) 158–175

[10] Caselles, V., Kimmel, R., Sapiro, G.: Geodesic active contours. In: Fifth International Conference on Computer Vision, IEEE, IEEE Computer Society Press (1995) 694–699

[11] Chesnaud, C., Rfrgier, P., Boulet, V.: Statistical region snake-based segmentation adapted to different physical noise models. IEEE Trans. on Pattern Analysis and Machine Intelligence **21** (1999) 1145–1156

[12] Radeva, P., Vitria, J.: Region-based approach for discriminant snakes. In: Proc. Int. Conf. on Image Processing. (2001) 801–804

[13] Tsai, A., Yezzi, A., Willsky, A.: A curve evolution approach to smoothing and segmentation using the mumford-shah functional. In: Proceedings of IEEE Conf. on Comp. Vision and Pattern Recognition. (2000) 119–124

[14] Chan, T.F., Vese, L.: A level set algorithm for minimizing the mumford-shah functional in image processing. In: Proc. 1st IEEE Workshop on Variational and Level Set Methods in Computer Vision. (2001) 161–168

[15] Whitaker, R., Elangovan, V.: A direct approach to estimating surfaces in tomographic data. Medical Image Analysis **6** (2002) 235–249

[16] Lorigo, L., Faugeras, O., Grimson, E., Keriven, R., Kikinis, R., Nabavi, A., Westin, C.F.: Co-dimension 2 geodesic active contours for the segmentation of tubular strucures. In: Proceedings of IEEE Conf. on Comp. Vision and Pattern Recognition. (2000) 444–452

[17] Tasdizen, T., Whitaker, R., Burchard, P., Osher, S.: Geometric surface processing via normal maps. ACM Trans. on Graphics (2003) 1012–1033

[18] Tasdizen, T., Whitaker, R.: Higher-order nonlinear priors for surface reconstruction. *Submitted to:* IEEE Trans. on PAMI (2004)

[19] Osher, S., Sethian, J.: Fronts propogating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. Journal of Computational Physics **79** (1988) 12–49

[20] Droske, M., Meyer, B., Rumpf, M., Schaller, C.: An adaptive level set method for medical image segmentation. In Leahy, R., Insana, M., eds.: Proc. of the Annual Symposium on Information Processing in Medical Imaging, Springer, Lecture Notes Computer Science (2001)

[21] Adalsteinson, D., Sethian, J.A.: A fast level set method for propogating interfaces. Journal of Computational Physics (1995) 269–277

[22] Whitaker, R.T.: A level-set approach to 3D reconstruction from range data. International Journal of Computer Vision **29** (1998) 203–231

[23] Peng, D., Merriman, B., Osher, S., Zhao, H., Kang, M.: A pde based fast local level set method. J. Comput. Phys. **155** (1999) 410–438

[24] Owens, J.D.: Computer Graphics on a Stream Architecture. PhD thesis, Stanford University (2002)

[25] Goodnight, N., Lewin, G., Luebke, D., Skadron, K.: A multigrid solver for boundary value problems using graphics hardware. University of Virginia Technical Report CS-2003-03 (2003)

[26] Larsen, E.S., McAllister, D.: Fast matrix multiplies using graphics hardware. In: SC'2001 Conference CD, Denver, ACM SIGARCH/IEEE (2001) UNC.

[27] Strzodka, R., Rumpf, M.: Using graphics cards for quantized FEM computations. In: Proceedings VIIP Conference on Visualization and Image Processing. (2001)

[28] Rumpf, M., Strzodka, R.: Level set segmentation in graphics hardware. In: International Conference on Image Processing. (2001) 1103–1106

[29] Sherbondy, A., Houston, M., Nepal, S.: Fast volume segmentation with simultaneous visualization using programmable graphics hardware. In: IEEE Visualization. (2003) To Appear

[30] Bolz, J., Farmer, I., Grinspun, E., Schröder, P.: Sparse matrix solvers on the GPU: Conjugate gradients and multigrid. In: ACM Transactions on Graphics. Volume 22. (2003) 917–924 (Proceedings of ACM Siggraph 2003).

[31] Krüger, J., Westermann, R.: Linear algebra operators for GPU implementation of numerical algorithms. In: ACM Transactions on Graphics. Volume 22. (2003) 908–916 (Proceedings of ACM SIGGRAPH 2003).

[32] Beers, A.C., Agrawala, M., Chaddha, N.: Rendering from compressed textures. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, ACM Press (1996) 373–378

[33] Kraus, M., Ertl, T.: Adaptive texture maps. In: Graphics Hardware 2002. (2002) 7–16

[34] Malpica, N., Solórzano, C., Vaquero, J., Santos, A., Vallcorba, I., García-Sagredo, J., Pozo, F.: Applying watershed algorithms to the segmentation of clustered nuclei. Cytometry **28** (1997) 289–297

[35] Sijbers, J., Verhoye, M., Scheunders, A., Van der Linden, A., Van Dyck, D., Raman, E.: Watershed-based segmentation of 3d mr data for volume quantization. Magnetic Resonance Imaging **15** (1997) 679–688

[36] Yoo, T.S., Ackerman, M.J., Vannier, M.: Toward a common validation methodology for segmentation and registration algorithms. In: International Conference on Medical Image Computing and Computer-Assisted Intervention MICCAI'2000. Lecture Notes in Computer Science, Cambridge, England, Springer Verlag (2000) 422–431

[37] Udupa, J.K., Leblanc, V., Schmidt, H., Imielinska, C., Saha, K., Grevera, G., Zhuge, Y., Molholt, P., Currie, L., Jin, Y.: A Methodology for Evaluating Image Segmentation Algorithms. In: SPIE Medical Imaging, San Diego (2002)

[38] Chalana, V., Yongmin, K.: A methodology for evaluation of boundary detection algorithms on medical images. IEEE Trans. Medical Imaging **16** (1997) 642–652

[39] Gerig, G., Jomier, M., Chakos, M.: Valmet: A new validation tool for assessing and improving 3d object segmentation. In: MICCAI 2001: Fourth International Conference on Medical Image Computing and Computer-Assisted Intervention, Heidelberg, Germany, Springer-Verlag (2001) 516–528

[40] Zijdenbos, A., Dawant, B., Margolin, A.: Morphometric analysis of white matter lesions in mr images: Method and validation. IEEE Transactions on Medical Imaging **13** (1994) 716–724

[41] Lefohn, A., Cates, J., Whitaker, R.: Interactive, gpu-based level sets for 3d brain tumor segmentation. In: MICCAI 2003. (2003) To appear

[42] Jannin, P., Fitzpatrick, J., Hawkes, D., Pennec, X., Shahidi, R., Vannier, M.: Validation of medical image processing in image-guided therapy. IEEE Trans. on Medical Imaging **21** (2002) 1445–1449

[43] Prastawa, M., Bullitt, E., Gerig, G.: Robust estimation for brain tumor segmentation. In: MICCAI 2003, Heidelberg, Germany, Springer-Verlag (2003)

[44] Warfield, S., Zou, K., Wells, W.: Validation of image segmentation and expert quality with an expectation-maximization algorithm. In: MICCAI 2002: Fifth International Conference on Medical Image Computing and Computer-Assisted Intervention, Heidelberg, Germany, Springer-Verlag (2002) 298–306

[45] Huttenlocher, D., Klanderman, G., Rucklidge, W.: Comparing images using the hausdorff distance. IEEE Transactions on Pattern Analysis and Machine Intelligence **15** (1993) 850–863

[46] Leemput, K., Maes, F., Vandermeulen, D., Suetens, P.: Automated model-based tissue classification of mr images of the brain. IEEE Transactions on Medical Imaging **18** (1999) 897–908

[47] Whitaker, R.T.: Geometry-Limited Diffusion. PhD thesis, The University of North Carolina, Chapel Hill, North Carolina 27599-3175 (1993)

[48] Lefohn, A., Kniss, J., Hansen, C., Whitaker, R.: A streaming narrow-band algorithm: Interactive deformation and visualization of level sets. IEEE Transactions on Visualization and Computer Graphics (2004) To appear

[49] Kniss, J., Kindlmann, G., Hansen, C.: Multi-Dimensional Transfer Functions for Interactive Volume R endering. TVCG **8** (2002) 270–285

[50] Cabral, B., Cam, N., Foran, J.: Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware. In: ACM Symposium On Volume Visualization. (1994)

[51] Ackerman, M., Yoo, T., Jenkins, D.: The visible human project: from data to knowledge. In: Proceedings of CARS2000, Amsterdam, Elsevier Press (2000) 11–16

[52] Whitaker, R., Xue, X.: Variable-conductance, level-set curvature for image denoising. Proceedings. 2001 International Conference on Image Processing **3** (2001) 142–145

[53] MIT Artificial Intelligence Laboratory and the Surgical Planning Laboratory at Brigham and Women's Hospital http://www.slicer.org: 3D Slicer Users Guide. (2004)

[54] Cates, J., Whitaker, R., Jones, G.: Case study: Case study: An evaluation of user-assisted hierarchical watershed segmentation. Medical Image Analysis (2004) Under review.

[55] Tou, J., Gonzalez, R.: Pattern Recognition Principles. Addison-Wesley (1974)