Interactive, GPU-Based Level Sets for 3D Brain Tumor Segmentation

Aaron Lefohn	Joshua Cates	Ross Whitaker
School of Computing	School of Computing	School of Computing
Univ. of Utah	Univ. of Utah	Univ. of utah
lefohnae@cs.utah.edu	cates@cs.utah.edu	whitaker@cs.utah.edu

University of Utah, School of Computing
Technical Report UUCS-03-004

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

April 16, 2003

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 presents a tool for 3D segmentation that relies on level-set surface models computed at interactive rates on commodity graphics cards (GPUs). The mapping of a level-set solver to a GPU relies on a novel mechanism for GPU memory management. The interactive rates for solving the level-set PDE give the user immediate feedback on the parameter settings, and thus users can tune three separate parameters and control the shape of the model in real time. We have found that this interactivity enables users to produce good, reliable segmentations. To support this observation, this paper presents qualitative and quantitative results from a study of brain tumor segmentation.

1 Introduction

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 parameter space when an example result from a point in that space requires minutes or hours to generate.

The proposed tool 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. The tool can be applied to a general set of applications by tuning three free parameters. Despite its general nature, we demonstrate the effectiveness of this tool 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 15 times faster than previous solutions).
- A mapping of the sparse, level-set computation to a GPU, a new numerical scheme for retaining a thin band structure in the solution, and a novel technique for dynamic memory management between the CPU and GPU.
- Quantitative and qualitative evidence that interactive level-set models are 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 and GPUs. Section 3 describes our GPU level-set solver, Sect. 4 presents the results of our study on brain tumor segmentation, and Sect. 5 summarizes our 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*. 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 [3]. Here we merely review the notation and describe the particular formulation that is relevant to this paper.

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 [4, 5]

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

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. [6].

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.

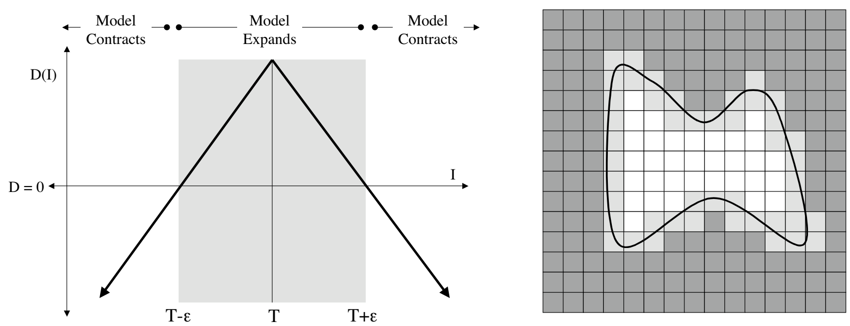
For the work in this paper we have chosen a very simple speed function to demonstrate the effectiveness of *interactivity* in level-set solvers. The speed function at any one point is based solely on the input intensity I at that point:

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

where T controls the brightness 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 $T - \epsilon$ and $T + \epsilon$, 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 $T \pm \epsilon$ range. Even with this simple scheme a user would have to specify three free parameters, T, ϵ, and α, *as well as* an initialization. This speed term is a simple approximation to a one-dimensional statistical classifier, which assumes a single density (with noise) for the regions of interest.

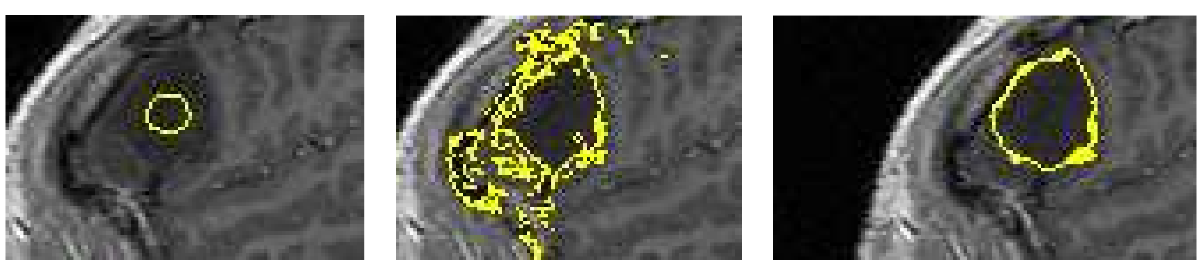
If a user were to initialize a model in a volume and use the speed term in eq (2) without any curvature the results would be virtually the same as a simple flood fill over the region bounded by the upper and lower thresholds. However, the inclusion of the curvature term alleviates the critical *leaking* 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 figure 2.

The purpose of this paper is not to advocate for any one level-set formulation or speed function, but rather to address an issue that is relevant to virtually all level-set segmentation strategies; that is, a good segmentation depends on a proper specification of free parameters and the initialization.



(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.



(a) (b) (c)

Figure 2: 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.

Solving level-set PDEs on a volume requires proper numerical schemes [7] 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 eq (1) in which the surface motion is strictly inward or outward. In such cases (1) can be solved somewhat efficiently using the *fast marching method* [3] and variations thereof [8]. 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 require a curvature term and simultaneously require the model to expand and contract to match the data.

Efficient algorithms for solving the more general equation 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 (near points where $\phi = 0$). 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 Figure 1b. Adalsteinson and Sethian [9] have proposed the *narrow band method*, which updates the embedding, ϕ, on a band of 10-20 pixels around the model, and reinitializes that band whenever the model approaches the edge. Whitaker [10] 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. Even with this very narrow band of computation, updates rates using conventional processors on typical medical data sets (e.g. 256^3 voxels) are not interactive. This is the motivation behind our GPU-based solver.

2.2 Graphics Processing Units

GPUs 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[11] 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.

Currently GPUs must be programmed via graphics APIs such as OpenGL[12] or DirectX[13]. All computations must, therefore, be cast in terms of computer graphics primitives such as vertices, textures, texture coordinates, etc. Figure 3a depicts the computation pipeline of a typical GPU. A *render pass* is a set of data passing completely through this pipeline. It can

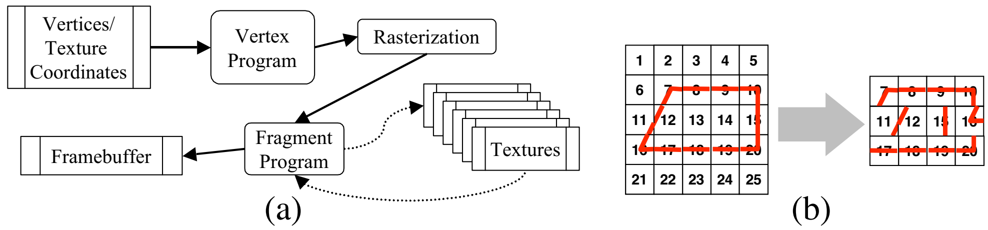


Figure 3: (a) The modern graphics processor computation pipeline. (b) The proposed method relies on packing active tiles into 2D texture—a compressed format.

also be thought of as the complete processing of a stream by a given kernel.

Grid-based computations, such as the level-set partial differential equations (PDEs), are solved by first transferring the initial data into texture memory and then rendering a graphics primitive that relies on this texture. In the simplest case, a 2D slice is computed by drawing a quadrilateral that has the same number of grid points (pixels) as the texture. Memory addresses that identify each fragment's data value as well as the location of its neighbors are given as texture coordinates. A fragment program then uses these addresses to read data from texture memory, perform the computation, and write the result back to memory. A 3D grid is processed by repeating the above process on each 2D slice. This computation model has been used by a number of researchers to map a wide variety of demanding problems to the above computation paradigm. Examples include matrix multiplication, finite element methods, Navier-Stokes solvers, and others[14, 15, 16]. All of these examples demonstrate a homogeneous sequence of operations over a densely populated grid structure.

Rumpf *et. al.*[17] were the first to show that the level-set equations could be solved using a graphics processor. Their solver implemented the 2D level-set method using a time-invariant speed function for flood-fill-like image segmentation without the associated curvature. The authors have demonstrated a full 3D level-set solver, with curvature, running on a graphics processor[18]. Neither of these approaches take advantage of the sparse nature of level-set PDEs and, therefore, they demonstrated results that are only marginally better (e.g. twice as fast) than sparse-field CPU implementations.

This paper presents a GPU computational model that supports *sparse* grid problems. Sparse 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. Section 3 describes how our design addresses these challenges.

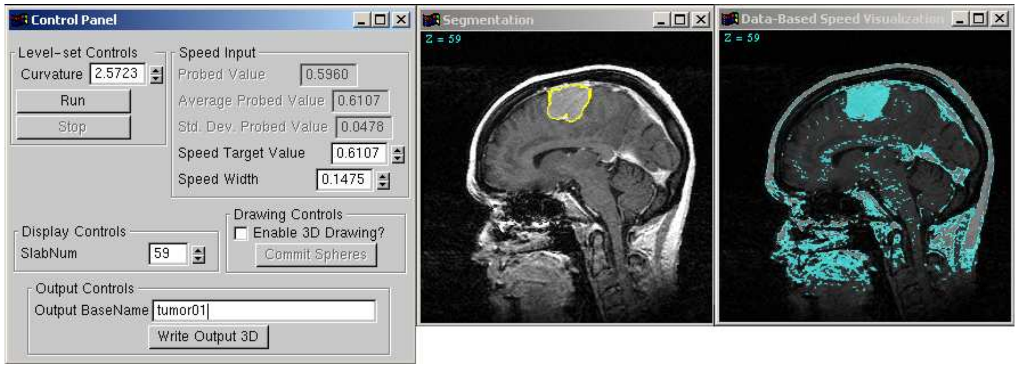


Figure 4: The user interface of our segmentation application. The center window shows a slice of the MRI volume overlaid with the current segmentation. The right window displays the sign of the speed function.

3 System Design and Implementation

This section describes the interactive, level-set segmentation tool and the GPU implementation that makes it possible.

3.1 Interface and Usage

Our system consists of a graphical user interface (GUI) that presents the user with two volume slices and a control panel. The first slice window displays the current segmentation as a yellow line overlaid on top of the MRI data. The second slice viewing window displays a visualization of the speed function that clearly delineates the positive and negative regions. The GUI has controls for setting the three free speed parameters, a start/stop button to control the solver, and controls to save the 3D segmentation to file. The user can query grey-scale values in the MRI slice viewer and create spherical surface models. A screen shot of our interface is shown in Fig. 4.

3.2 GPU Level Set Solver Implementation

This section gives a brief description of the design of our GPU level-set solver. A more comprehensive description is available online at [19].

The efficient solution of the level-set PDEs relies on only updating voxels that are on or near the isosurface. The narrow band and sparse field methods achieve this by operating on sequences of heterogeneous operations. For instance, the sparse-field method [10] keeps a linked list of *active* voxels on which the computation is performed. Such algorithms are not well suited for streaming architectures and thus the mapping of the sparse field algorithm to GPUs requires a very different approach.

The sparse GPU level-set solver decomposes 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. 3b). 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.

There are several important details that make this strategy effective. First, because active 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. This is accomplished by adding an additional speed term to the PDE update equation, which forces 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,

$$G_r = \phi g_\phi - \phi|\nabla\phi|, \tag{3}$$

where ϕ is the value of the embedding at a voxel and $|\nabla\phi|$ is the gradient in the direction of the isosurface. The target gradient, g_ϕ, is set based the numerical precision of the level set data. This speed term is strictly a numerical construct; it does not affect the movement of the zero level set, i.e. the surface model, and its role in the segmentation process is computed without free parameters or user intervention.

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 figure 6 ran at 4 steps per second on the GPU and 0.25 steps per second on the CPU.

4 User Study

The purpose of this study was to determine if our algorithm can produce volumetric delineations of brain tumor boundaries comparable to those done by experts (e.g. radiologists or neurosurgeons) using traditional hand-contouring. We applied 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 chose nine cases for our study: three meningioma (cases 1-3) and 6 low grade glioma (4-6, 8-10). One case, number 7, was omitted because a quick inspection showed it that its intensity structure was too complicated to be segmented by the proposed tool—such a problem remains as future work, as we will discuss in Section 5. We performed *no preprocessing on the data*, and there are no hidden parameters in this study—all parameters in our system were set by the users in real time, as they interacted with the data and the models.

Five users were selected from among the staff and students in our group and trained briefly to use our software. We asked each user 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 recorded their time to complete each tumor. None of our users were experts in reading radiological data. It was not our intention to test for tumor recognition (tissue classification), but rather to test whether parameters could be selected for our algorithm to produce a segmentation which mimics those done by the experts. To control for tumor recognition, we allowed each user to refer to a single slice from an expert segmentation. Users were 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.

4.1 Metrics

We consider three factors in evaluating our segmentation method [20]: validity of the results (accuracy), reproducibility of the results (precision), and efficiency of the method (time). To quantify accuracy we established a ground truth from the expert segmented slices using the STAPLES method [21]. This method is essentially a sophisticated averaging scheme that accounts for systematic biases in the behavior of experts in order to generate a fuzzy ground truth (W) as well as sensitivity and specificity paramaters (p and q respectively) for each expert and each case. The ground truth segmentation values for each case are represented as an image of values between zero and one that indicates the probability of each pixel being in the tumor. Each subject generates a binary segmentation which, compared against the grount truth, gives values to obtain p and q for that subject. For our analysis we

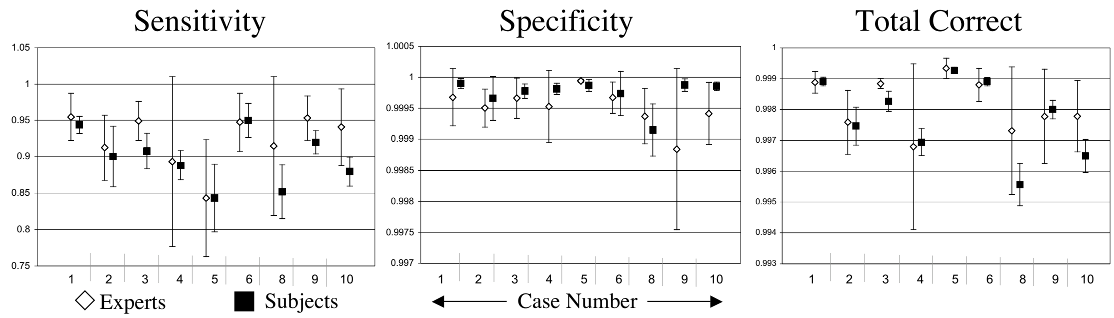


Figure 5: Results from the user study in compare with expert hand contouring reveal an overall comparable performance with a tendency to underestimate the region of tumor.

also considered a third metric, *total correct fraction* which is the total number of correctly classified pixels (weighted by W) as a percentage of the total size of the image.

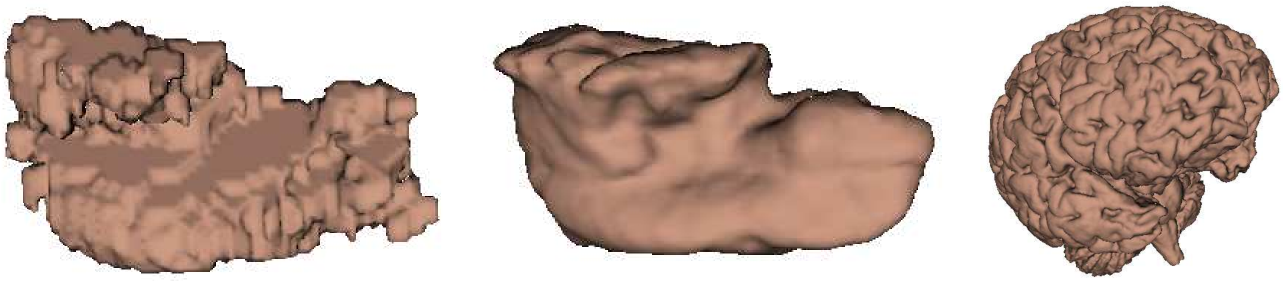
To assess interoperator precision in segmentations, we used the metric proposed by [20], which consists of pairwise comparisons of the cardinality of the intersection of the positive classifications divided by the cardinality of the union of positive classifications. To analyze efficiency, we calculated the average total time (user time plus processing time) taken for a segmentation.

4.2 Results

For a typical segmentation of a tumor using our tool a user scrolls through slices until they find the location of the tumor. With a mouse, the user queries intensity values in the tumor and sets initial values for the parameters T and ϵ based on those intensity values. They initialize a sphere near or within the tumor and initiate deformation of that spherical model. As the model deforms the user scrolls through slices, observing its behavior and modifying 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 will modify the model parameters between 10 and 30 times.

Figure 5 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 for the experts and the users in our study.

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. The average correct fraction of our users was better than the experts in 4 out of 9 cases. A general trend is that our users tended to underestimate the tumor relative to the experts, as indicated by lower values of p. 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.



(a) (b) (c)

Figure 6: (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. (c) A segmentation of the cerebral cortex from a 256 x 256 x 175 MRI volume using the same tool took 6 minutes.

The segmentations in our study show a much higher degree of precision than the expert hand segmentations. Mean precision [20] 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 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 $99.68\% \pm 0.29\%$, 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 quantative 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 6a–b show a segmentation by an expert with hand contouring compared with a segmentation done by one of our subjects. Screen-captured movies of a user interacting with our system are available online at [19].

5 Summary and Conclusions

A careful implementation of a sparse level-set solver on a GPU provides a new tool for interactive 3D segmentation. Users can manipulate several paramaters 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 other anatomy (e.g. Figure 6c).

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, more complicated greyscale profiles, and vector-valued data. Also, future work will include a true 3D interface, with cutting planes and real-time volume rendering of the models and surrounding tissue.

Acknowledgments

The authors would like to thank Gordon Kindlmann for his *nrrd* library (used for dataset manipulation and I/O), part of the *teem* toolkit available at `http://www.cs.utah.edu/~gk/teem`. Milan Ikits' *GLEW* sofware was also used extensively for OpenGL extension management. We also want to thank Simon Warfield, Michael Kaus, Ron Kikinis, Peter Black and Ferenc Jolesz for making the tumor database publicly available. This work was supported by grants #ACI0089915 and #CCR0092065 from the National Science Foundation.

References

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

[2] S. K. Warfield, M. Kaus, F. A. Jolesz, and R. Kikinis, "Adaptive, template moderated, spatially varying statistical classification," *Medical Image Analysis*, vol. 4, no. 1, pp. 43–45, 2000.

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

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

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

[6] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," in *Fifth International Conference on Computer Vision*, pp. 694–699, IEEE, IEEE Computer Society Press, 1995.

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

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

[9] D. Adalsteinson and J. A. Sethian, "A fast level set method for propogating interfaces," *Journal of Computational Physics*, pp. 269–277, 1995.

[10] R. T. Whitaker, "A level-set approach to 3D reconstruction from range data," *International Journal of Computer Vision*, vol. October, no. 3, pp. 203–231, 1998.

[11] J. D. Owens, *Computer Graphics on a Stream Architecture.* PhD thesis, Stanford University, Nov. 2002.

[12] M. Segal and K. Akeley, "The OpenGL graphics system: A specification (version 1.2.1)." http://www.opengl.org, 2003.

[13] Microsoft Corporation, "Direct3D." http://www.microsoft.com/directx, 2002.

[14] M. Rumpf and R. Strzodka, "Using graphics cards for quantized FEM computations," in *IASTED Visualization, Imaging and Image Processing Conference*, 2001.

[15] M. J. Harris, G. Coombe, T. Scheuermann, and A. Lastra, "Physically-based visual simulation on graphics hardware," in *Proc. SIGGRAPH/EG Graphics Hardware Workshop '02*, ACM, 2002.

[16] W. Li, X. Wei, , and A. Kaufman, "Implementing lattice boltzmann computation on graphics hardware," in *The Visual Computer*, (Heidelberg, Germany), Springer-Verlag, to appear 2003.

[17] M. Rumpf and R. Strzodka, "Level set segmentation in graphics hardware," in *International Conference on Image Processing*, pp. 1103–1106, 2001.

[18] A. Lefohn and R. Whitaker, "A gpu-based, three-dimensional level set solver with curvature flow." University of Utah tech report UUCS-02-017, December 2002.

[19] A. Lefohn, J. Cates, and R. Whitaker, "Interactive, gpu-based level sets for 3d brain tumor segmentation: Supplementary information." http://www.sci.utah.edu/ lefohn/work/rls/tumorSeg, 2003.

[20] J. Udupa, V. LeBlanc, H. Schmidt, C. Imielinska, P. Saha, G. Grevera, Y. Zhuge, L. Currie, P. Molholt, and Y. Jin, "A methodology for evaluating image segmentation algorithms," in *Proceedings of SPIE Vol. 4684*, pp. 266–277, SPIE, 2002.

[21] S. K. Warfield, K. H. Zou, and W. M. Wells, "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), pp. 298–306, Springer-Verlag, 2002.