Interactive, GPU-Based Level Sets for 3D Segmentation

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

Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT 84112
{lefohn,cates,whitaker}@sci.utah.edu

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. 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 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 segmentation, as supported by qualitative and quantitative results.

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 slow computation times combined with intensive parameter tuning. 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. 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.

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

R.E. Ellis and T.M. Peters (Eds.): MICCAI 2003, LNCS 2878, pp. 564–572, 2003.
© Springer-Verlag Berlin Heidelberg 2003

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 $\bar{v}(t)$ describes the point-wise velocity 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.

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 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 different 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 the curvature term 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 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.

Solving level-set PDEs on a volume requires proper numerical schemes [6] 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

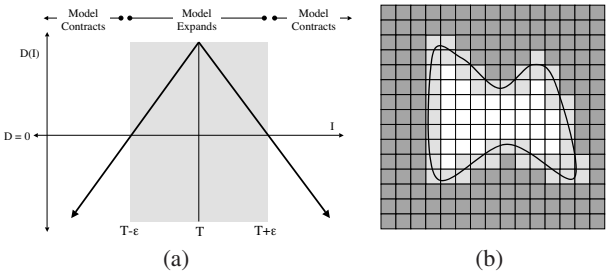

Fig. 1. (a) A speed function based on image intensity causes the model to expand over regions with greyscale values within the specified range and contract otherwise. (b) Efficient implementations of level sets entail computing the solution only near the moving wavefront.

are required to compute significant deformations. Efficient algorithms for solving the general level-set equations 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 [7,8] 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. However, even with a narrow band of computation, updates rates with these algorithms using conventional processors on typical medical data sets (e.g. 256^3 voxels) are not interactive.

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 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 paper presents a GPU computational model that supports dynamic, 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. Section 3 describes how our design addresses these challenges.

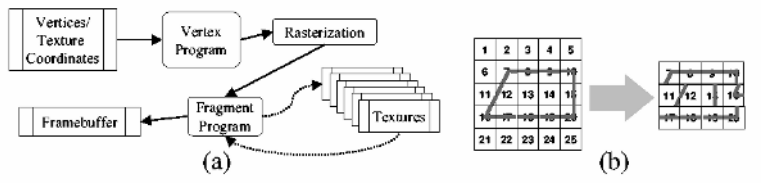

Fig. 2. (a) The modern graphics processor computation pipeline. (b) The proposed method relies on packing active tiles into 2D texture—a compressed format.

3 System Design and Implementation

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 greyscale values in the MRI slice viewer and create spherical surface models. A screen shot of our interface is shown in Fig. 3.

3.2 GPU Level Set Solver Implementation

This section gives a high-level description of our GPU level-set solver. A comprehensive description is available in Lefohn et al. [9].

The efficient solution of the level-set PDEs relies on updating only voxels that are on or near the isosurface. The narrow band and sparse field methods achieve this by operating on sequences of heterogeneous operations. Like the narrow band and sparse field CPU-based solvers, our sparse GPU level-set solver computes only those voxels near the isosurface. To run efficiently on GPUs, however, our solution must also have the following characteristics: efficiently updated texture-based data structures, no *scatter*

Fig. 3. 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.

write operations, and be highly data-parallel. We achieve these goals by decomposing the volume into a set of small 2D tiles (e.g. 16×16 pixels each). Only those tiles with non-zero derivatives are stored on the GPU (Fig. 2b). 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.

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. Using these texture coordinates, the GPU performs 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's automatic *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 (< 64 kB) is read back by the CPU and used to update the data structures that track the active volume regions. The CPU then sends the texture coordinates and vertices for the new set of active tiles to the GPU, and the next PDE iteration 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 5 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 [10]: 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 STAPLE method [11]. 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 ground truth, gives values to obtain p and q for that subject. For our analysis we 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 [10], 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 4 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.

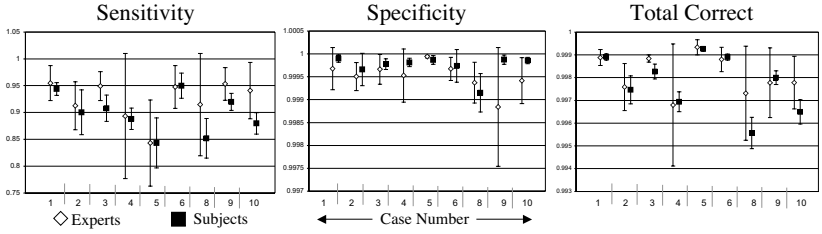

Fig. 4. Results from the user study in compare with expert hand contouring reveal an overall comparable performance with a tendency to underestimate the region of tumor.

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.

The segmentations in our study show a substantially higher degree of precision than the expert hand segmentations. Mean precision [10] 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 approxi-

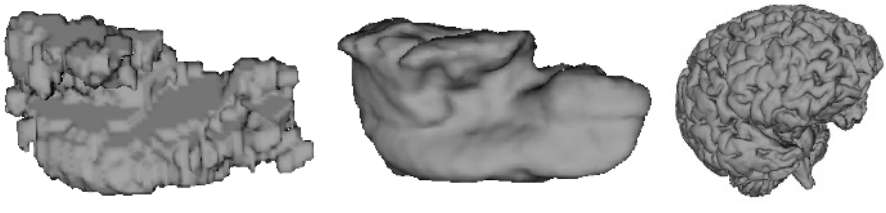

Fig. 5. (a) An expert hand segmentation of a tumor from the HBW database shows significant inter-slice artifacts. (b) A 3D segmentation of the same tumor from one of the subjects in our study. (c) A segmentation of the cerebral cortex from a 256 x 256 x 175 MRI volume using the same tool took 6 minutes.

mately 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.

The metrics requiring ground truth were computed on only a single 2D slice, which was extracted from the 3D segmentations, because of 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 inter-slice inconsistencies and fatigue. Figures 5a–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 [12].

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 5c).

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.

Acknowledgments. Thanks to the participants of our user study. We also thank Evan Hart, Mark Segal, Jeff Royal and Jason Mitchell at ATI for donating technical advice and hardware to this project. Simon Warfield, Michael Kaus, Ron Kikinis, Peter Black and Ferenc Jolesz provided the tumor database. This work was supported by NSF grants ACI0089915 and CCR0092065 and NIH/NLM N01LM03503.

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., Kaus, M., Jolesz, F.A., Kikinis, R.: Adaptive, template moderated, spatially varying statistical classification. Medical Image Analysis **4** (2000) 43–45
3. 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)
4. 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
5. 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
6. Osher, S., Sethian, J.: Fronts propogating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. Journal of Computational Physics **79** (1988) 12–49

7. Adalsteinson, D., Sethian, J.A.: A fast level set method for propogating interfaces. Journal of Computational Physics (1995) 269–277
8. Whitaker, R.T.: A level-set approach to 3D reconstruction from range data. International Journal of Computer Vision **October** (1998) 203–231
9. Lefohn, A., Kniss, J., Hansen, C., Whitaker, R.: Interactive deformation and visualization of level set surfaces using graphics hardware. In: IEEE Visualization. (2003) To Appear
10. Udupa, J., LeBlanc, V., Schmidt, H., Imielinska, C., Saha, P., Grevera, G., Zhuge, Y., Currie, L., Molholt, P., Jin, Y.: A methodology for evaluating image segmentation algorithms. In: Proceedings of SPIE Vol. 4684, SPIE (2002) 266–277
11. Warfield, S.K., Zou, K.H., Wells, W.M.: 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
12. Lefohn, A., Cates, J., Whitaker, R.: Interactive, GPU-based level sets for 3D brain tumor segmentation: Supplementary information. `http://www.sci.utah.edu/lefohn/work/rls/tumorSeg` (2003)