A Parallel Cellular Automata with Label Priors for Interactive Brain Tumor Segmentation

Edward Kim, Tian Shen, Xiaolei Huang
Lehigh University
Department of Computer Science and Engineering, Bethlehem, PA, USA
{edk208,tis207,xih206}@lehigh.edu

Abstract

We present a novel method for 3D brain tumor volume segmentation based on a parallel cellular automata framework. Our method incorporates prior label knowledge gathered from user seed information to influence the cellular automata decision rules. Our proposed method is able to segment brain tumor volumes quickly and accurately using any number of label classifications. Exploiting the inherent parallelism of our algorithm, we adopt this method to the Graphics Processing Unit (GPU). Additionally, we introduce the concept of individual label strength maps to visualize the improvements of our method. As we demonstrate in our quantitative and qualitative results, the key benefits of our system are accuracy, robustness to complex structures, and speed. We compute segmentations nearly 45x faster than conventional CPU methods, enabling user feedback at interactive rates.

1 Introduction

Reliable image segmentation is an important yet challenging task in medical image analysis. Fully automatic methods of segmentation can produce decent results; however, an entirely autonomous method can vary significantly in accuracy. Even the ground truth segmentation in medical imaging may differ between physicians. Therefore, interactive [2, 9, 5, 11, 7, 6] segmentation methods have increased in popularity. The Graph Cut [2] method attempts to solve the min cut/max flow problem. Snakes [6] and Level Sets [9] are active contour methods that evolve a curve based upon geometric and image constraints. For the problem of brain tumor segmentation, Lefohn *et al.* [9] implemented a level set solver on the GPU. Quantitative results of this level set formulation compare well with hand contouring results. Kaus *et al.* [8] used an atlas and statistical information to segment brain tumors. Cates *et al.* [3] used a hierarchical watershed segmentation method where users can select and combine catchment basin regions to produce the final tumor model. Additionally, methods that allow the use of any number of label classifications include Random walks [5], Growcut [11], and Seeded ND Cellular Automata [7]. Random walks has interactive properties but it scales in computation time with multiple labels. Growcut uses Cellular Automata (CA) [12] with intensity based transition rules for image segmentation. The Seeded ND-CA method by Kauffmann *et al.* uses CA to compute the Ford-Bellman shortest path for segmentation on an older generation GPU. Although the Growcut method and Seeded ND-CA method seem dissimilar, Kauffmann *et al.* proves the equivalence between Growcut to his method and states that their results are identical.

No single segmentation method is suitable for all kinds of images [4], and further, there is no general algorithm accepted to segment tumors from magnetic resonance images [1]. We were drawn to the CA method for several reasons. First, the CA method supports n-dimensions and m-label categories where the number of labels *does not* increase computational time or complexity. This ability makes it suitable for simultaneously segmenting multiple brain lesions with complex structures. Second, for an effective interactive segmentation, speed and usability are crucial. Since CA is an iterative method where each cell independently follows a set of rules, this naturally lends itself to an efficient parallel implementation. Further, the iterative nature of this method enables the user to interact with the method and visualize the results during the segmentation process. Previously proposed CA methods, e.g. Growcut [11] and Seeded ND-CA [7], only use local voxel information to guide their evolution, making them prone to errors where boundaries exist.

To overcome this problem, we present an interactive method that integrates label priors learned from the user interactions in a parallelized CA. Our method allows users to easily segment and update the segmentation of any number of different structures in the brain. We focus on brain tumor segmentation and make the following contributions,

1. Statistical analysis of user input seeds' intensities to create a global model of the intensity distribution for each classification label. The states of the CA are influenced by this model via the newly defined cell evolution rules.
2. A GPU implementation that parallelizes the automata computation, which achieves a ∼45x speed-up compared to conventional CPU methods and enables the visualization and manipulation of 3D MRI data at interactive rates.

2 Cellular Automata (CA)

The CA model consists of a grid of cells (or voxels in our case) that influence a set of neighboring cells according to transition or evolution rules. In our segmentation application, some cells will be assigned labels (user seeds) and these cells will attempt to propagate their labels to neighboring cells. At label boundaries, neighbors will compete with each other based on the transition rules. Mathematically speaking, we can define our cellular automata as a triple, $\mathcal{CA} = (S, N, \delta)$ where S is a nonempty set of *states*, $N \subseteq \mathbb{R}^3$ is the *neighborhood*, and $\delta : S^N \rightarrow S$ is the *local transition rule*. The area of computation is our volume, $v \in V \subseteq \mathbb{R}^3$, where v represents a voxel in our volume. We define our neighborhood system as the von Neumann neighborhood (6-voxel neighborhood),

$$N(v) = ||v - q||_1 := \sum_{i=1}^{3} |v_i - q_i| = 1 \qquad q \in \mathbb{R}^3 \quad (1)$$

where q are the adjacent voxels. The cell states for voxel v, S_v, are defined as a 4-tuple, $(l_v, \theta_v, I_v, \varphi_v)$, where l_v is the label of the current cell, θ_v is the strength of the cell and $\theta_v \in [0, 1]$, I_v is the voxel intensity, φ_v is the number of enemies (voxels that have differing labels) surrounding the cell.

For the initialization of our method, the user paints seeds onto a 2D slice for any number of label classifications (i.e. foreground and background for the 2-label case) which assigns the corresponding label to that voxel and also initalizes the strength of that voxel to 1.0. All uninitialized voxels have strength 0.0. After initialization, the label propagation iterations begin. Our basic transition rule for cell v's states S_v, is based on the following monotonically decreasing function,

$$g(|I_v - I_q|) \cdot \theta_q > \theta_v \quad (2)$$

Where the g function is defined in the basic case as,

$$g(x) = 1 - \left(\frac{x}{max|I|}\right) \quad (3)$$

Where $max|I|$ is the maximum intensity in the volume. Intuitively, the function g modulates how heavily the neighbor, q, influences v. The more similar q's intensity is to v's, the heavier q's influence is to v. If neighbor q overpowers v, i.e. ($g(|I_v - I_q|) \cdot \theta_q > \theta_v$ is true), then v takes q's label and update its strength, ($l_v = l_q$; $\theta_v = g(|I_v - I_q|) \cdot \theta_q$). However, if this transition rule is not satisfied, the cell state remains unaltered. This operation is performed for each neighbor in the 6-voxel neighborhood, and the maximum strength label is taken as the final label classification. As long as the monotonically decreasing property of equation 3 is maintained, the CA will converge to a steady state.

3 Methodology

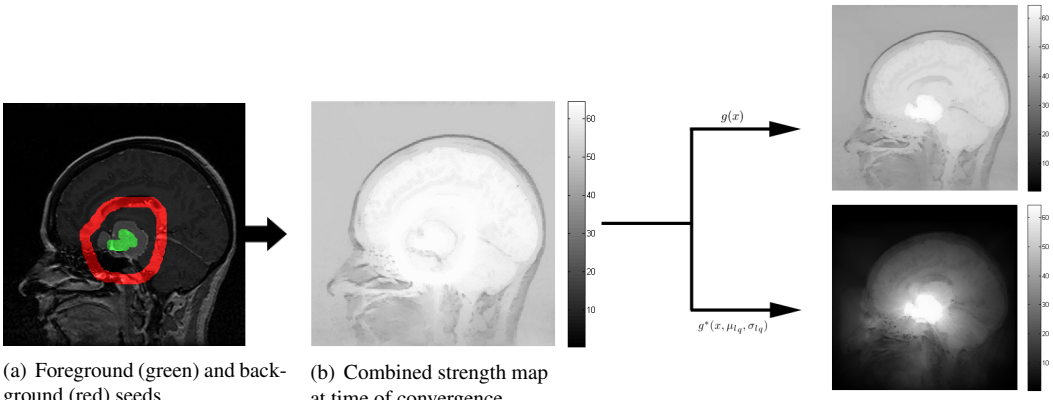
The basic CA transition rules using the g function defined in eq. (3) are limited to local voxel information only, making the basic CA's results prone to errors where boundaries exist and neighbors with different labels compete. In this section, we first introduce a new modulation function leveraging user seed interactions to reduce the local bias and better delineate boundaries, and then show the effectiveness of the new method through a visualization of the strength term, θ_v, in our state vector, S_v.

3.1 Statistical User Seed Distribution

In our analysis of the CA algorithm, we note that the strength term, θ_v, in the cell states is the central deciding factor for the label determination of our segmentation. Therefore, we can achieve a better segmentation result by introducing prior knowledge to better control this term. We use seed labels initialized at the start of the procedure to estimate a Gaussian probability density function of intensities representing each label set. We calculate the mean, μ_l, and standard deviation, σ_l, of the intensities of the user initialized seeds in each of the different labels, and use them as prior knowledge. The transition rule in eq. (2) is followed, as demonstrated in our Algorithm 1, but with a new piecewise continuous, monotonous decreasing function that is exponentially influenced by seed density distributions, $g^*(x, \mu_{l_q}, \sigma_{l_q})$, defined below,

$$g^*(x, \mu_{l_q}, \sigma_{l_q}) = \begin{cases} 1 - \left(\frac{x}{w_1 \cdot max|I|}\right) & \text{if } |I_q - \mu_{l_q}| < \sigma_{l_q} \\ \left(1 - \left(\frac{x}{w_1 \cdot max|I|}\right)\right)\left(e^{\left(-w_2 \cdot (|I_q - \mu_{l_q}| - \sigma_{l_q})\right)}\right) & \text{if } |I_q - \mu_{l_q}| \geq \sigma_{l_q} \end{cases} \quad (4)$$

Where the notation, μ_{l_q} and σ_{l_q}, denote the mean and standard deviation of the label at voxel q respectively. Further, w_1 and w_2 are user defined weights. Setting $w_1 > 1$ decreases the effect of intensity differences; whereas, setting $w_2 > 1$ exponentially increases the effect of intensity differences. Here, if $|I_q - \mu_{l_q}| < \sigma_{l_q}$, i.e. the neighboring voxel attacking the current voxel has an intensity value that lies within the standard deviation of its label's intensity, it



(a) Foreground (green) and background (red) seeds.

(b) Combined strength map at time of convergence.

(c) Individual strength maps for $g(x)$ (top) and $g^*(x, \mu_{l_q}, \sigma_{l_q})$ (bottom) for the foreground label.

Figure 1. Case 2 from the Harvard Medical School at the Brigham and Women's Hospital (HBW) [8, 13] brain tumor dataset and strength maps. In (c), the $g^*(x, \mu_{l_q}, \sigma_{l_q})$ method better maintains the CA strength within the tumor tissue region and exponentially drops the strength outside its statistical range as computed by the label seeds. In this example, we set our weights, w_1 and w_2 to 2.0.

will follow a weighted linear penalty function. However, if $|I_q - \mu_{l_q}| \geq \sigma_{l_q}$, i.e. the difference between the q's intensity, I_q, and mean intensity of the q's label, μ_{l_q}, lies outside the statistical range, σ_{l_q}, our method will drop the strength of a neighboring attack exponentially. This penalty term is shifted by σ_{l_q} in order to maintain continuity at $(|I_q - \mu_{l_q}| = \sigma_{l_q})$.

3.2 Cellular Automata Individual Label Strength Maps

The reasoning behind our method can be derived from the visualization of the strength term, θ_v. The visualization of this term is defined as our strength map. In the original formulation, the maximum strength of each voxel is stored in θ_v which is associated with a specific label, l_v. However, if we store only the maximum strength information, the resulting strength map is a convoluted mix of different labels and is visually indecipherable. This is because there is no delineation between the strengths of different labels, see Fig. 1 (b). Conversely, by storing and analyzing the individual label's strength map, we can qualitatively motivate our work. For the specific example shown in Fig. 1 (c), the individual strength map is generated for the foreground only, and describes the strength of the foreground label across the entire image. By visual inspection, it can be seen that the original function, $g(x)$, unknowingly propagates the strength of the foreground label into different tissue classes. In contrast, the proposed method, $g^*(x, \mu_{l_q}, \sigma_{l_q})$, maintains the CA strength within its own label, but then drops exponentially in the other tissue classes. We present quantitative results in section 4.

3.3 CUDA Implementation

CUDA, Compute Unified Device Architecture, is a programming model and software environment for GPU programming. In CUDA, the basic computational elements are *threads*. A collection of threads is called a *block* and the combination of all the blocks makes up the *grid*. Typically, the grid covers the entire area of computation for a GPU program (in our case, the MRI volume). A single GPU program is called a *kernel* and it is the *kernel* that exploits the power of the GPU.

In our implementation, we first create a 3D texture volume that contains the MRI slices. We define our *grid* as the dimensions of the texture volume and launch the evolution *kernel*, Algorithm 1. This algorithm incorporates equation, eq. (4), and assigns new values to our cell states from iteration t to $t + 1$. This method continues until the states converge on the entire volume (label and strength); however, typically the final satisfactory segmentation is achieved many iterations before convergence and can be stopped by the user.

To achieve additional speed up, we propose a narrow band modification to improve our overall computation time. We append a new state value to our existing 4-tuple cell states to create a 5-tuple, $(l_v, \theta_v, I_v, \varphi_v, f_v)$ where f_v is the narrow band propagation value, $\{true, false\}$. This state variable is set to *true* if an enemy exists in the neighborhood, $N(v)$ or if the voxel's strength has changed from time, t to $t + 1$. If none of these conditions are true, we set the value to *false* and the GPU may skip the neighborhood computations. Additionally, by counting the number of enemies in a neighborhood, we can also control smooth-

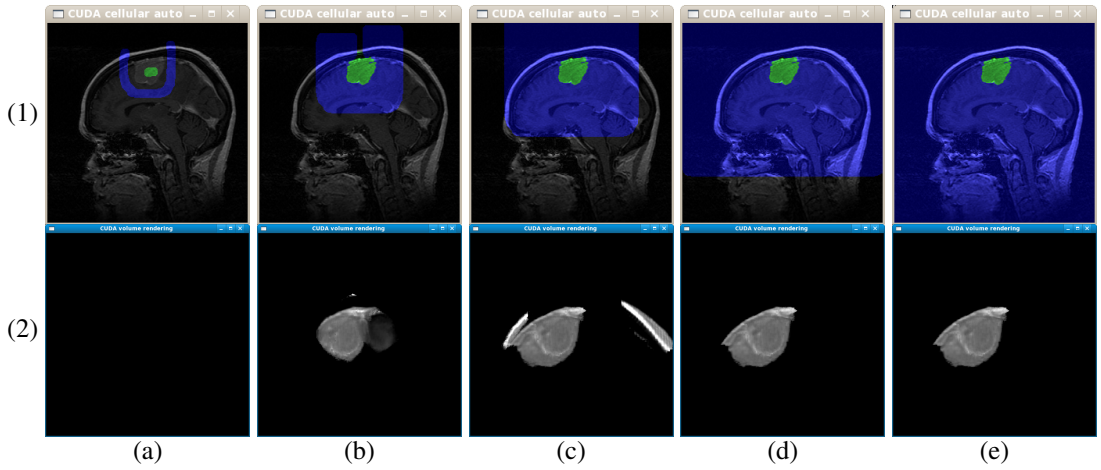


Figure 2. Segmentation process of Case 1 in the HBW database. (1) 2D stack rendering of one of the slices of the volume. (2) Volume rendering window of the result on the foreground label highlighted in green. Results presented at iteration 0 (a), iteration 25 (b), iteration 50 (c), iteration 100 (d), and finally at convergence, iteration 176 (e).

ness as described in [11].

For our user interface, we present two windows: a 2D image of the slices within the MRI volume, and a 3D volume render of the GPU texture. For the 2D image slices, the user can scroll through the volume stack slice by slice, and draw seed points for any label classification. After placing the seed points, the user can execute any number of iterations of Algorithm 1 and watch the CA method progress. For the 3D volume render window, the results of the CA method are visualized in real time. The user interface and process can be seen in Fig. 2.

Algorithm 1 Evolution Kernel for 5-tuple Cell States, $(l_v, \theta_v, I_v, \varphi_v, f_v)$

Step 1. Check if this voxel is in the narrow band as described in Sec. 3.3. If it is, proceed to Step 2; otherwise, skip this voxel.
Step 2. Fetch entire neighborhood, using the GPU texture, tex3D(x, y, z)
for $\forall q \in N(v)$ **do** {for each neighboring voxel}
  if $l_v^t \neq l_q^t$ **then** {labels do not match}
    $\varphi_v^{t+1} = \varphi_v^{t+1} + 1$; {increase the enemy count}
  end if
  if $g^*(|I_v - I_q|, \mu_{l_q}, \sigma_{l_q}) \cdot \theta_q^t > \theta_v^t$ **and** $g^*(|I_v - I_q|, \mu_{l_q}, \sigma_{l_q}) \cdot \theta_q^t > \theta_v^{t+1}$ **then**
    $l_v^{t+1} = l_q^t$ {overpowered; take your neighbors label}
    $\theta_v^{t+1} = g^*(|I_v - I_q|, \mu_{l_q}, \sigma_{l_q}) \cdot \theta_q^t$ {update maximum strength}
  end if
end for

4 Results

We evaluate our method using several metrics on the Brain Tumor Segmentation Database made available by the Harvard Medical School at the Brigham and Women's Hospital (HBW) [8, 13]. This dataset consists of ten 3D MRI brain tumor patients specifically selected by neurosurgeons as a representative sample of a larger database. For our results, we considered speed (both computational and operator) and validity.

4.1 Validity

As noted by Grady *et al.* [5] with seed based segmentation methods, strict measurements of error are practically meaningless because we can arbitrarily increase our segmentation accuracy by adding more seeds. Thus, in our experiments, we compare our results to other methods [2, 5, 11, 7] that have identical seed initializations. Here in Table 1, we present the Dice overlap (DSC) [10] for one 2D slice in each of the ten tumor cases, where each method was given identical initializations with no further interaction allowed. The seeds used for the experiment can be seen in Fig. 3, but in the case of multiple labels e.g. case 9, only one tumor foreground/background segmentation is calculated to accommodate the 2-label methods. Typically, we see that all methods perform well when the tumor boundary is easily delineated from normal brain tissue. But when the boundary is less defined, as in cases 4, 8, and 10, our method is able to leverage the label intensity distributions to achieve a better segmentation. For our method, we heuristically found that setting our weights, w_1 and w_2 to 2.0 worked well across all cases.

4.2 Computational Speed

As a tool for interactive segmentation, it is essential that our method be able to produce results on 3D volumes efficiently. Because each cell in our CA acts independently, the algorithm can be efficiently parallelized. For our computational experiments, we compared our CUDA GPU imple-

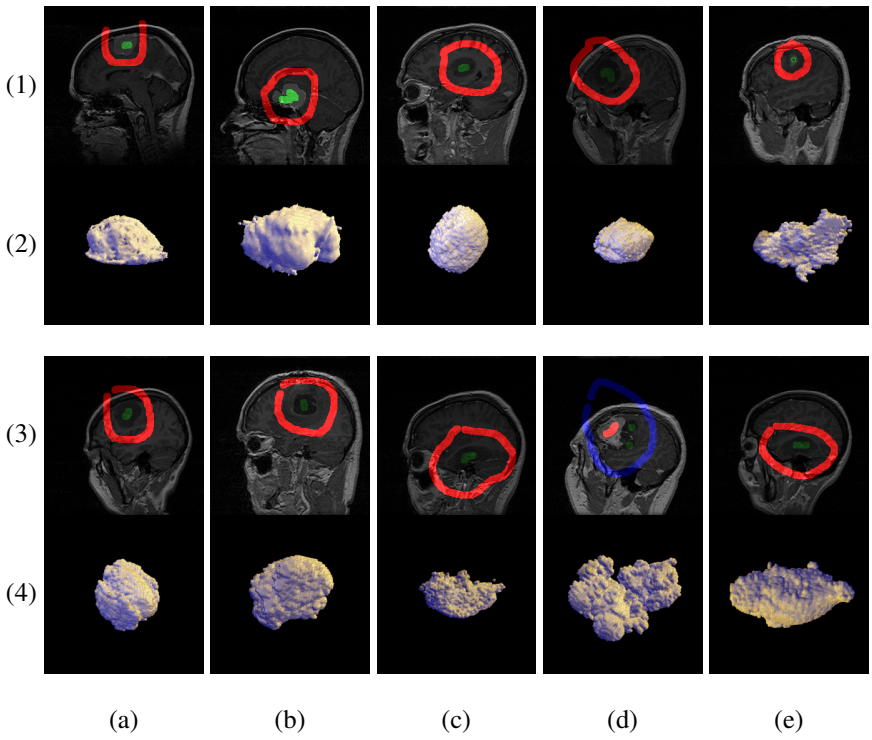


Figure 3. Seed initializations displayed in rows (1)&(3), and renderings of results, rows (2)&(4). Columns (a)-(e) in rows (1)&(2) are Cases 1-5 and (a)-(e) in rows (3)&(4) are Cases 6-10. The renderings of our results were obtained from one of the users in our user study, and have been rotated and resized for optimal viewing.

Table 1. Validity comparisons on a 2D image slice from each of the ten volumes. Seed inputs were identical for all methods and no user interactions were allowed.

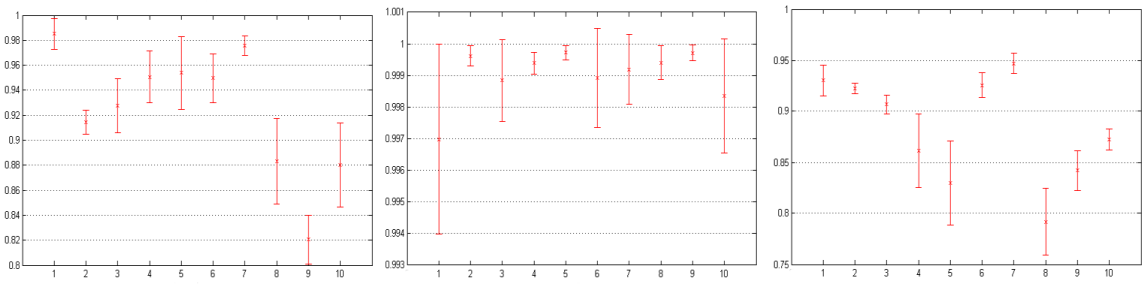
	DSC overlap for Cases 1- 10									
Method	1	2	3	4	5	6	7	8	9	10
Graph Cut [2]	0.939	0.950	0.945	0.749	0.805	0.948	0.957	0.781	0.873	0.740
Random Walker [5]	0.930	0.949	0.951	0.672	0.887	0.930	0.958	0.725	0.873	0.707
Growcut [11] or Seeded ND-CA [7]	0.933	0.951	0.958	0.720	0.881	0.953	0.952	0.694	0.801	0.835
Our method	**0.932**	**0.950**	**0.956**	**0.905**	**0.869**	**0.959**	**0.969**	**0.821**	**0.915**	**0.879**

mentation running on an NVIDIA 280 GTX versus a CPU implementation on an AMD 4600+ Dual Core processor. The size of the tumor volumes are 256x256x124 and the average number of iterations it takes for our method to converge on a tumor segmentation is 191.7. If we run the segmentation algorithm on the CPU, it would average 393.22 seconds to compute. In comparison, our GPU implementation takes only 8.68 seconds on average, resulting in a 45.27x speed up.

4.3 User Study on Validity and Usability

For our user study, we asked 5 participants to test our software tool. Three of the users were experienced in reading radiological data and the other two had no previous experience. These users were trained on the user interface of the software tool and were then asked to segment tumors from each of the ten cases available from the HBW database. The participants were shown where the tumor was in the volumes and could use our system to segment the tumor until they were satisfied with their result. We then recorded the time it took for these users to segment out each tumor, the final sensitivity, P, specificity, Q, and DSC of the entire volume, see Fig. 4 (calculations for P,Q, and DSC can be found in [10]). Our P and Q results are consistently on par or above those presented in Lefohn *et al.* [9] and Cates *et al.* [3]. And although in case 9 multiple tissue lesions exist, users are able to segment labels simultaneously without added complexity.

The average operator time of a single volume segmentation on our system is 3m$\pm$2m. This is shorter than that



(a) sensitivity (b) specificity (c) DSC

Figure 4. The sensitivity, P, specificity, Q, and DSC from our user study in case numbers 1 through 10 from the HBW dataset. The graphs plot the mean and standard deviation for the entire tumor segmentation volume gathered from five users.

of those presented in Kaus *et al.* [8] (5-10 minutes), Cates *et al.* [3] (5-10 minutes), and Lefohn *et al.* [9] (6±3 minutes). We hypothesized a marginal speed up due to newer hardware; however, we also achieve greater speed efficiency from our GPU implementation and little or no time waiting for processing, effectively decreasing our total operator time across all users.

5 Conclusion

We proposed an interactive segmentation method that enables users to quickly and efficiently segment tumors in MR brain volumes. Our method utilizes statistical seed distributions to overcome the local bias seen in the traditional cellular automata framework. Our results show improved accuracy, robustness, and competitive usability. Further, with a GPU implementation, the method produces results at interactive rates. For our future work, we plan to work with a greater number of brain structures and explore incorporating additional information to guide our CA. We would also like to explore higher dimensional data and improve our user interface.

6 Acknowledgments

The authors would like to thank our user study participants and also Simon Warfield, Michael Kaus, Ron Kikinis, Peter Black and Ferenc Jolesz for making the tumor database publicly available. This work was partially supported by the Grants US National Institutes of Health (NIH) NLM-HHSN276200900722P and US National Science Foundation (NSF) IIS-0812120.

References

[1] N. Archip, F. Jolesz, and S. Warfield. A Validation Framework for Brain Tumor Segmentation. *Academic radiology*, 14(10):1242–1251, 2007.

[2] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1124–1137, 2004.

[3] J. Cates, R. Whitaker, and G. Jones. Case study: an evaluation of user-assisted hierarchical watershed segmentation. *Medical Image Analysis*, 9(6):566–578, 2005.

[4] J. Duncan and N. Ayache. Medical image analysis: Progress over two decades and the challenges ahead. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 85–106, 2000.

[5] L. Grady, T. Schiwietz, and S. Aharon. Random Walks for Interactive Organ Segmentation in Two and Three Dimensions: Implementation and Validation. *MICCAI*, pages 773–780, 2005.

[6] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[7] C. Kauffmann and N. Piché. Seeded ND medical image segmentation by cellular automaton on GPU. *International Journal of Computer Assisted Radiology and Surgery*, pages 1–12, 2009.

[8] M. Kaus, S. K. Warfield, A. Nabavi, P. M. Black, F. A. Jolesz, and R. Kikinis. Automated Segmentation of MRI of Brain Tumors. *Radiology*, 218(2)(586-91), 2001 Feb.

[9] A. Lefohn, J. Cates, and R. Whitaker. Interactive, gpu-based level sets for 3d segmentation. *MICCAI*, pages 564–572, 2003.

[10] A. Popovic, D. La, M. Engelhardt, and K. Radermacher. Statistical validation metric for accuracy assessment in medical image segmentation. *International Journal of Computer Assisted Radiology and Surgery*, 2:169–181, 2007.

[11] V. Vezhnevets and V. Konouchine. Grow-Cut - Interactive Multi-Label N-D Image Segmentation. *Graphicon*, 2005.

[12] J. Von Neumann. Theory of self-reproducing automata. *University of Illinois Press. Ed. and Completed by A. Burks*, 1966.

[13] S. K. Warfield, M. Kaus, F. A. Jolesz, and R. Kikinis. Adaptive, Template Moderated, Spatially Varying Statistical Classification. *Medical Image Analysis*, 4(1)(43-55), 2000 Mar.