An Effective Interactive Medical Image Segmentation Method Using Fast GrowCut

Linagjia Zhu[1], Ivan Kolesov[1], Yi Gao[2], Ron Kikinis[3], and Allen Tannenbaum[1]

[1] Stony Brook University {liangjia.zhu, ivan.kolesov, allen.tannenbaum}@stonybrook.edu
[2] University of Alabama at Birmingham gaoyi@uab.edu
[3] Harvard Medical School kikinis@bwh.harvard.edu

Abstract. Segmentation of anatomical structures in medical imagery is a key step in a variety of clinical applications. Designing a generic, automated method that works for various structures and imaging modalities is a daunting task. In this paper, we present an effective interactive segmentation method that reformulates the GrowCut algorithm as a clustering problem and computes a fast, approximate solution. The method is further improved by using an efficient updating scheme requiring only local computations when new user input becomes available, making it applicable to high resolution images. The algorithm may easily be included as a user-oriented software module in any number of available medical imaging/image processing platforms such as 3D Slicer. The efficiency and effectiveness of the algorithm are demonstrated through tests on several challenging data sets where it is also compared to standard GrowCut.

1 Introduction

Segmentation has long been one of the most important tasks in medical image analysis; see [1] and the references therein. Though numerous algorithms have been proposed and published, the segmentation of general anatomical structures across a variety of modalities remains a challenging task. Automatic segmentation is attractive, but user intervention is inevitable in real applications. The type of interaction differs; for instance, a user may make a series of clicks around target boundaries [2] or draw sample regions [3–5] to guide the segmentation. Processing speed and intuitiveness of the interaction are two critical properties that must be considered in designing a useful interactive segmentation system. The GrowCut algorithm is a widely used interactive tool for segmentation because of several key features: 1) natural handling of N-D images, 2) support of multi-label segmentation, 3) on-the-fly incorporation of user input, and 4) easy implementation. Starting from manually drawn seed and background pixels, GrowCut propagates these input labels based on principles derived from cellular automata in order to classify all of the pixels. Since all pixels must be visited during each iteration, the computational complexity grows quickly, as image size increases. This implementation detracts from its applicability to the

segmentation of 3D images. The focus of this paper is to design an interactive segmentation system that maintains the positive features of GrowCut while improving its efficiency, making it applicable to medical volumes. GrowCut is compared to other interactive methods in [4, 6].

In this paper, we reformulate GrowCut as a clustering problem based on finding a shortest path that can be solved efficiently with the Dijkstra algorithm [7]. However, the reformulation results in a static problem, changing the "dynamic" property of GrowCut. That is, editing is not allowed while running the Dijkstra algorithm. As an alternative, an efficient method we call the *adaptive Dijkstra algorithm* is introduced to incorporate user inputs; it updates only those local regions affected by the new input. This formulation and its implementation are described in Section 2. Experimental results in Section 3 demonstrate the efficiency and effectiveness of the proposed method. Conclusions follow in Section 4.

2 Fast GrowCut

In this section, we outline the GrowCut algorithm [4], and present the fast version. Accordingly, let $P \subset R^N, N \geq 2$ denote the image domain. For $p \in P$, denote by $N(p)$ the Moore neighborhood of p, i.e., the 8 directly neighboring pixels of p in 2D and 26 voxels of p in 3D. The influence of a point $q \in N(p)$ on p is defined as $g(||C_q - C_p||_2) \in [0, 1]$, a monotonically decreasing function, where C_p and C_q are feature vectors at points p and q, respectively. For scalar images, C_p is simply the intensity at point p. The *strength* $\theta_p \in [0, 1]$ and *label* $l_p \in \{BG, FG, \cdots\}$ are associated with a point p to track the state of the GrowCut process.

Initially, only seed pixels are labeled; the strength θ is set to 1 for these pixels and to 0 otherwise. The GrowCut algoritm, summarized in Algorithm 1, terminates after a fixed number of iterations K or when no pixels change state. Convergence is guaranteed when K is sufficiently large because the updating process is monotonic and bounded [6].

Algorithm 1 GrowCut

1: Initialize l_p and θ_p for $\forall p \in P$
2: **while** not reach termination conditions **do**
3: **for** $\forall p \in P$ **do**
4: $l_p^{t+1} = l_p^t$; $\theta_p^{t+1} = \theta_p^t$ ▷ copy previous state
5: $q^* = \text{argmax}_{q \in N(p)}\{g(||C_q - C_p||_2) \cdot \theta_q^t\}$
6: $l_p^{t+1} = l_{q^*}^t$, $\theta_p^{t+1} = g(||C_{q^*} - C_p||_2) \cdot \theta_{q^*}^t$ ▷ update state at p

2.1 GrowCut as Clustering

From Algorithm 1, it is seen that $\forall p \in P$, l_p in the steady state is determined by l_{s^*}:

$$s^* = \underset{s \in S}{\operatorname{argmax}} \max_{q_i \in H(s,p)} \{\theta(s) g(||C_{q_1} - C_s||_2) g(||C_{q_2} - C_{q_1}||_2) \cdots g(||C_p - C_{q_n}||_2)\}, \tag{1}$$

where S is the set of seed pixels, and $H(s,p)$ is any path connecting pixel p and seed s. This observation may be reached via proof by contradiction as was done in [6].

It is known that the GrowCut algorithm converges slowly, especially when applied to 3D medical images because it traces through the entire image domain at each iteration [8]. The speed of GrowCut can be significantly increased if the updating process is organized in a more structured manner. This goal can be achieved by approximating the product defined Eq. (1) as discussed below.

Suppose $x_i \in [0,1]$, $g(0) = 1$, and $g'(0) = -\alpha \leq 0$. Define $Q(\cdot)$ as:

$$Q(x_1, \cdots, x_n) = \prod_{i=1}^{n} g(x_i). \tag{2}$$

Applying the Taylor expansion and Weierstrass product inequality [9] to Eq. (2), we see that:

$$\begin{aligned} Q(x_1, \cdots, x_n) = \prod_{i=1}^{n} \left(1 - \alpha x_i + O(x_i^2)\right) &\geq 1 - \sum_{i=1}^{n} \Big(\alpha x_i - O(x_i)\Big) \\ &= 1 - \alpha \sum_{i=1}^{n} x_i + O\big((\sum_{i=1}^{n} x_i)^2\big). \end{aligned} \tag{3}$$

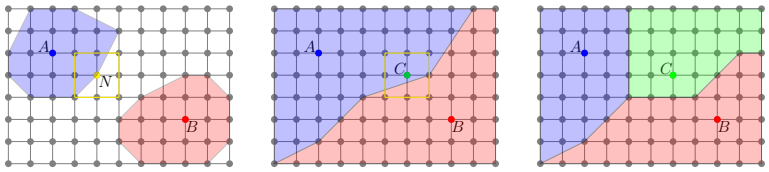
Thus, $1 - \alpha \sum_{i=1}^{n} x_i$ gives an approximation to the lower bound of $Q(x_1, \cdots, x_n)$. Normalizing $||C_p - C_q||_2$ to lie in $[0,1]$ and applying the result of Eq. (3) to Eq. (1), an approximate solution to the GrowCut algorithm is given as:

$$\tilde{s}^* = \underset{s \in S}{\operatorname{argmin}} \min_{q_i \in H(s,p)} \{||C_{q_1} - C_s||_2 + ||C_{q_2} - C_{q_1}||_2 + \cdots + ||C_p - C_{q_n}||_2\}. \tag{4}$$

Note that the inner part of Eq. (4) can be considered as the shortest weighted distance from p to s, and the outer part as the clustering based on this distance. One advantage of this formulation is that Eq. (4) can be solved very efficiently with the Dijkstra algorithm [7]. The fastest implementation employs a Fibonacci heap [10] and has a time complexity of $O(|E| + |V| \log |V|) = O(|V|(\log |V| + |N|))$ where E and V are the edges and vertices in the graph, respectively. The computational complexity of GrowCut is $O(|V||N|K) \gg O(|V|(\log |V| + |N|))$, where K is the number of iterations. As a special case, when $g(\cdot)$ is an exponential function, maximizing Eq. (2) is equivalent to minimizing its argument [6].

2.2 Adaptive Dijkstra Algorithm

If we employ the standard Dijkstra algorithm to solve Eq. (4), we lose the "dynamic" property of GrowCut, which is crucial for a fully interactive system. Thus, we apply the Dijkstra algorithm *adaptively* that allows real-time interaction. The algorithm is summarized in Algorithm 2 and illustrated in Fig. 1.



(a) Front propagation (b) New seed C (c) Local update

Fig. 1. The *adaptive Dijkstra algorithm*. (a)Expansion of the front by updating the neighbors (yellow square) of each point N on the front. (b) Addition of a new seed C to the final segmentation from seeds A and B (blue and red). (c) Pixels updated (green area) after adding seed C.

Initially, when starting a new segmentation, labels of the seed image are assigned to the current label image $labCrt$. It is assumed that the target labels are positive integers. The current distance map $distCrt$ is generated by setting its values to 0 at label points and ∞ at all other points. $distCrt$ is used to initialize the Fibonacci heap $fibHeap$. Then, the algorithm always chooses the point with the smallest distance in the heap to update its neighbors' distance and label information (lines 19–23), illustrated by Fig. 1(a). Once a point is removed from the heap, its state is fixed. New neighbors are updated while they are in the heap. This process continues until the heap is empty and the final segmentation is reached; see Fig. 1(b). Finally, $distCrt$ and $labCrt$ are stored (line 28) as $distPre$ and $labPre$, respectively, to be used in the next round of segmentation.

When updating the current segmentation, the new seeds added during the latest user editing stage are compared to $labPre$, and only those points whose labels are changed are selected as new seeds to update the current segmentation (lines 6–9). The distance $distCrt$ from these new seeds is propagated if its value is smaller than $distPre$, which may reassign labels based on the shortest-path clustering rule. Otherwise, the algorithm either keeps the previous state or terminates (lines 14–18). Finally, the distance and label information are locally updated (lines 26–27), see Fig. 1(c), which significantly reduces the refinement time.

2.3 Implementation

The proposed method was implemented based in 3DSlicer [11], which provides a framework for interactive operations and visualization. Specifically, the al-

Algorithm 2 Adaptive Dijkstra

function ADIJKSTRA(img, $seeds$, $labCrt$, $distCrt$, $labPre$, $distPre$, $newSeg$)
  if $newSeg$ **then**
    $labCrt = label(seeds)$;
    $distCrt(p) = 0$ if $p \in S; distCrt(p) = \infty$ otherwise
  else ▷ initialize from current segmentation
    if $seeds(p) \neq 0$ and $label(seeds(p)) \neq labPre(p)$ **then**
      $labCrt(p) = label(seeds(p)); distCrt(p) = 0$
    else
      $labCrt(p) = 0; distCrt(p) = \infty$
  Initialize $fibHeap$
  while $fibHeap$ is not empty **do** ▷ segmentation/refinement loop
    get p^* with the smallest distance from $fibHeap$
    if not $newSeg$ **then**
      if $distCrt(p^*) == \infty$ **then**
        break;
      else if $distCrt(p^*) > distPre(p^*)$ **then**
        $distCrt(p^*) = distPre(p^*)$; $labCrt(p^*) = labPre(p^*)$
        continue; ▷ update locally
    for $\forall q \in N(p^*)$ **do** ▷ regular Dijkstra
      $dist = distCrt(p^*) + ||C_q - C_{p^*}||_2$
      if $dist < distCrt(q)$ **then**
        $distCrt(q) = dist$; $labCrt(q) = labCrt(p^*)$
        update $fibHeap$ at q
  if not $newSeg$ **then**
    for $\forall p, distCrt(p) < \infty$ **do** ▷ get updated points
      $labPre(p) = labCrt(p); distPre(p) = distCrt(p)$ ▷ update local states
      $labCrt = labPre; distCrt = distPre$
  $labPre = labCrt; distPre = distCrt$ ▷ save current results

gorithm is part of Editor toolbox, allowing all editing tools in 3DSlicer to be utilized. The whole implementation is delivered as a part of an open source package available at [12].

3 Experimental Results

We compared the proposed method with a publicly available GrowCut implementation [8]. In this test, the lungs were segmented from an MR image. The test was performed on a computer with Quad CPU 2.6GHz and 8G RAM. The ROI size was $502 \times 336 \times 43$ pixels, shown as rectangular boxes in Fig. 2. To illustrate the efficiency of the proposed method, three rounds of editing were used to segment the lungs. After each round of editing, the adaptive Dijkstra algorithm was performed. As shown in Fig. 2, the update only happens locally, around new user inputs. The Dice coefficient and volume overlap ratio are used to show how accurately the proposed method approximates the GrowCut method. Fig. 3 shows the final segmentation result. A detailed comparison is presented

in Table 1. According to these results, the proposed method is markedly faster ($\sim 8.4\times$ and $128\times$ faster than the GrowCut module in 3DSlicer [8] for initial segmentation and refinement, respectively), while maintaining a high approximation accuracy (97% in both measures). The differences (3% in both measures) may arise from: 1) approximation in Eq. (1); 2) implementation of GrowCut in 3DSlicer uses a fixed number of iterations, which may be exhausted before convergence. The proposed method consumed more memory (storing $labPre$ and $distPre$) to achieve the significant improvement in computation time.

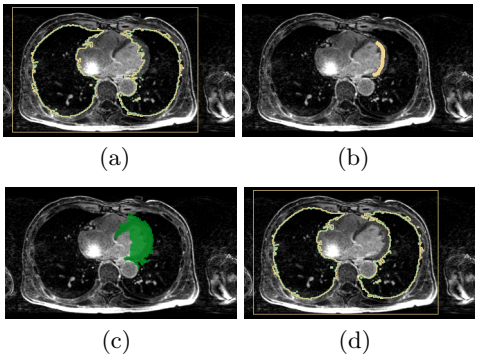


Fig. 2. Effect of user interaction on lungs segmentation: (a) current segmentation, (b) new seeds, (c) region updated by adaptive Dijkstra, and (d) new segmentation.

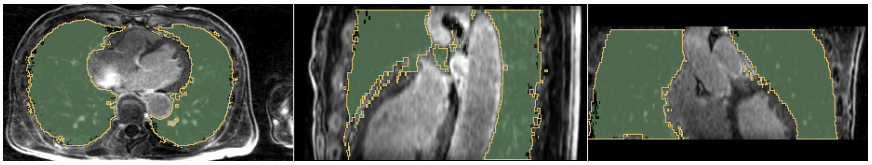


Fig. 3. Segmentation results from GrowCut [8] (green region) and the proposed method (yellow contour) in axial, coronal, and sagittal views, respectively.

Below, two examples are used to illustrate the proposed method for multi-label segmentation. The first is a commonly used example for brain tumor and ventricle segmentation; see Fig. 4. In this example, user inputs are mainly in the axial and coronal views, where background seeds roughly enclose the two target labels. The segmentation was performed within an automatically determined ROI. In the second example, the blood pool is segmented from a cardiac CT image (Fig. 5). In this test, four chambers are labeled separately; user labels were mainly drawn in the axial and coronal views. Seeds were also marked around the valve between the right ventricle and the atrium in the sagittal view in

Table 1. Original GrowCut vs. the proposed implementation for lungs segmentation.

Method	Time (seconds)			Memory (MB)	Similarity	
	1st round	2nd round	3rd round		Dice	Vol. Overlap
GrowCut[8]	210	255	269	200	97%	97%
Proposed	25	2	2	522		

order to get a better delineation in places where no clear boundary is defined (see Fig. 5(a)). As seen in these results, only a few user inputs are required to compute the segmentations.

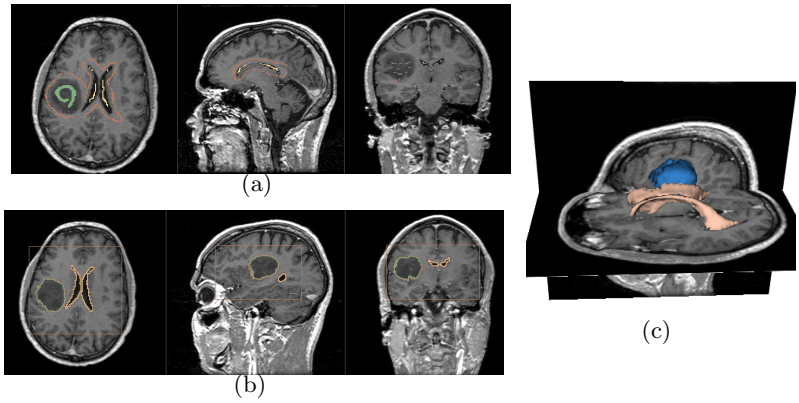


Fig. 4. Brain tumor and ventricle segmentation. (a) Seed image and (b) segmentations in axial, coronal, and sagittal views, respectively. (c) Segmented targets in 3D view.

4 Conclusions

In this paper, we presented an effective and efficient 3D medical image segmentation method. It provides a fast approximation to the GrowCut approach by reformulating the original problem of maximization of a product as the minimization of a summation, which allows for a very efficient implementation using Dijkstra. Results demonstrate the accuracy and speed improvement of the proposed method. In some future work, we plan to investigate more rigorously how close the proposed method approximates the original GrowCut. Further, we want to introduce a smoothing procedure into the interactive approach in order to refine the multi-label segmentation. Regarding the latter point, we are exploring the incorporation of the methodology proposed in [5] in which level sets are interactively utilized.

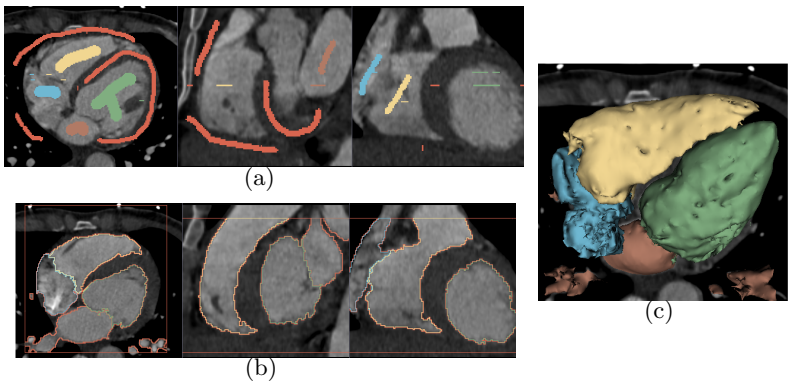


Fig. 5. Heart blood pool segmentation. (a) Seed image and (b) segmentations in axial, coronal, and sagittal views, respectively. (c) Segmented targets in 3D view.

Acknowledgements

This research was supported by the National Center for Research Resources under Grant P41-RR-013218,the National Institute of Biomedical Imaging and Bioengineering under Grant P41-EB-015902 of the National Institutes of Health through the Neuroanalysis Center of Brigham and Women's Hospital, and the National Alliance for Medical Image Computing, funded by the National Institutes of Health through the NIH Roadmap for Medical Research, under Grant U54 EB005149.

References

1. Suri, J.: Computer vision, pattern recognition and image processing in left ventricle segmentation: The last 50 years. Pattern Analysis and Applications **3**(3) (2000) 209–242
2. Mortensen, E.N., Barrett, W.A.: Interactive segmentation with intelligent scissors. Graphical Models and Image Processing **60**(5) (September 1998) 349–384
3. Boykov, Y., Jolly, M.P.: Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In: International Conference on Computer Vision. (2001) 105–112
4. Vezhnevets, V., Konouchine, V.: "growcut" – interactive multi-label N-D image segmentation by cellular automata. In: Proc. Graphicon. (2005) 150–156
5. Karasev, P., Kolesov, I., Fritscher, K.D., Vela, P., Mitchell, P., Tannenbaum, A.: Interactive medical image segmentation using PDE control of active contours. IEEE Transactions on Medical Imaging **32**(11) (2013) 2127–2139
6. Hamamci, A., Kucuk, N., Karaman, K., Engin, K., Unal, G.: Tumor-Cut: Segmentation of brain tumors on contrast enhanced MR images for radiosurgery applications. IEEE Transactions on Medical Imaging **31**(3) (2012) 790–804
7. Dijkstra, E.W.: A note on two problems in connexion with graphs. NUMERISCHE MATHEMATIK **1**(1) (1959) 269–271

8. Veeraraghavan, H., Miller, J.: 3D Slicer. https://www.slicer.org/slicerWiki/index.php/Modules:GrowCutSegmentation-Documentation-3.6
9. Honsberger, R. More Mathematical Morsels (1991) 244–245
10. Fredman, M., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM (JACM) **34**(3) (July 1987) 596–615
11. 3D Slicer. http://www.slicer.org/
12. Fast GrowCut. http://wiki.slicer.org/slicerWiki/index.php/Documentation/4.3/Modules/FastGrowCut/