Segmentation of Volume Data Based on 3D Lazy Snapping

Chia-Yang Sun* and Cheng-Wei Ku†
National Center for High-Performance Computing, Hsinchu 30076 Taiwan R.O.C.
Tel: + 886-3-5776085 #248 Fax: +886-3-577-3620
*E-mail: morrissun@nchc.org.tw
†E-mail: c00cwk00@nchc.org.tw

***Abstract*— The image segmentation techniques for medical image play important roles in computer-aided detection, analysis and measurement. Due to the computed tomography (CT) and the magnetic resonance imaging (MRI) technologies, the doctor can obtain volume images to visualize the structures of internal organ of human body. In this paper, we present a volume segmentation method by extending the Lazy Snapping [1] to 3D. A high-level painting tool is provided for the user to indicate which part is object and which is background in volume by marking some lines. The segmentation result can be determined with little interaction without voxel-precise selection. In order to improve the performance, we adopt the watershed used in [1] to pre-process the volume data. And we implement the 3D watershed to handle the volume data. However, large size of volume data still needs great computation. It is hard to achieve a real time response. To improve the efficiency without losing the accuracy, we introduce a volume of interest (VOI) specified by the user to restrict the data size for processing. Our system also implements a user-friendly interface to browse the volume of interest area in three cutting planes.**

I. Introduction

Image segmentation is the technique to partition the image into groups. The segmentation result can help us to find the groups which have some characteristics we are interested, and such groups are called the foreground or objects of interest. When the objects in images are located, they can be processed and analyzed without considering other information in the background. Currently imaging acquisition devices has generated higher resolution of volume data. This advance has encouraged medical researchers and biologists to bring their data in 3D instead of looking through a 2D microscopy. To locate the objects in the 3D data or so called volume data, it is tedious and inefficient to perform 2D segmentation on each slice. Therefore, different 3D segmentation methods [2] for volume data have been proposed to handle such problems.

Nowadays, the medical imaging such as computed tomography (CT) and the magnetic resonance imaging (MRI) and other image modalities provide a non-invasively mapping of the anatomy for a subject. Traditionally, physicians can make diagnosis or learn potential life saving information by observing huge amount slices. But the volume visualization technique allows scientists and physicians to virtually interact with the structure of anatomy. In the volume rendering paradigm, we can use the transfer function to assign a specified scalar value to a color and opacity. Because the complex structure of organs and tissues, it is hard to visualize the specific object by just making certain tissues transparent. To solve this issue, 3D segmentation can be conducted to partition the structures of anatomy and render the separated objects individually. Applying 3D segmentation techniques in medical imaging can improve visualization, detection of disease, analysis and measurement.

Lazy Snapping is developed based on graph cut [3] and focuses on “image cutout” for a 2D image. Although it is a semi-automatic segmentation method, an intuitive user interface is designed to allow non pixel-level selection for the user. It provides instant visual feedback, making cutout boundary close to the object contour despite the presence of ambiguous or low contrast edges. The Lazy Snapping is utilized to assist in the segmentation of 3D polygon meshes [4] and separated 3D models from the crime scene environments obtained with a stereo sensor [5].

In this paper, we propose a 3D segmentation algorithm by extending the Lazy Snapping to 3D. For 3D Lazy Snapping, we have to convert the data to a graph and then handle the segmentation problem by solving the graph cut. In order to reduce the complexity of graph and improve the efficiency, we construct the graph by the small regions in the data pre-processing result instead of single voxels. We implement 3D watershed which can over-segment the data to small regions in the pre-processing phase. However, large size of volume data still needs great computation. To improve the efficiency without losing the accuracy, we introduce a volume of interest (VOI) specified by the user to restrict the data size for processing. In the graph cut phase, the segmentation problem maps to a graph cut problem and the result is a binary segmentation. The segmentation result are "object" and "background" dependent on the choice of seeds assigned by the user. We also implement an interface which allows the user to browse, locate VOI and segment the object of interest in three cutting planes.

The remainder of the paper is organized as following: in section II we will implement and evaluate two 3D watershed algorithms. Section III focuses on the 3D Lazy Snapping algorithm. Section IV provides the experiment results and conclusions are presented in Section V.

II. 3D WATERSHED

Watershed is an image segmentation algorithm to separate the image into small regions and boundaries of regions locate on edges of the image well. Although direct application of the watershed algorithm often leads to over-segmentation, a large number of segmented regions can be used to describe the image instead of pixels. In the past few years, two conceptually distinct techniques have been developed to calculate the watershed. One of them is immersion simulation and the other is raining simulation. In subsection A and B, we will discuss the 2D immersion method and the 2D raining method first and then expand them to 3D respectively. The experiment results of 3D immersion and 3D raining simulation are shown in subsection C.

A. Immersion Simulation

For 2D immersion simulation [6], the image is considered as a geographic surface with the intensity of pixel representing the elevation. Suppose that a hole is punched in each local minimum and the topography is immersed from below by letting water rise through the holes. As the water continues to rise, it will over flow from one catchment basin to another. A dam is built to prevent water coming from different minimum from merging. Until the water rising to the highest mountain, all potential dams have been built. The final dams correspond to the watershed lines which are the desired result and locate well on boundaries of the image. The segmentation result consists of many small regions and watershed lines.

For 3D immersion simulation, it is not intuitive to treat 3D data as a geographic surface. However, we still can perform immersion simulation by letting water rise from each local minimal. We have to check 26 neighbors of each pixel to see if there is water coming from another minimum and going to merge instead of checking only 8 neighbors in 2D. The segmentation result consists of many 3D regions which are formed with voxels and watershed lines. Before the immersion, 2D immersion algorithm is used to sort the pixels of input image in the incremental order of their intensity values and stores the pixels in different arrays according to its intensity value. To improve the efficiency, these pixels can be accessed directly through the sorted array without scanning the whole image to find them over and over again. Although the sorted array works well in most of 2D image, sorting voxels of 3D data to sorted arrays takes lots of memory.

B. Raining Simulation

For 2D watershed based on raining simulation, it is assumed that raindrops fall on the geographic surface of the image. Each raindrop must flow down to the valley from its falling point along the steepest descent path. Sun et al. [7] introduced the connected component to describe the route which the raindrop passed between the falling point and the valley. All the connected components lead to the same valley forming a catchment basin. This algorithm can label all catchment basins by scanning the whole image four times. It has better efficiency than immersion simulation. In 2007, Osma-Ruiz et al. [8] proposed an efficient algorithm to compute the shortest paths of raindrops. And all catchment basins can be labeled using only two scan of whole image. This algorithm has been proved to perform better than [7].

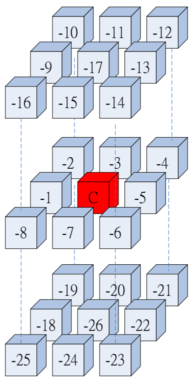


Fig. 1 The 26 possible directions that the raindrop in voxel C may fall. Label voxel C with the number of falling direction.

We implement a 3D watershed algorithm based on raining simulation proposed by [8]. For this method, we consider the 26 directions that raindrops may fall in 3D space. The criterion to assign directions is showed in Fig. 1. In addition, the constant value -27 indicates that the point has not been analyzed yet, and -28 means that it is in the process of being labeled.

C. 3D Watershed Experiment Results

We implement two kinds of 3D watershed algorithm that mentioned before. Our goal is to find a suitable method to compute the 3D over-segmentation. The CT data from 5 cases are processed by 3D watershed algorithms. In addition, the mean color of each region in the segmentation result is computed. Table I shows the experiment results. It shows that the raining algorithm has lower memory usage than immersion algorithm and has better efficiency in most cases. We use the 3D raining simulation watershed in our 3D lazy snapping system as a data pre-processor.

TABLE I
3D WATERSHED EXPERIMENT RESULTS
• All measurements are performed on a PC with Core 2 Extreme 3.0GHz CPU and 8GB memory.

Data resolution	Immersion simulation		Raining simulation	
	Time (sec.)	Max memory usage (MB)	Time (sec.)	Max memory usage (MB)
512×512×32	8.187	188.125	6.266	87.574
512×512×102	21.204	562.812	21.562	361.730
512×512×132	39.610	751.035	34.000	348.824
512×512×139	30.407	764.304	29.797	495.523
512×512×162	42.344	981.859	44.797	540.171

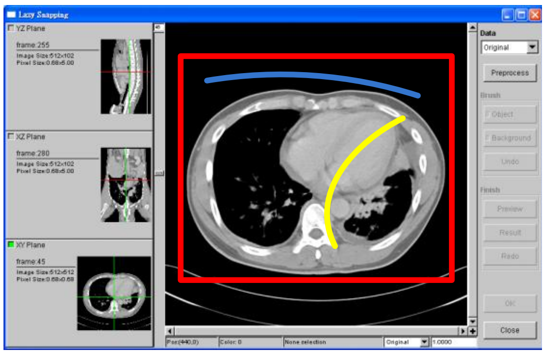


Fig. 2 3D Lazy Snapping UI. The red rectangle is the selection of VOI. The yellow line is drawn to indicate the object, and the blue line to indicate the background.

III. 3D Lazy Snapping Algorithm

In this section, we will describe the detail about the user-interface of our system, the graph construction using the result from 3D watershed segmentation, the weights of edges and min-cut result.

A. UI Design

We implement a viewer to display volume data and it allows the user to browse the data in one of cutting planes, for anatomy i.e. coronal plane, sagittal plane and axial plane. In order to improve efficiency, we then perform the pre-processing by using 3D watershed segmentation on VOI determined by the user instead of whole data. The user can select a spatial region as a VOI that encompasses the object of interest, as the red rectangle shown in Fig. 2. The width and length of VOI corresponded to the dimensions of rectangle while height of VOI is the slices number in another dimension perpendicular to the rectangle. If the VOI didn't be assigned before pre-processing, the whole data is equal to VOI.

After pre-processing, the user can indicate the 3D object to be segmented simply by marking sample object and background elements in the scene. The high level painting-type UI is provided to mark object and background by drawing lines, and it does not require very precise user input. The user can switch the slices in three cutting planes to find the object and mark corresponding seeds on object and background respectively. The segmentation process is triggered once when the user push the "Preview" button. The user can then either augment the set of object or background seeds, allowing the system to re-compute the segmentation.

B. Graph Construction

Input VOI data is expressed as the graph $g = \langle \nu, \varepsilon \rangle$, where ν is the set of nodes and ε is the set of all edges connecting pairs of adjacent nodes. The nodes are regions in the over-segmentation result and the edges are adjacency relationships between neighboring regions. Each region represents a node p and there is an edge connecting to each node that corresponds to an adjacent region. A source terminal node S and a sink terminal node T are created that are not belonging to any regions. For each non terminal node, two edges were created from this node to source terminal node and sink terminal node respectively. Fig. 3(a) illustrates this graph.

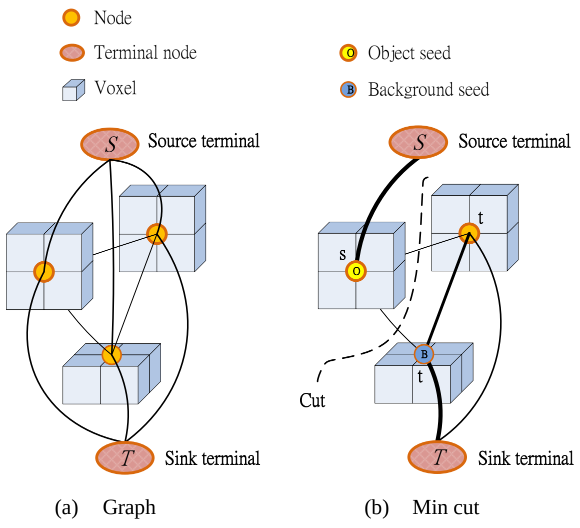


Fig. 3 A 3D Segmentation example for a 3 nodes image and each node has 4 voxels. The weight of each edge is reflected by the edge's thickness.

The volume segmentation problem can be converted to a binary labeling problem. The goal of segmentation is to assign a unique label x_i for each node $i \in \nu$, i.e. $x_i \in (0\,(\text{background}), 1\,(\text{object}))$. The labeling problem can be solved by using graph cut on g .

C. Weights of Edges

An edge connecting a pair of neighboring nodes *p* and *q* will be denoted by {*p*,*q*}. The weight of edge represents the similarity of these connecting nodes. The higher weight number indicates greater similarity of the two connected nodes. The edge weight is defined in table II. A penalty term $E(x_i, x_j)$ is assigned due to the gradient along the segmentation boundary. Because of $|x_i - x_j|$, only the adjacent nodes along the segmentation boundary have to be considered with this term. The more similar the colors of the two nodes are, the larger the weight is, and thus the segmentation boundary is less likely on the object boundary.

TABLE II
WEIGHTS OF EDGES FOR GRAPH.
Where O and B denote the subsets of nodes marked as "object" and "background" seeds. N denotes a neighboring system.
$d_i^F = \min_n \|C(i) - K_n^F\|$ and $d_i^B = \min_m \|C(i) - K_m^B\|$,where $\{K_n^F\}$ and $\{K_m^B\}$ are mean colors of the foreground and background clusters.
$E(x_i, x_j) = |x_i - x_j| \frac{1}{\|C(i) - C(j)\|^2 + 1}$, where $C(i)$ represents the color value of node i. [1]

EDGE	WEIGHT	FOR
$\{p, S\}$	∞	$p \in O$
	0	$p \in B$
	$\frac{d_i^F}{d_i^F + d_i^B}$	$p \in v, p \notin O \cup B$
$\{p, T\}$	0	$p \in O$
	∞	$p \in B$
	$\frac{d_i^B}{d_i^F + d_i^B}$	$p \in v, p \notin O \cup B$
$\{p, q\}$	$E(x_i, x_j)$	$\{p, q\} \in N$

D. Min-cut/Max-flow

To get the graph cut, we use the min-cut/max-flow algorithm which ref to [9]. A min cut is a partition of node set which has two subsets such that the terminal nodes become separated. For volume segmentation problem, the min cut is the segmentation boundary while all nodes connecting to source terminal node are object nodes, and all nodes connecting to sink terminal node are background nodes. Fig. 3(b) shows the min cut result.

IV. RESULTS

In this section, we will discuss two successful cases processed by 3D Lazy Snapping. The first case, we used 3D Lazy Snapping to assist the diagnosis and the detection for lung cancer. In the other case, 3D Lazy Snapping was served as a training tool and it created a 3D mouse model for educational purpose by visualizing mouse organs from the segmentation result.

A. Lung Cancer Segmentation

There are two challenges for lung cancer segmentation. First, the lung cancer cells stick on the normal cells. Second, the boundaries between lung cancer cells and normal cells are not obvious.

We recognized the lung cancer cells by the diagnosis of experienced doctors. And then our system was used to assist the user in segmenting the lung cancer cells. We loaded thorax CT scan data as input to our system and located the VOI on the lung cancer cells, as shown in Fig. 4. The CT data are DICOM format with resolution 512×512 and 112 slices total. Due to the VOI selection, we can get a quick response and save the memory usage. We marked object and background seeds by drawing lines on slices 45 and 48, as shown in Fig. 5. The segmentation result in slice 45~48 can be seen in Fig. 6. We augmented object and background seeds to refine the segmentation result until all the lung cancer cells had been segmented, as shown in Fig. 7.

Fig. 8 shows more examples produced by 3D Lazy Snapping.

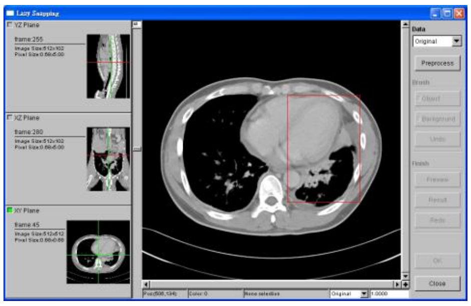
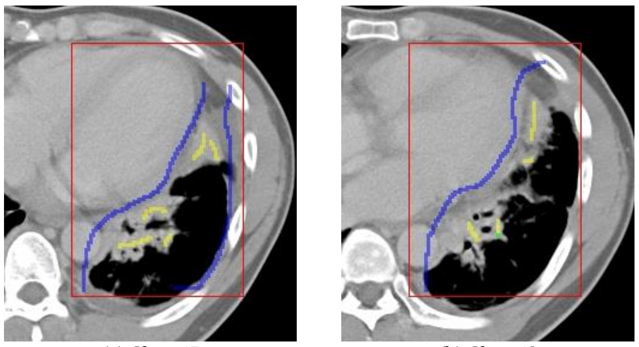
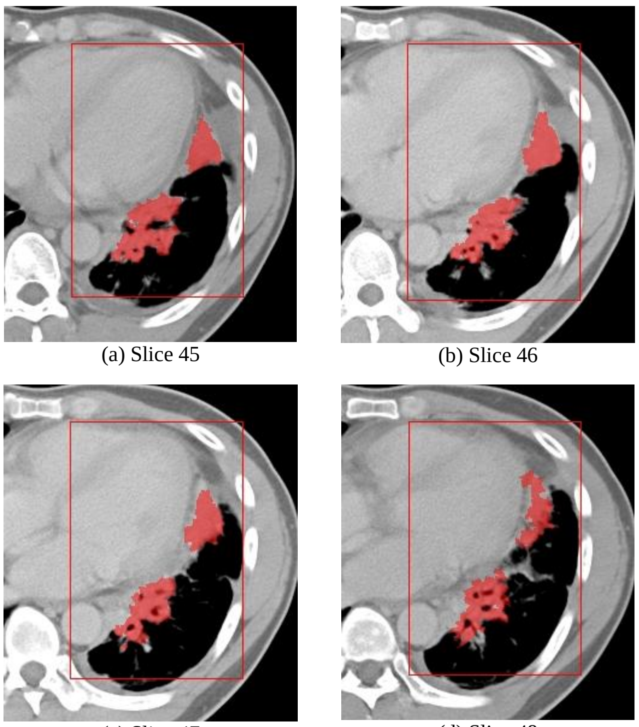


Fig. 4 3D Lazy Snapping UI with CT data loaded.



(a) Slice 45 (b) Slice 48

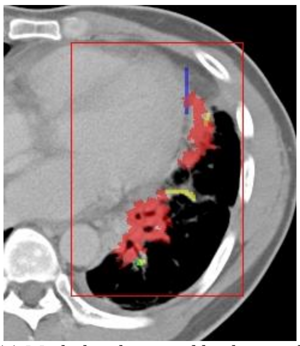
Fig. 5 Mark the object and background seeds in different slices.



(a) Slice 45 (b) Slice 46

(c) Slice 47 (d) Slice 48

Fig. 6 Lung cancer segmentation result in slice 45~48.


(a) Mark the object and background seeds in slice 48

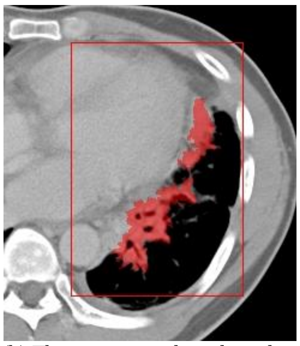
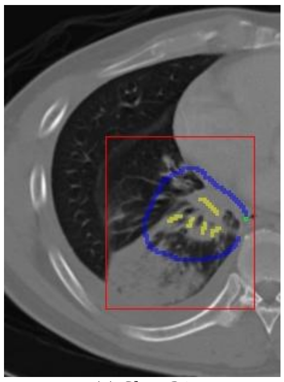
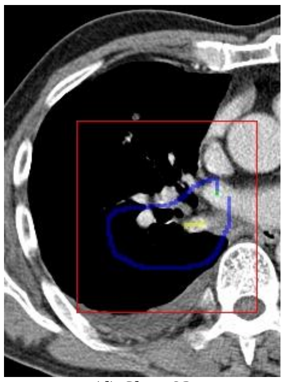
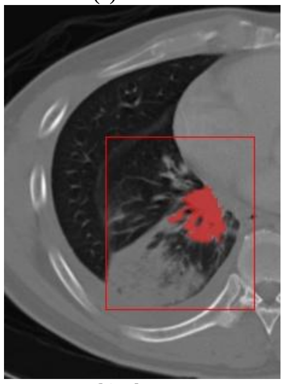
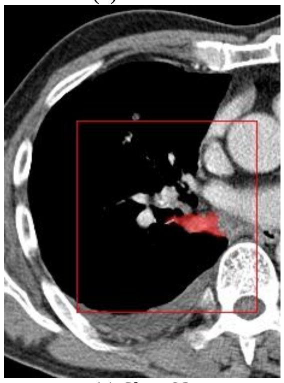

(b) The re-computed result in slice 48

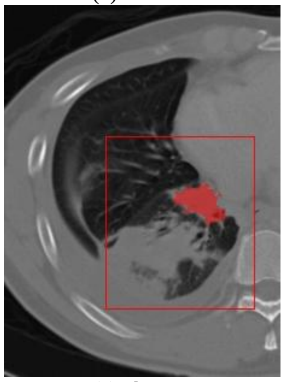
Fig. 7 Augment the object and background seeds to re-compute the result.


(a) Slice 31


(d) Slice 83


(b) Slice 31


(e) Slice 83


(c) Slice 28

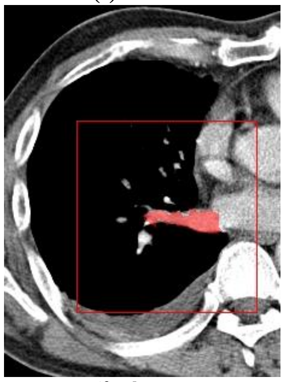

(f) Slice 80

Fig. 8 More experiments: The left column shows the lung cancer segmentation experiment (a,b,c) of one case, and the right column shows the other case(d,e,f). The top row shows the VOI and the marking lines. The second row and the third row show the segmentation results in different slices of these two cases.

B. *Mouse Organs Segmentation*

The laboratory mouse is mostly provided to researchers for study of its DNA and disease diagnosis. In order to reduce the sacrifice of laboratory mouse, the 3D mouse model has been constructed as a training tool by visualizing the organs of mouse. Our system was used to segment the mouse organs and then each organ could be visualized individually. To get volume data, the mouse was frozen to the ice and it was finely sectioned by cryostat-microtome at 10 um for microscopy. Each microscopic image can clearly reflect the structure and true color of all mouse organs. After we required a set of image, we registered the image set first to align every image position and then resized it. The data size is 750×400 with X and Y dimension and 501 slices with Z dimension. We demonstrated the kidney segmentation by locating VOI on one of the kidney and marking the object and background seeds on XY plane, as shown in Fig. 9, 10. The kidney segmentation result is shown in Fig. 11. Another example we demonstrated was the central nervous system segmentation. In order to get a better view of whole central nervous system, we switched the cutting plane to XZ plane and located the VOI, as shown in Fig. 12. Our system allowed the segmentation to be performed in any cutting plane. The central nervous system segmentation result is shown in Fig. 13.

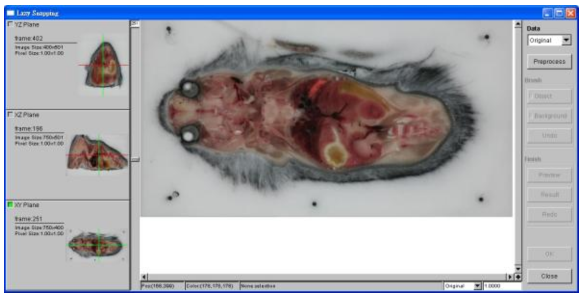
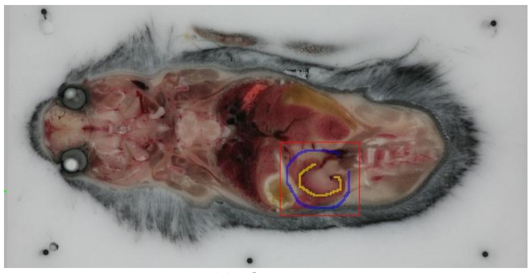


Fig. 9 3D Lazy Snapping UI with the mouse data loaded.


(a) Slice 253

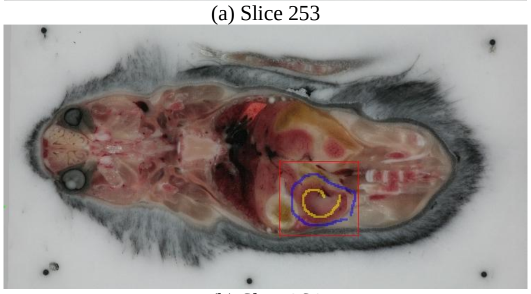
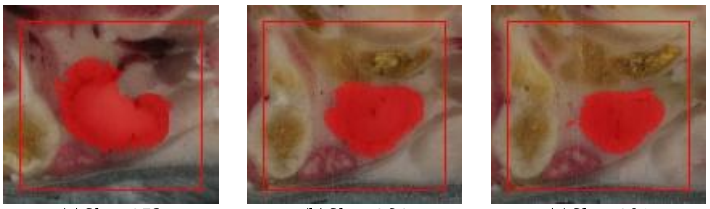

(b) Slice 264

Fig. 10 Locate the VOI and mark object seeds and background seeds.



(a)Slice 253 (b)Slice 264 (c)Slice 291

Fig. 11 Kidney segmentation result.

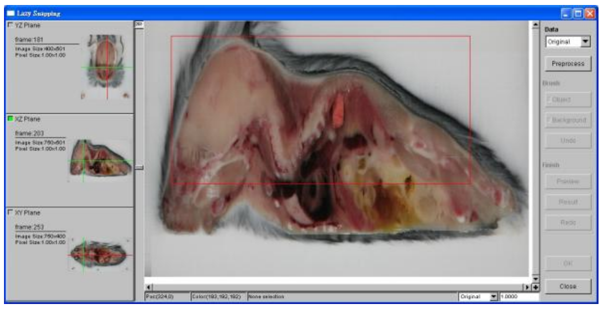
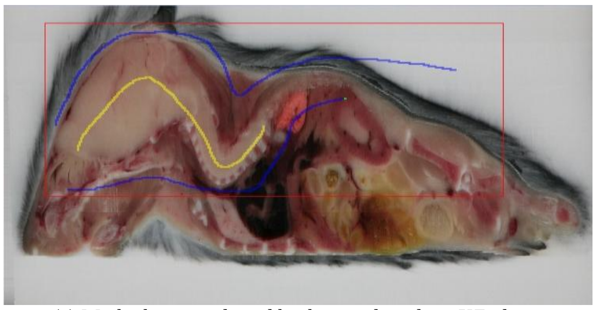
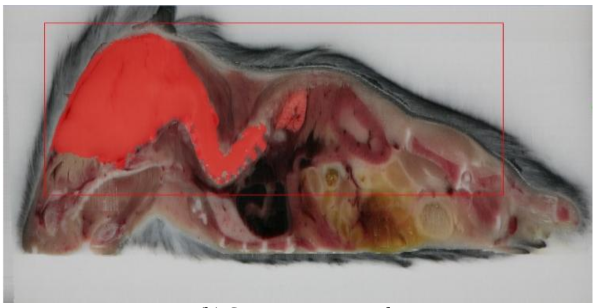


Fig. 12 Locate the VOI in XZ plane.



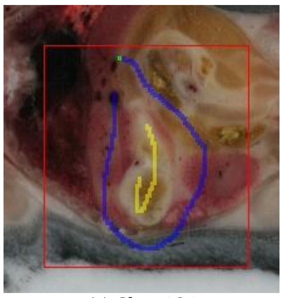
(a) Mark object seeds and background seeds in XZ plane.



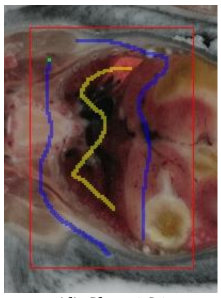
(b) Segmentation result

Fig. 13 Central nervous system segmentation.

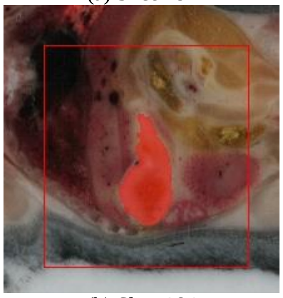
Fig. 14 shows more examples produced by 3D Lazy Snapping. Various kinds of mouse organs can be partitioned and treated as individual objects.



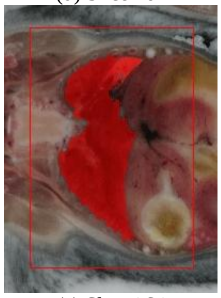
(a) Slice 284



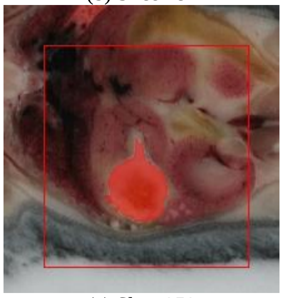
(d) Slice 264



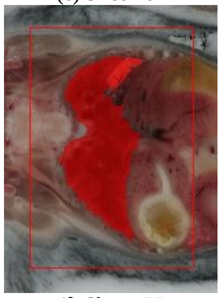
(b) Slice 284



(e) Slice 264



(c) Slice 252



(f) Slice 255

Fig. 14 More experiments: The left column shows the segmentation of stomach (a,b,c), and the right column shows the segmentation of lungs (d,e,f). The top row shows the VOI and the marking lines. The second row and the third row show the segmentation results in different slices of these two cases.

V. CONCLUSIONS

In this paper, we propose a 3D Lazy Snapping volume segmentation system. The 2D Lazy Snapping has been developed by integrating a pre-computed over-segmentation and graph cut algorithm to achieve an instant response for image segmentation. For 3D Lazy Snapping, we implement 3D watershed to over-segment the volume data and perform the graph cut on the graph which is constructed using the result from 3D over-segmentation. To save the processing time and let the user get a quick response, we process VOI that encompasses the object of interest instead of the whole volume data. An easy-to-use UI has also been developed which allows the user to browse the slices of volume data in three cutting planes and to select a spatial region as a VOI. The high-level painting tool allows the user to specify the object by marking some lines on it and the lines do not need to be on the boundary of object precisely. To evaluate the usability of our system, we perform 3D Lazy Snapping to lung cancer and mouse organs segmentation. Both cases can prove that our system works well and the user can get the result with little interaction in a short time.

We are trying to improve the system performance by applying parallel computing in data pre-processing phase and graph cut segmentation phase. Boundary editing function in [1], is a useful tool for helping the user get precise segmentation result. The boundary editing tool should also be expanded to 3D.

ACKNOWLEDGMENT

We thank Chang Gung Memorial Hospital and National Laboratory Animal Center for data supporting and consulting. We also thank Charlie H. Chang and Dr. Ching-Yao Lin for suggestions.

REFERENCES

[1] Y. Li, J. Sun, C. K. Tang, and H. Y. Shum, “Lazy Snapping," *ACM Transactions on Graphics*, vol. 23, pp. 303-308, August 2004.

[2] S. Lakare, “3D segmentation techniques for medical volumes,” Technical report, State University of New York at Stony Brook, 2000.

[3] Y. Boykov and M. P. Jolly, “Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images,” In *Proceedings of Internation Conference on Computer Vision*, Vancouver, Canada, July 2001.

[4] S. Wilson, S. Gill, A. Topol, A. Hogue, and M. Jenkin, “Lazy snapping of 3D datasets,” In *Grace Hopper Conference*, Orlando, Florida, USA, October 2007.

[5] M. Kwietnewski, S. Wilson, A. Topol, S. Gill, J. Gryz, M. Jenkin, P. Jasiobedski, and H-K Ng, “MED: A multimedia database system for 3D crime scene representation and analysis,” In *IEEE 24th International Conference on Data Engineering*, Cancún, México, April 2008.

[6] L. Vincent and P. Soille, “Watersheds in digital spaces: An efficient algorithm based on immersion simulations,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, No. 6, pp. 583-598, June 1991.

[7] H. Sun, J. Yang, and M. Ren, “A fast watershed algorithm based on chain code and its application in image segmentation,” *Pattern Recognition Letters*, vol. 26, pp. 1266-1274, July 2005.

[8] V. Osma-Ruiz, J. I. Godino-Llorente, N. Saenz-Lechon, and P. Gomez-Vilda, “An improved watershed algorithm based on efficient computation of shortest paths,” *Pattern Recognition*, vol. 40, pp. 1078-1090, March 2007.

[9] 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*, vol. 26, No. 9, pp. 1124-1137, September 2004.