Partial 3D Component Retrieval from 2D Image Slices Using Contour Structure

Saeid Belkasim, Yong Li and Xiujuan Chen

***Abstract*— Given an image stack, instead of requiring the whole 3D model, researchers may be interested in the local 3D structure of the specimens. Partial 3D component retrieval from 2D image slices represents a difficult and challenging problem. To group related objects on different layers of image slices, sequential matching of adjacent 2D objects has to be performed. In this paper we propose a new approach for object contour matching and partial 3D component retrieval based on hierarchical contour structure. Object matching involves heavy computing and is time consuming. We designed two parallel algorithms for image object matching and partially image component retrieval based on the contour structure using MPI. The experimental results show the contour structure model is suitable for 3D reconstruction from 2D image slices and our parallel algorithms achieve a good speedup.**

***Index Terms*— 3D reconstruction, edge detection, segmentation, 3D component retrieval, neuron confocal image**

I. Introduction

3D reconstruction of biological structures from image slices has been widely used in biological, biomedical research as well as disease diagnosis and treatment [1]. Many 3D reconstruction software packages have been developed in the recent years, including NEUROLUCIDA [2], a semi 3D reconstruction package for neuron anatomical analysis, and 3D-Doctor [3], a vector based architecture for 3D modeling. A shortcoming of these commercial products is that the new created 3D object lacks the capability of 3D component partial retrieval and is hard to convert 2D image slices into standard database automatically.

In this paper, we provide a new method for 3D reconstruction and 3D partial retrieval using a contour data structure. The basic idea is, for each image stack, we first segment the objects to get the object contours from individual image slices and then link the related contours on the adjacent image slices. That is, we have two stages: a separating stage, which divides individual image slices into object contours, and a grouping stage, which links the related object contours on the adjacent image slices.

In the first stage, object contours are extracted from image slices and converted into an xml database. In the xml database, each contour is represented by a node. Each node has several elements which represent contour's features, including contour layer, length, area, centroid, moments and coordinates of all the pixels on the contour. In this way, instead of having to interact with raw image data, the following algorithm only needs to take the xml database as the input. In this stage, each image slice is divided into several objects which are considered not belonging to the same 3D component.

Next step, for each object contour, we decide if there is a similar object contour on its neighbor slice. If we find one, we will make a link between them and the xml database is updated accordingly. The linking information of the object contour is represented by the elements of its corresponding node in xml database. In this stage, two adjacent image slices are linked by those object contours which are considered belonging to the same 3D component.

Now, the xml database is made of several tree structures. Each tree structure is a 3D component. Comparing with original image slices, the xml database is easy to maintain and is convenient for 3D reconstruction and partial 3D component retrieval.

Object contour matching requires heavy mathematical computation, which is time consuming. We also parallize the algorithms and achieve good speedups.

The paper is organized as follows: In Section II, we introduce image slice segmentation. In Section III, we introduce contour matching between adjacent slices. In Section IV, we introduce xml image database and 3D object reconstruction and partial 3D retrieval. The experimental result is discussed in section V.

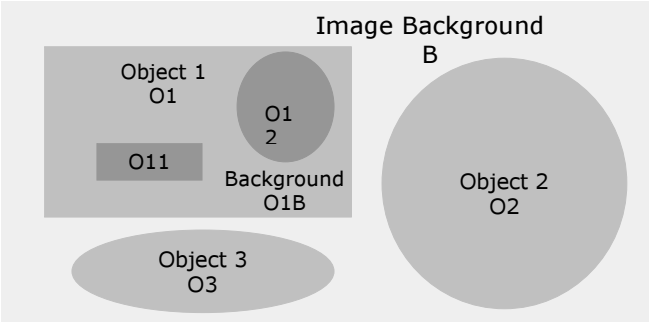
II. Image Slice Segmentation

In this project, since we need not only reconstruct the 3D model but also implement 3D component segmentation for 3D partial retrieval, a novel tree mapped data structure is proposed using contours. The procedure to map the whole image stack data set into the new tree structure can be divided into two stages: separating and grouping. In this section we will discuss the first stage. The separating stage creates xml contour database for each image slice. This process includes image enhancement, optimal thresholding, edge detection, and object contour segmentation. The segmented image objects are saved as boundary contour coordinates and sequentially ordered to form a hierarchical xml database for each image slice. The data structure has a format similar to the one described in Fig. 1 where the images are divided into sequentially ordered contours. The same procedure is repeated for every image slice of 2D image stacks.

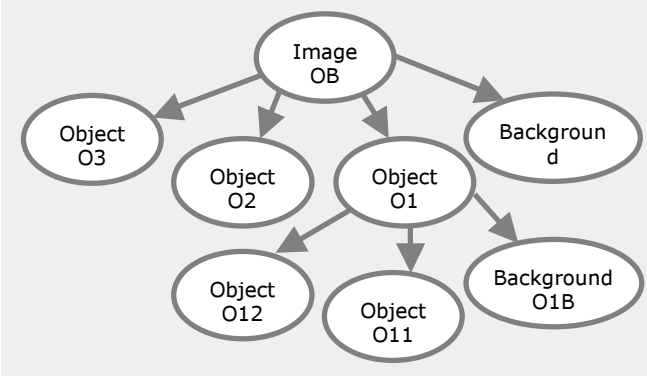
Several methods can be effectively used to detect edges [4] [5]. The main problem with these methods is the lack of

S.O. Belkasim, Y. Li and X. Chen are with the Department of Computer Science, Georgia State University, P.O. Box 3994, Atlanta, GA 30302-3994, USA (e-mail: sbelkasim@cs.gsu.edu, yli11@gsu.edu, and xchen8@gsu.edu

continuity of edges, which requires post-processing to link the broken edges. The linking algorithms may introduce unnecessary ambiguity and incorrect links of noisy data. In this project, we use automatic segmentation method introduced in [6] [7]. The optimum automatic thresholding procedure is combined with edge detection to produce a continuously connected object border and leads to a fully segmented image.



(a)



(b)

Fig. 1. (a) The sequential contour assignment (b) The hierarchical tree data structure representation of (a)

After implementing optimum thresholding, we obtain the binary data for each image slice. Next, we use an 8-connective path template to link contour pixels of the object boundary. The contour is recognized by the coordinators of its pixels and saved as an object node in the xml database. We also use a contour length filter to remove the tiny contours, which are considered to be noise. Contour length filter is not only useful for image noise removal but also very helpful to simplify the contour matching process [8]. In this stage, original image files are one-time processed and stored in an xml file. Comparing with raw image files, the xml database is better for the image retrieval and analysis. Since segmented image object boundary contours are the basic units in the database, it is convenient for us to extract the information in which we are interested from the database instead of handling the whole image files. Fig. 2 summarizes the above steps.

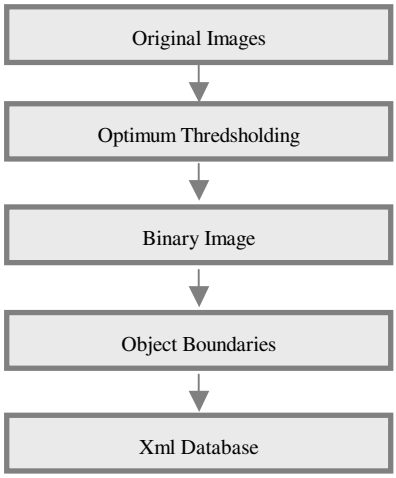


Fig. 2 Flowchart of image stack preprocessing

III. Contour Matching between Adjacent Slices

In the contour data structure, 2D images are stored in the xml database in a sequential order, slice by slice. Fig. 3 displays two adjacent confocal microscopic image slices of crayfish neuron and their corresponding contours regenerated from the xml database. It is worth noticing here that every image regardless of its complexity can be represented with a contour data structure. The main advantage of using contour data structures is that it can be efficiently used in finding the relationship between objects on adjacent layers. For example, in Fig. 3, the pointed contours obviously belong to the same 3D component and this relationship is represented as a link element in the xml database.

The grouping stage is to connect the object contours on the adjacent image slices and map the whole image stack dataset into a new tree structure represented by the xml database. The object features such as contour length, area, centroid and moments are important in implementing contour object matching between two adjacent layers. These features are calculated in the previous stage.

2D object recognition and matching is very important in many areas and many methods have been proposed such as template matching, string matching, shape-specific point matching, principal axis matching, dynamic programming, mutually-best matching, chamfer matching, graph matching, relaxation, elastic matching, and etc [9] [10]. To simplify our matching problem, we assume that each two contours belonging to the same 3D component in two adjacent slices will have the similar shapes. This essential assumption is made based on the fact that adjacent slices are very close and adjacent contours of the same object will differ by very few pixel points. The experimental measurements taken from this project indicate that the distances between two adjacent slices are around 2.1*um*. In most cases, this assumption is valid particularly when the contour shape

changes gradually and continuously. Fuzzy logic system (FLS) is used to help in this case to refine the matching decision. The inputs of the FLS include contour length, area moments, contour centroid and overlapped areas of two contours.

The two stages are combined in a tree model where each node in the tree is a representative of a segmented object and each edge in the tree is a representative of contour matching between two objects on adjacent slices.

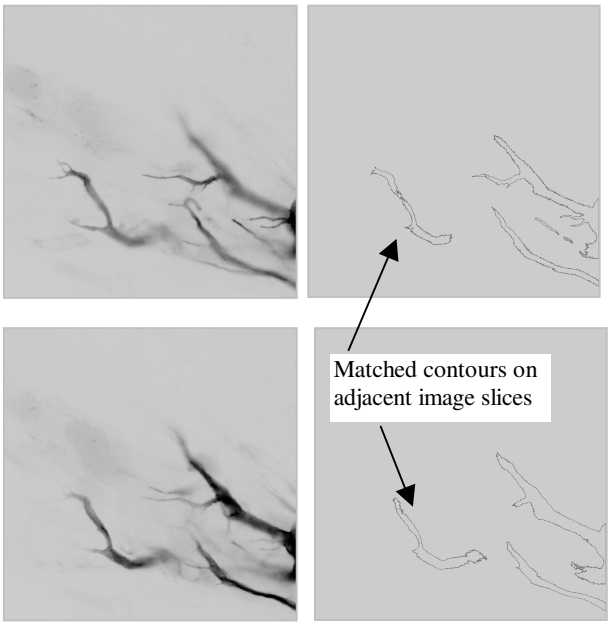


Fig. 3. (a) Original Confocal microscopic image slice of Crayfish neuron
(b) Enhanced contour image generated from the xml database

IV. 3D Reconstruction and Partial 3D Retrieval

In this project, instead of interacting with original image slices, 3D object reconstruction and partial 3D component retrieval are based on the xml image database. The procedure of describing a 3D image stack using the contour xml tree data structure is stated in the following scenario:

1. Each segmented object in the image slice corresponds to an object node in the xml file.
2. In the xml database, an object node has several elements which define the object features.
3. Layer element determines to which layer the object belongs. The objects on the same slice have the equal layer values.
4. Link element indicates the centroid of the matched contours on the next adjacent layer.

Fig. 4 shows an example of how a contour object is represented in the xml database. In the xml database, each object corresponds to a contour. Contour element records the boundary pixels in the original image slice which determine the contour shape.

```
...
<Object>
<Image_Name>01\lgaff049a01002.tif</Image_Name>
<Size>2048 2048</Size>
<Layer>3</Layer>
<Link>(1516 2020)</Link>
<Centroid>1504 2032</Centroid>
<Area>1762</Area>
<Location>(1466 2033)</Location>
<Contour_Length>546</Contour_Length>
<Contour>
 (1466,2033)(1466,2034).....
</Contour>
...
</Object>
 ...
```

Fig. 4 Contour objects in the xml database

Contour features are represented by the object elements of Contour_Length, Area, Moment, Centroid and etc. They are calculated once and can be easily extracted and repeatedly used. We can add more features of contours by defining more elements in the database. The database is easily maintainable and lends itself for parallel programming. The element Link and Layer in the xml database determine the overall topology of the 3D image structure.

From the xml contour database, we can reconstruct the solid object shape by first drawing the object boundary according to each contour element and then fill the pixels inside contour boundary using the original data. Applying this process to all the contours which have the same Layer value, we obtain all the segmented objects of an entire image slice. To construct the 3D model of an image stack, we repeat the refill process for all image slices and apply surface rendering techniques to create the 3D model [11] [12]. Fig. 5 is the 3D model rebuilt from our xml database representing 20 crayfish neuron slices.

In the biological research, instead of requiring the whole 3D model, researchers may be interested in the local 3D structures of specimens. For example, in Fig. 3 we know the two highlighted contours belonging to the same neuron branch. It may be useful for certain applications to retrieve the 3D branch in which the 2D contours reside.

Fig. 5 3D model of a crayfish neuron confocal image stack

To fulfill the 3D partial retrieval, we use the xml tree

structure described above for 3D content based component retrieval. Since we use contours as the basic units to represent the 3D volumetric data, the corresponding 3D object can be divided into several components, each of which is made of a group of connected contours. Given an arbitrary pixel on a 2D image, we can easily identify its corresponding contour. By applying a contour *depth-first search* in the tree structure, we can again easily find the 3D subcomponent in which the contour resides. Our experimental data have shown that the 3D component retrieval from the contour xml database is extremely faster than retrieval from the original image slices. Fig. 6 shows the result of 3D component by querying the database using the pointed contours in Fig. 3.

Fig. 6 3D component of a crayfish neuron branch

Using the tree structure, image querying schema can be extended by defining various searching rules for the xml contour database.

3D image reconstruction and retrieval could be the bottleneck of computing performance when large amounts of image slices involved. Since the whole image stack volume dataset has been stored in the contour based xml database, an individual processor can interact with the xml database efficiently without image loading operations and preprocessing. The contour structured database makes the distribution of the contour matching tasks among multiple processors much simple. We designed two parallel algorithms for 3D image reconstruction and retrieval respectively. Since in the xml database, each contour has a Link element indicating the matched contour on its neighboring slice, and a Layer element indicating its slice level, given a contour matching or retrieval task, a processor thus can travel the tree structure database efficiently through the two xml elements to find out its corresponding object contour and repeatedly to form the 3D component without communicating with other processors. Distributing retrieval tasks among multiple processors is straightforward based on the contour xml database. We implement the parallel program in MPI on a SGI Origin 2000 machine and reach the speedups of 8.36 and 11.08 by using 16 processors for tree structure construction and 120 3D component retrievals.

V. CONCLUSION AND FUTURE WORK

In this paper we presented a novel tree structure for 3D component reconstruction and retrieval. Image boundaries or contours are proved to be more efficient in extracting and handling 3D objects. We successfully defined a preliminary model to segment image slices and group their related contours into 3D object components. Our experimental results indicated that, the contour structure is suitable for both 3D reconstruction and partial 3D image retrieval. This project represents a primary stage of a larger project aimed at performing an automatic object retrieval and quantitative analysis of neuron structure from confocal microscopy imaging database.

As future extension of this work, a larger dataset is under construction to have more tests performed to increase the confidence on our scheme and to have a database with a larger collection of biological data that can be retrieved and analyzed more efficiently.

REFERENCES

[1] A. Sarti, Ortiz de Solorzano, C. Lockett and S. Malladi, "A Geometric Model for 3-D Confocal Image Analysis", *IEEE Transactions on Biomedical Engineering*, Vol. 47, No 12, pp. 1600 – 1609, 2000.

[2] JR Glaser and EM Glaser, "Neuron Imaging with Neurolucida - A PC Based System for Image Combining Microscopy", Medical Imaging and Graphics, Vol. 14, No. 5, pp 307-17, 1990.

[3] Mehta, B.V. and Marinescu, R. "Comparison of Image Generation and Processing Techniques for 3D Reconstruction of the Human Skull", Proceedings of the 23rd Annual International Conference of the IEEE Publication, Vol. 4, pp. 3687- 3690, 2001.

[4] W. Frei and C. C. Chen, "Fast Boundary Detection: A Generalization and a New Algorithm", IEEE Transaction Computer, Vol. C-26, No. 10, pp. 988-988, 1977

[5] J. Canny, "A Computational Approach to Edge Detection", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, pp. 679-698, 1986.

[6] S. O. Belkasim, X. Hong, and O. Badir, "Content Based Image Retrieval Using Discrete Wavelet Transform", International journal of Pattern Recognition and Artificial Intelligence, Vol. 18, No. 1, pp. 19-32, 2004.

[7] S.O. Belkasim, Y. Li, E, Dogdu, X. Hong, Z. Li "Contented-Based Image Retrieval in Biological Databases". Int. Conf. on Computational Intelligence, Istanbul, Turkey, pp. 512-515, 2004.

[8] Y. Li, S. Belkasim, Y. Pan, D. Edwards and B. Antonsen, "3D Reconstruction Using Image Contour Data Structure", Proceedings of IEEE-EMBC, Shanghai, China, 2005.

[9] Wen-Yen Wu and Mao-Jiun J. Wang, "Two-Dimensional Object Recognition Through Two-Stage String Matching", IEEE Transactions of image processing. Vol. 8, No. 7, July 1999.

[10] R. C. Veltkamp and M. Hagedoorn, "State of the art in shape matching", Technical Report UU-CS-1999-27, Utrecht, 1999.

[11] D. Meyers and S Skinner, "Surfaces from Contours", ACM Transactions on Graphics, Vol. 11, No. 3, pp. 228-258, July 1992.

[12] R. A. Drebin, L. Carpenter and P. Hanrahan, "Volume rendering", Comput. Graphics., Vol. 22, no. 4, pp. 65-74, 1988.