PROJECTION PLANE PROCESSING FOR SKETCH-BASED VOLUME SEGMENTATION

Shigeru Owada, Frank Nielsen, Takeo Igarashi

Sony Computer Science Labs, Inc.
3-14-13, Higashigotanda, Shinagawa-ku,
Tokyo, Japan 141-0022
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, Japan, 113-0033

Ryo Haraguchi, Kazuo Nakazawa

National Cardiovascular Center
5-7-1 Fujishiro-dai, Suita,
Osaka, Japan, 565-8565

ABSTRACT

Selecting a region of interest (ROI) within unsegmented volume data is one of the fundamental operations in volume data processing and analysis, yet it is difficult to perform the task efficiently. This paper proposes several simple and intuitive sketching user interface tools for the selection task, in which the user can directly click or draw a stroke on the volume-rendered object on the screen. The main contribution is that the user's input is pre-processed in 2D domain before applying the traditional 2D-to-3D stroke elevation algorithm (the Volume Catcher system [1]). We tested the system with real-world examples to verify the effectiveness of our approach.

Index Terms— User interface, Volume graphics, Segmentation

1. INTRODUCTION

Volume segmentation is the process of splitting volume data, usually provided as a stack of cross-sectional images, into several perceptual or semantic units. The procedure is fundamental to obtaining useful information from the volume region, such as shape, connectivity, and various measurements, including surface area, cubic volume, and number of components. Although the importance of volumetric segmentation has been recognized for decades and many sophisticated segmentation algorithms have been proposed, no fully automated method is yet available because segmentation remains dependent on the observer's subjective interpretation, which is impossible to obtain without user intervention. Most segmentation methods have focused on low-level features, such as edge detection and texture analysis, and have achieved some degree of success. The difficulty is in high-level recognition that is related to the semantics of data. Therefore, it is crucial for the user to give appropriate guiding information to achieve the desired segmentation result.

From the user's perspective, one problem with 3D volumetric segmentation is that 3D guiding information is difficult to specify, as the typical input device is a mouse, which provides only 2D information. Most existing systems deal with this problem by allowing the user to access the cross-section of volume data [2]. However, one drawback of this strategy is that it requires tedious 3D interaction: the user has to specify the cutting plane and then provide guiding information on that plane.

Two methods address this issue by supplying information on the projection plane (the volume-rendered image) directly and automatically inferring depth information by analyzing the data [1, 3]. These two systems offer different user-interaction techniques. Owada et al.'s Volume Catcher system draws a contour of the ROI, while Yuan et al.'s Volume Cutout system provides a rough sketch of the foreground and background regions. These two methods can significantly simplify the volume segmentation process, but they remain "point designs" in the large design space of 2D sketching interfaces for 3D volume segmentation. Many interfaces have been proposed for 2D image segmentation [4, 5]; these 2D user interface designs should also be applicable to 3D segmentation domain.

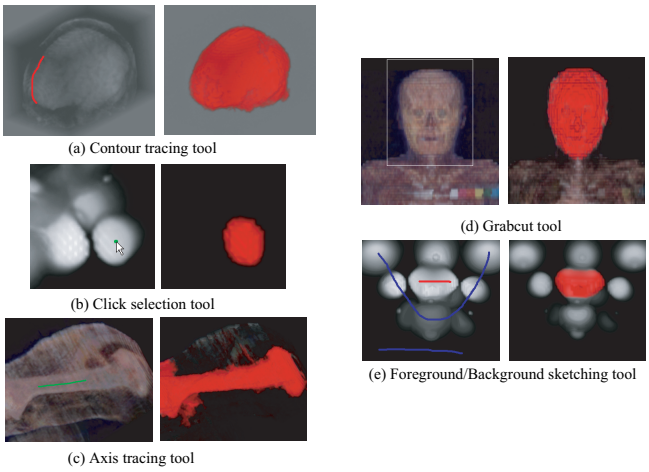

Fig. 1. The proposed volume segmentation tools. Our framework can potentially be combined with many more user interface tools.

In this study, we generalize these latter two approaches and propose a wider variety of sketching interfaces for volume segmentation, as well as an implementation framework that can potentially incorporate a large number of existing and future user interface techniques. Our system currently supports the following five user interfaces: contour tracing; direct clicking to select small isotropic regions; tracing the central axis to select a tubular target; boundary rectangle selection; and foreground/background sketching (Figure 1). Our imple-

mentation framework consists of a 2D image processing module as the front end and an extension of the Volume Catcher system as the back end segmentation module. The 2D image processing module takes the user's 2D gestures (e.g., clicks and strokes) and generates a set of 2D curves that represent the contour fragments of the ROI by examining the rendered image. The generated curves are then passed to the extended Volume Catcher system to return the desired ROI. This workflow is illustrated in Figure 2. The Volume Catcher system is suitable for the task because it can segment the volume data using only partial contours, whereas the approach used in the Volume Cutout system requires the entire portion of the ROI to be visible. This implementation framework can also easily accommodate other 2D region selection or boundary selection interfaces, such as grammar-based segmentation [6].

As in the original Volume Catcher system, our system requires an external 3D segmentation module that takes point constraints as input, which specify the inside or outside of the desired region. The performance of our system strongly depends on which 3D segmentation algorithm is used. Although we used the 3D statistical region merging technique [7], many other algorithms could potentially be used. Therefore, it is beyond the scope of this paper to perform a quantitative performance evaluation of our system.

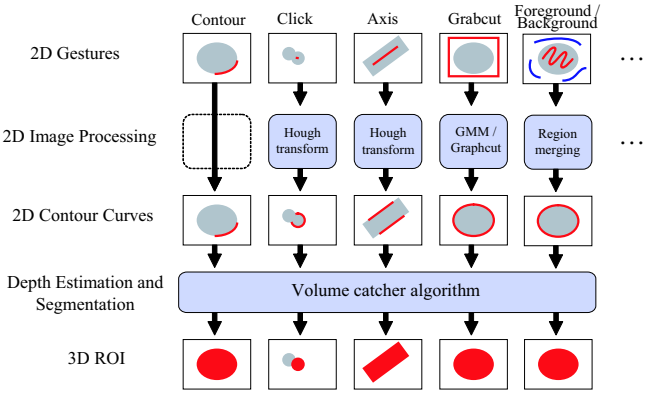

Fig. 2. The system overview

2. USER INTERFACE

After the user loads a volumetric model, the system renders it using a traditional volume-rendering method. The user can apply any rendering technique to enhance the appearance of the model [8]. Since our system applies 2D image processing to the rendered image, it is important to adjust these parameters and make the target region as visible as possible.

Our current implementation supports five interface tools for selecting an ROI: contour tracing, click selection, axis tracing, GrabCut, and foreground/background sketching. Contour tracing is the primary user interaction in which the user's input is sent directly to the original Volume Catcher algorithm. The other tools first apply 2D image processing methods to extract 2D contour curves, which are then sent to the Volume Catcher algorithm.

Contour-tracing tool (Figure 1a)
This is the tool used in the original Volume Catcher system [1]. The user traces a section of (or the entire) contour of the target ROI using a 2D freeform stroke.

Click selection tool (Figure 1b)
This tool is a shortcut operation for selecting a small, spherical ROI. The user simply clicks near the center of the ROI, and the system returns the target 3D ROI as the output. This tool allows the user to select multiple ROIs with successive clicks, so it is especially useful when selecting many small spherical ROIs, such as beans. This tool assumes that the target region is nearly spherical and that the user clicks near the center.

Axis-tracing tool (Figure 1c)
This tool is designed for selecting tubular regions, such as blood vessels, nerve fibers, and long bones. The user traces the central axis of the target ROI, and the system returns the corresponding 3D tubular region. This tool assumes that the target tubular region has a consistent radius within the target area. If this is not the case, the user is advised to use the contour-tracing tool instead.

GrabCut tool and foreground/background sketching tool (Figure 1d,e)
These user interfaces are borrowed from previous systems [5, 4]. When using the GrabCut tool, the user specifies a boundary rectangle that encompasses the ROI. When using the foreground/background sketching tool, the user draws a few strokes that indicate the inside and outside of the target ROI on the screen.

3. ALGORITHM

Our system involves multiple steps (Figure 2). First, 2D image processing modules take 2D gestures and generate partial or entire 2D contours of the target ROI. Then, a depth-estimation module estimates the depth of these 2D contour curves. Finally, the volume-segmentation module places constraint points inside and outside the 3D contour and returns the segmented target ROI. The last two modules are basically identical to the original Volume Catcher system. The following section describes these modules in detail. Note that this workflow is quite general and consequently it is easy to extend the user interface by incorporating additional 2D preprocessing techniques.

3.1. 2D preprocessing

Most of our tools (except for the contour-tracing tool) are implemented as a plug-in 2D preprocessing module that takes 2D gestures as input and returns one or more contour fragments of the target ROI as output. Since contours are a common output of 2D segmentation techniques, almost all existing 2D image segmentation techniques can be adopted to our framework. We perform hard segmentation (binary segmentation), where each pixel belongs to either the foreground or background.

As mentioned previously, the Volume Catcher system can accept open or even disconnected contour input. Therefore, unclear portions of the contour (resulting from occlusion, imprecise input, or other reasons) can be omitted, and the missing portions can be found later using the volumetric segmen-

tation algorithm, which uses much richer information than a single 2D image.

Contour-tracing tool

The contour curves drawn by the user are input into the Volume Catcher algorithm directly; therefore, the image processing module does nothing in the current implementation.

Click selection tool

The click selection tool computes a circle that best approximates the contour of the circular region centered at the clicked point. To do this, a Sobel filter [9]generates a differential image, and then a search for the optimal circle along which the average differential value is maximized is performed (Figure 3b, c). This is similar to the standard Hough transform algorithm [9].

Axis-tracing tool

This tool is again implemented using a the Hough transform-like technique. The output is two side strokes that trace the contour of the tubular region, generated by displacing the original axis stroke sideways with a certain offset value. The objective is to find the optimal offset value with which the two side strokes are located where the differential values are large. Since only the sides of the axis are of interest, the distance field of the curve is calculate using the Vector Distance Transform (VDT) algorithm [10], and then each pixel is checked to determine whether the closest point on the stroke is one (or both) of the two endpoints of the stroke. If so, the pixel is removed and ignored for the subsequent operations (Figure 3e). The distance transform algorithm computes the closest point automatically as the distance field image is generated.

Finally, the pixels are classfied according to the distance values assigned and the best offset value is found by voting. The computed partial contours are shown in Figure 3f.

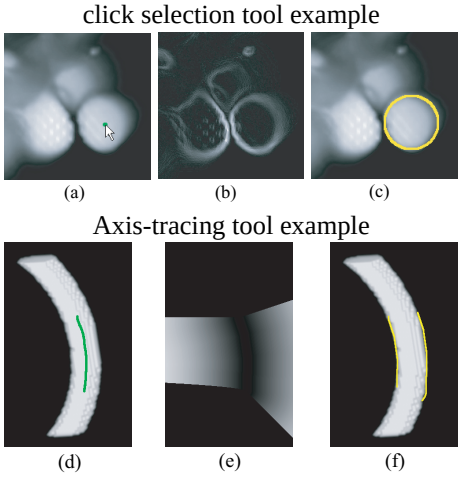

Fig. 3. Examples of the click selection tool (a-c) and axis-tracing tool (d-f)

GrabCut tool

The GrabCut system takes a bounding rectangle of the ROI as input, finds initial foreground and background pixels, and then alternately performs fitting to Gaussian Mixture Models and Graph Cuts until convergence [5]. Since border matting is not performed, the result is given as a binary image, each pixel of which is indexed as foreground or background. The raw output of the algorithm usually contains a jagged boundary. Therefore, it is smoothened using the opening operation in mathematical morphology [9]. Finally, the outermost contour of the foreground region is traced and input into the Volume Catcher algorithm.

Foreground/background sketching tool

This user interface essentially specifies a small number of pixels that are clearly categorized as foreground or background. A variety of algorithms are available to segment volume data based on the information [4]. We currently use a statistical region-merging technique [7], and then apply the opening operation to the segmentation result to smoothen the boundary, extract the outermost contour, and send it to the Volume Catcher algorithm.

3.2. Elevation of 2D contours and segmentation

This process takes the partial or complete contour of the target ROI in the screen space and estimates its depth so that the resulting 3D contours are located near the boundary of the target ROI. Then the volume data is segmented using the 3D contours information. This is almost identical to the Volume Catcher algorithm [1], with the exception to support discontinuous or looped contours.

If two or more contour fragments are given initially, it is necessary to correlate them. If we apply the 2D and 3D algorithms to them independently, the resulting 3D paths may have substantially discontinuous depth values at the boundary. This means that the resulting 3D strokes may stick to separate objects in the data, which is undesirable. Therefore, we connect the curved sweep surfaces associated with the disjoint contour fragments when computing the optimal depth value. That is, we map them into a single parameter space and obtain a single continuous path in the space, guaranteeing that the depth values for the computed 3D strokes are continuous. If the contour stroke is closed (whether the stroke is single or consists of multiple fragments), it is also important to guarantee continuity of the depth values of the end points that correspond to the top and bottom rows in the parameter domain. This is illustrated in Figure 4. To satisfy this requirement, we introduce an additional condition of $\mid f(y_1) - f(y_{max}) \mid \leq c$ when computing the optimal path in the parameter space. Although the dynamic programming is performed only once for a single open stroke, it must be performed X_{max} times in this case. To find the global maximum, the optimal path that starts from each point on the edge $y = 1$ is determined, and then the costs for paths that satisfy the new condition are checked.

Note that the degree of continuity to be maintained at the sweep surface boundary and other places should differ. It is also possible to control the continuity of the resulting 3D path around the sweep planes boundary by modifying the continuity constant c locally.

4. RESULTS AND DISCUSSION

We applied our technique to several datasets. Figure 5 shows results using our new sketching user interface tools. In most cases, the user can be less careful in providing 2D information. The click selection tool is especially useful, more so

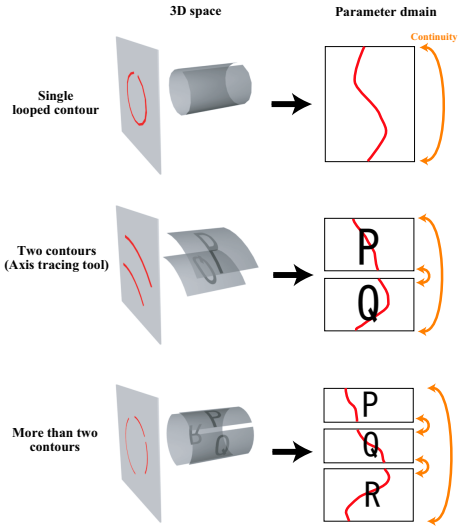

Fig. 4. Parameterization of sweep planes for various cases. Separate sweep planes are mapped to a single parameter domain to make the boundaries continuous. In addition, the top edge and the bottom edge of the parameter domain are also connected.

than expected, because it works robustly and also most objects have a specific direction from which the object is seen as a circle. The foreground/background sketching tool is also useful because the 2D segmentation algorithm tends to return good results. Conversely, the axis-tracing tool is not very efficient because it requires a relatively precisely traced central axis of the ROI. It may be less intuitive, but more reliable, to trace the contour of the tubular region directly using the contour-tracing tool. The GrabCut tool was also less effective than the others, because of the poor 2D segmentation ability of volume-rendered images. We needed to additionally specify foreground and background regions to obtain the correct result in Figure 5c, as suggested in the original paper [9]. This may have been caused by the limited color variation of the data we used.

Based on these observations, we conclude that the quality of the result is strongly affected by the quality of the 2D segmentation results, which stems from a good choice of segmentation algorithm and visualization parameters, such as opacity/color transfer function, or camera parameters.

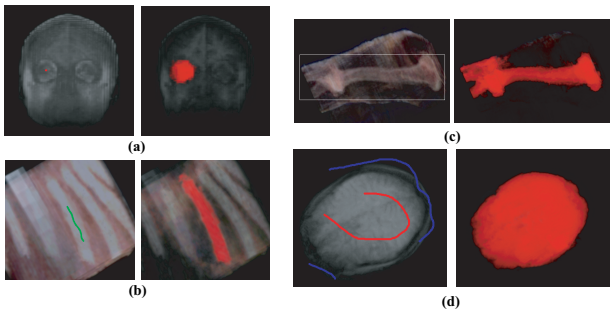

Fig. 5. Results

5. LIMITATIONS AND FUTURE WORK

Our system can easily and intuitively segment volumetric data through unprecedented user interfaces. However, we did not apply our technique to solve real problems facing professional users. Therefore, a large-scale evaluation test is necessary to make our system useful to targeted users.

Other future plans include implementing a mechanism to fix partially failed results, applying surface representations, and designing better user feedback.

6. ACKNOWLEDGEMENTS

Part of our dataset was taken from the Visible Korean Human dataset (http://vkh3.kisti.re.kr/new). We thank Justin Talbot for distributing his implementation of the GrabCut system (http://www.justintalbot.org).

7. REFERENCES

[1] Shigeru Owada, Frank Nielsen, and Takeo Igarashi, "Volume catcher," in *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, New York, NY, USA, 2005, pp. 111–116, ACM Press.

[2] Fan-Yin Tzeng, Eric B. Lum, and Kwan-Liu Ma, "A novel interface for higher-dimensional classification of volume data," in *Proceedings of IEEE Visualization 2003*. 2003, pp. 505–512, IEEE.

[3] Xiaoru Yuan, Nan Zhang, Minh X. Nguyen, and Baoquan Chen, "Volume cutout," *The Visual Computer (Special Issue of Pacific Graphics 2005)*, vol. 21, no. 8–10, pp. 745–754, 2005.

[4] Yuri Boykov and Marie-Pierre Jolly, "Demonstration of segmentation with interactive graph cuts.," in *ICCV*, 2001, p. 741.

[5] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake, ""grabcut": interactive foreground extraction using iterated graph cuts," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 309–314, 2004.

[6] Feng Han and Song Chun Zhu, "Bottom-up/top-down image parsing by attribute graph grammar.," in *ICCV*, 2005, pp. 1778–1785.

[7] R. Nock and F. Nielsen, "Grouping with bias revisited," in *IEEE International Conference on Computer Vision and Pattern Recognition*, A. Bobick L.-S. Davis, R. Chellapa, Ed. 2004, pp. 460–465, IEEE CS Press.

[8] Barthold Lichtenbelt, Randy Crane, and Shaz Naqvi, *Introduction to volume rendering*, Prentice-Hall, Inc., 1998.

[9] John C. Russ, *The Image Processing Handbook Fourth Edition*, CRC Press, 2002.

[10] James C. Mullikin, "The vector distance transform in two and three dimensions," *CVGIP: Graph. Models Image Process.*, vol. 54, no. 6, pp. 526–535, 1992.