Interactive Volume Cutting

Kevin Chun-Ho Wong Tommy Yu-Hang Siu Pheng-Ann Heng Hanqiu Sun

Department of Computer Science and Engineering,
Chinese University of Hong Kong, Shatin, N.T. , Hong Kong
chwong@cse.cuhk.edu.hk, yhsiu@cse.cuhk.edu.hk
pheng@cse.cuhk.edu.hk, hanqiu@cse.cuhk.edu.hk

Abstract

In 2D, Intelligent Scissors is an efficient interactive tool for image segmentation. By interactive use of a dynamic-programming graph-searching algorithm, a region of interest in the image can be accurately obtained. In this paper, we introduce the use of Intelligent Scissors for contour detection on a volumetric data surface. It is fast enough to be used in an interactive virtual environment, in which the user can intuitively select the contours in volumetric data in an accurate and robust manner. Moreover, we extend our work to the volume data manipulation, cutting off the interesting part of the volume by providing a contour on its surface. The cutting surface is computed by a fast dynamic programming algorithm. By using this tool, many new volumetric data models can be created from an existing one in an effective way.

Keywords: Volume Visualization, Interactive Volume Segmentation

Introduction

Interactive volumetric data manipulation is an important focus in recent Visualization and Graphics interface research. *Galyean and Hughes*[3] developed a desk-top Polhemus-based system for manipulating a volumetric model interactively. *Serra et al.*[11] proposed a general system for free-form creation of 3D objects with given volume data. *Wang and Kaufman*[14] proposed interactive volume sculpting techniques to carve beautiful models from a textured volumetric block. However, the existing volume sculpting tools rarely provide smooth, intuitive and accurate cutting.

In 1995, *Mortensen and Barrett*[7] proposed an interactive tool for 2D image segmentation, called Intelligent Scissors, by which users can easily and accurately outline the region of interest (ROI) in an image. We borrow these ideas and design a new, intuitive and accurate 3D volume cutting methodology.

To cut off the volume of interest, the user first draws a closed contour on the volume surface as the boundary. Then, a cutting surface will be generated by a fast dynamic-programming computation. The user-defined boundary is similar to a wire loop with an arbitrary shape. When this is taken out of soapy water, a film spanning the loop with minimum area is formed. The shape of the cutting surface which aims to separate two volumes is similar, but it is driven to voxels with high gradient magnitude.

In medical data visualization, interactive volume cutting can provide various views of the internal structures of the human body. In volume data editing, an accurate volume data cutting can also benefit the artists to design more complicated volume model by cutting and pasting existing volume data. These applications explain the motivation of our work.

This paper discusses how to detach a three dimensional ROI from a given volume intuitively, correctly and efficiently. The next section addresses some basic issues concerning contour detection on a volume surface, and introduces the use of Intelligent Scissors on the volume model surface. Section *Volume Cutting* discusses the computation of the cutting surface with dynamic programming techniques, as well as some topological problems occurring when arbitrary shapes of volume and contour are permitted. We illustrate its use in Section *Implementation and Results*, and give conclusions in Section *Conclusions*.

Contour Selection

There are two straightforward approaches to generalizing the Intelligent Scissors technique (described below) to volumetric data. The first is to apply the original 2D version on the screen buffer to detect the contour. The second is to consider the volume data set as a large 3D graph and then search for the shortest path (contour segment) between two nodes (voxels) with Intelligent Scissors.

Both methods have their disadvantages. For the first, since the cost function is evaluated by the information (such as depth and intensity) in the screen buffer, its values depend strongly on the direction of view. The starting point will be lost if the viewpoint is changed during the use of Intelligent Scissors to define a segment. For the

second method, although the cost function is view independent, the computational time is extremely large. It is not suitable for interactive application.

Therefore, we only deal with the surface voxels in volumetric data. We model these voxels as a graph and apply a graph searching algorithm to find a global optimal boundary from a seed point. This 'Intelligent Scissors' technique originated in the field of image processing [4]. We apply it to 3D iso-contour detection, and implement it in the Virtual WorkBench[9, 10], a virtual environment with 3D display and input (see Figure 5). With this interface, we can mark a particular surface voxel as the start point, and different paths can be displayed while the 3D stylus moves along the surface. This helps the user select the most suitable iso-contour interactively.

Intelligent Scissors

In 2D image segmentation, 'Intelligent Scissors' has been proved to be an efficient method. In the context of volume visualization, we modify the algorithm so that it can be applied on a graph generated from the isosurface of volumetric data. The points in the mesh contain not only their positions but also the interpolated gradients, which will be used to evaluate the cost function, discussed below. Following the idea of Intelligent Scissors, we calculate global optimal paths by Dijkstra's shortest path algorithm.

To use Dijkstra's algorithm, we have to define local costs. In image processing, these depend on the 2D gradient at the pixel positions. In [12], edge detection measures the strength indicator G of neighbor pixels,

$$G = \sqrt{I_x^2 + I_y^2} \tag{1}$$

where a pixel is preferable if its G is large. Thus the local cost between pixels u and v can be formulated as

$$\text{cost}(u, v) = \max(G) - \frac{1}{2}\big(G(u) + G(v)\big) \tag{2}$$

To find a contour fitting the surface, however, geometric information is more important than intensity. Therefore, we create the local cost between vertices u and v according to their gradient magnitudes and the dot product of their gradient vectors. The resulting cost function is

$$\text{cost}(u, v) = \|p_u - p_v\| \big(w_g * f_g(u, v) + w_n * f_n(u, v)\big) \tag{3}$$

$$f_g(u, v) = 1 - \frac{G(u)+(G(v)}{2 \max(G)} \tag{4}$$

$$f_n(u, v) = \frac{1 - Nu \cdot Nv}{2} \tag{5}$$

where p_u, p_v, N_u, N_v are the position vectors and normal vectors at u and v respectively, and w_g and w_n are the weighting factors controlling the influence of f_g and f_n. The gradient vector at a vertex is evaluated by trilinear interpolation between gradients at nearby voxels. By using this formula, the Dijkstra algorithm would tend to find a path which has large gradient change.

Dijkstra's algorithm

We use the Dijkstra algorithm [2] to do the searching. This algorithm mainly does 2D dynamic programming to find all paths from all points to the seed point s that are globally optimal; *i.e.*, such that the sum of costs along each path is minimal. The search need not finished a full pass, but can be stopped when the search reaches the position selected by the 3D stylus. This can save much time since unnecessary paths would not be calculated. The user rarely moves the stylus away from s faster than the wave of Dijkstra results is computed, so the earliest results are exactly those needed early. If the seed point is changed, the searching must be started again. Time can be saved here since not all the points need to be recalculated. For example, if u is the seed point and v is the new seed point, all points passed through the route from u to v need not be re-calculated. If $p \rightarrow v \rightarrow u$ is the shortest path from a point p to u, then $p \rightarrow v$ must be the shortest path from p to v.

Dijkstra's algorithm (Table 1) uses dynamic programming to update the cost of each point step by step. For each point p, a pointer points to the neighbor through which passes the shortest path from p to the seed point s. Thus a path from any selected point to s can be established quickly. Note that the cost function c(u, v) can be preprocessed and the only changed function is B(u), which indicates the path from u to s.

Volume Cutting

As the user defines a closed contour on the volume surface, the ROI can be selected by estimating the shape of the *external surface* and the *cutting surface*. All voxels belonging to the selected ROI are bounded by these two surfaces. Figure 1 shows how to select it using contours.

The *external surface* is the part of the whole volume surface, which can be estimated by any mesh generation algorithm [8] and surrounds the interested volume when combined with the *cutting surface*. The shape of this cutting surface is influenced by the shape of the specified contour (boundary), the gradient of the voxels near this surface, and its smoothness.

For surface modeling, a deformable mesh [1, 6] can be used to fit the surface. However, the shape of the given contour cannot provide enough information about the topology and the connectivity of this mesh. To generate the junctions in the mesh, the given contour will be projected on a discrete grid surface. In this projection,

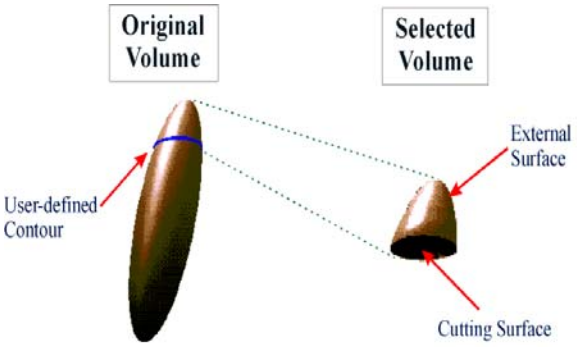


Figure 1: The selected volume is bounded by two surfaces (external and cutting surfaces). Their common edge is defined by the user.

Definitions:

s	Seed point.
L	List of active nodes.
B(u)	Back pointer from (u) indicates the potential optimal path.
P(u)	TRUE if node u is made permanent.
T(u)	Total cost from u to s.
c(u, v)	Local cost of edge $u \rightarrow v$.
min(L)	Get the node with minimum total cost from L and remove it.

Algorithm:

P(u) $\leftarrow$ FALSE for all u
T(s) $\leftarrow$ 0, T(u) $\leftarrow \infty$ for $u \neq s$
L $\leftarrow$ {all nodes}
while L $\neq \emptyset$ **do**
  $q \leftarrow$ min(L)
  P(q) $\leftarrow$ TRUE
  for each edge $q \rightarrow v$ s.t. P(v)=FALSE **do**
    if T(v) $>$ T(q) + c(q, v) **then**
      T(v) $\leftarrow$ T(q) + c(q, v)
      B(v) $\leftarrow q$
    end if
  end for
end while

Table 1: Dijkstra's algorithm for contour detection.

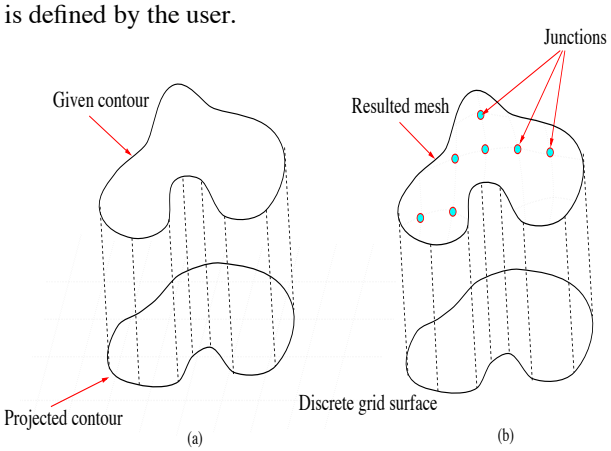


Figure 2: (a) A contour is projected on a discrete grid plane. (b) The junctions of the surface mesh is generated.

each junction can be bijectively mapped to an unique grid within the interior region of the projected contour (Figure 2). To increase sampling detail, the projected surface is rotated to an optimal orientation such that the number of junctions in the mesh is maximum. When the contour cannot have an one-to-one projection to the projected surface, we can divide it recursively into several smaller contours until all junctions in the cutting surface can be generated.

Cost function for the cutting surface

Two vital factors, the continuity between junctions and the voxel gradient near them, are considered for defining the cost function of the mesh. The first factor makes the surface smooth and flat. The second factor drives the surface to fit the iso-surface in the volume. Its function is similar to the image force used with an active contour [5] to fit the edge in a 2D image. The cost function $C(x, y, z)$

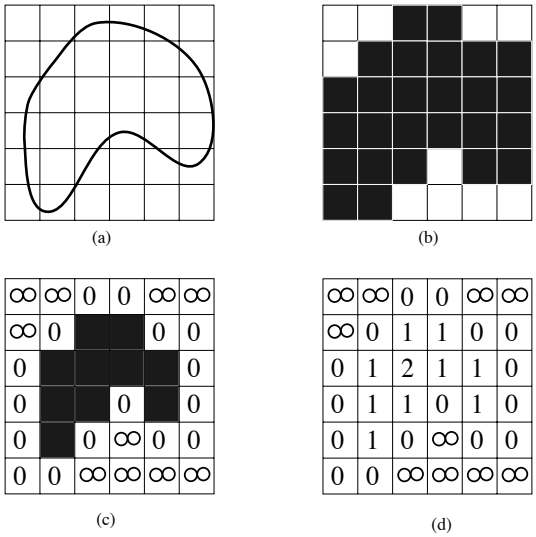


Figure 3: (a) A projected contour. (b) The interior region. (c) $L(x, y)$ will be assigned the value 0 when (x,y) is an edge pixel. (d) The resulting function $L(x, y)$. The region is separated into discrete contours by different numbers.

for a voxel with the intensity $I(x, y, z)$ can be defined by

$$C(x, y, z) \;=\; -\alpha\,|\nabla I(x, y, z)| + \beta S(x, y, z)\,, \qquad (6)$$

where α and β are the weighting parameters for the iso-surface fitting factor and the surface continuity respectively, ∇ is the gradient operator, and $S(x, y, z)$, discussed in next Section 'Continuity function', measures the continuity of the voxel at (x, y, z) with its neighbors on the mesh. Let $\mathcal{M}$ be the set of voxels involved in the cutting surface. Then the cost for a cutting surface $C_{\mathcal{M}}$ is defined as follows:

$$C_{\mathcal{M}} = \sum_{(x,y,z)\in\mathcal{M}} C(x, y, z) \qquad (7)$$

In general, the continuity factor $S(x, y, z)$ can be defined as the average value of the distances from itself to all of its neighbors. However, this definition is not efficient enough when seeking the minimum total cost, and can also make the resulting mesh too flat. We suggest a new continuity function, to speed up the computation.

Continuity function $S(x, y, z)$
All pixels within a closed loop on the grid surface can be classified by their shortest distances (measured in 'pixel hops') to the boundary. In other words, this closed region can be divided into many contours. As in Figure 3, these contours are considered as pixel chains on the discrete grid surface. In our algorithm, the continuity function

step	instruction
1.	$c \leftarrow 0, \forall x, y,\ L(x, y) \leftarrow \infty.$
2.	$\forall(x, y)$ within the projected contour, $I(x, y) \leftarrow 1.$
3.	For all edge pixels (x, y), $L(x, y) \leftarrow c$ $I(x, y) \leftarrow 0$
4.	**If** all $I(x, y) = 0$ **then** exit.
5.	$c \leftarrow c + 1$, **goto** 3.

Table 2: The algorithm to calculate the map of $L(x, y)$.

$S(x, y, z)$ is computed by measuring the 3D distance between neighbors' contours, rather than neighbor voxels. The integer function $L(x_p, y_p)$ indicates the shortest distance from the projected point (x_p, y_p) to the boundary of the projected interior region. The map of $L(x_p, y_p)$ can be evaluated by the simple algorithm shown in Table 2.

Then, the set of voxels $\mathcal{A}_{(x,y,z)}$ involved in the continuity computation can be found by the equations

$$\mathcal{A}_{(x,y,z)} = \mathcal{N}_{(x,y,z)} \cap \mathcal{B}_{(L(x_p,y_p)-1)} \cap \mathcal{G} \qquad (8)$$

$$\mathcal{N}_{(x,y,z)} = \left\{\, (x', y', z') \mid \left\| (x'_p, y'_p) - (x_p, y_p) \right\| < \varepsilon \,\right\} \qquad (9)$$

$$\mathcal{B}_k = \{\, (x, y, z) \mid L(x_p, y_p) = k \,\} \qquad (10)$$

where (x_p, y_p) and (x'_p, y'_p) are the projections of (x, y, z) and (x', y', z') respectively, and $\mathcal{G}$ is the voxel set which has been guaranteed to be involved in the surface. The parameter ε controls the connectedness of two neighbor voxels on the projected surface. It is always a real number selected from 1.0 to 1.5. For example, if $\varepsilon = 1.0$, two neighbor voxels will have 4 connectedness on the projected surface. Since the continuity factor of the voxel at (x, y, z) is only affected by the points in $\mathcal{A}_{(x,y,z)}$, $S(x, y, z)$ can be defined as

$$S(x, y, z) = \frac{\sqrt{\sum_{\vec{v}\in\mathcal{A}_{(x,y,z)}} \|\vec{v} - (x, y, z)\|^2}}{|\mathcal{A}_{(x,y,z)}|} \qquad (11)$$

Finding the cutting surface
The mesh defining the cutting surface can be computed by minimizing the cost function $C_{\mathcal{M}}$ (equation 7). This process can be performed efficiently by dynamic-programming techniques. Our algorithm (Table 3) has the following properties:

1. The junction locations are computed from the surface's boundary to its most interior points.

step	instruction
1.	$\mathcal{G} \leftarrow \mathcal{C} \cap \mathcal{B}_0$ *// $\mathcal{C}$ is the set of voxels touching the given contour.* *// and $\mathcal{G}$ will be used to find $\mathcal{A}$ by equ. 8.* $k \leftarrow 1$ $\forall i, j\ , V[i,j] \leftarrow null$ *// $V[i,j]$ is the array of 3D vectors which* *// defines the shape of resulting cutting surface.* $\forall i, j\ , c[i,j] \leftarrow \infty$
2.	**For each** $(x,y,z) \in \mathcal{C}$ $c[x_p, y_p] \leftarrow 0$ $V[x_p, y_p] \leftarrow (x,y,z)$ *// where (x_p, y_p) is the projected point* *// of (x,y,z) on the discrete grid surface.* **end for**
3.	$\mathcal{D} \leftarrow \mathcal{B}_k$ **if** $\mathcal{D} = \oslash$ **then** exit.

4.	**For each** $(x,y,z) \in \mathcal{D}$
	Compute $\mathcal{A}_{(x,y,z)}$ by equ.8.
	$m_{tmp} \leftarrow$
	$C(x,y,z) + \frac{\sum_{(x',y',z') \in \mathcal{A}_{(x,y,z)}} C(x',y',z')}{\|\mathcal{A}_{(x,y,z)}\|}$
	if $m_{tmp} < c[x_p, y_p]$ **then**
	$c[x_p, y_p] \leftarrow m_{tmp}$
	// Here, x_p, y_p are rounded off.
	$V[x_p, y_p] \leftarrow (x,y,z)$
	end if
	end for

5.	$\mathcal{G} \leftarrow \bigcup_{i,j} V[i,j] - \{null\}$ $k \leftarrow k+1$ **Goto** step 3.

Table 3: The algorithm to compute the cutting surface.

2. By the dynamic programming technique, our minimization is a one-pass process.

3. In 3D space, an 'interior region' cannot be clearly defined by a closed contour alone. Hence, a projected surface is used to clarify this definition.

4. The minimization of the cost function $C_{\mathcal{M}}$ makes the resulting surface smooth, flat and close to the isosurface.

5. Our surface fitting algorithm, similarly to active deformable models, works with a cost minimization process.

After this algorithm finishes, the array $V[i,j]$ contains the information about the shape of mesh. All the junctions involved in the mesh are stored in $\mathcal{G}$.

step	instruction
1.	Set $\mathcal{J}$ to be the set of junctions in the mesh representing the surface of the whole volume.
2.	$\mathcal{J} \leftarrow \mathcal{J} - \mathcal{H}$, where $\mathcal{H}$ is the node set representing the given contour.
3.	Randomly select a node v in $\mathcal{J}$ which is on the external surface covered by the segments of interest.
4.	Find the set of nodes $\mathcal{C}_v$ which is connected to v.

5.	**If** $\mathcal{C}_v = \mathcal{J}$ **then**
	return false *// The cut is invalid*
	else
	return true *// The cut is valid*

Table 4: This algorithm checks if a given contour can cut one volume into two sub-volumes.

Topological problems for the volume cutting

In general, if arbitrary volumes and contour shapes are considered, many unexpected volume cuts will occur. For example, the torus in the Figure 4a cannot be separated into two parts by the contour loop shown. Since any cutting surface defined by this type of contour will go outside the volume, the cutting is invalid (shown in Figure 4b).

Another kind of problem is that mesh junctions cannot be evenly generated by a simple planar surface. Figure 4c,d shows a C-shaped contour for which it is difficult to produce a good projection on a single plane.

Handling all of these situations would lead to extremely complicated topological problems. The most direct solution in practice is to ask the user for a well-conditioned contour when a bad case has occurred.

Assumptions for the well-conditional contour used in our algorithm

Assumption 1 The user-defined contour C must separate the surface of the original volume into two parts and completely detach the external surface of the ROI. Otherwise, this volume cannot be separated by any surface spanned by C.

Assumption 2 In the projection, there should exist a single closed contour, without any self-intersection.

Assumption 3 Every junction on the resulting cutting surface should have a unique mapping on the projected surfaces.

We introduce the algorithm in Table 4 to check whether the given contour satisfies Assumption 1.

Implementation and Results

We have implemented the algorithm on the Virtual Work-Bench [10, 9], a general purpose reach-in 3D interface

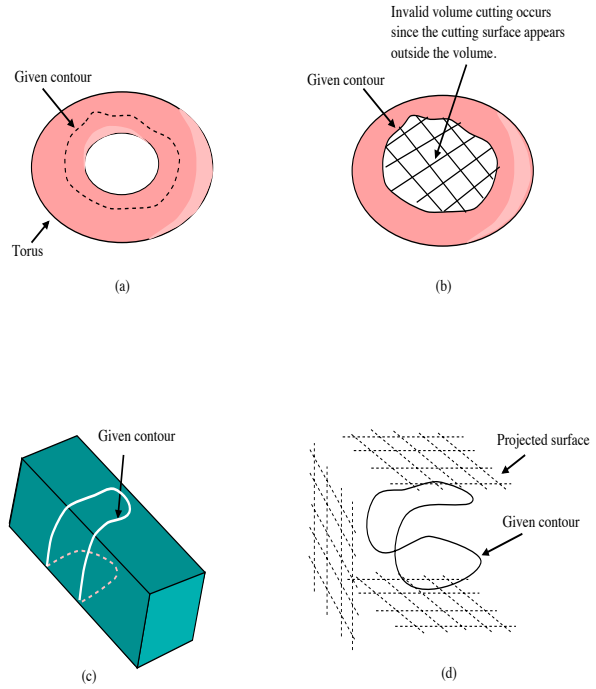


Figure 4: Figure a,b show a case of invalid cutting. The torus cannot be divided by this loop, as the cutting surface stays outside of the volume. Figure c shows a case of volume cutting with a C-shaped contour. Figure d illustrates that three projected planes are used in order to make the junctions on the mesh distributed evenly.

that supports stereo display with shutter glasses or mirrors, and 3D input with a 6DOF stylus. We choose this virtual environment because we found that a 2D display of volumetric data is seldom enough. Another main advantage of the Virtual WorkBench is that choosing a point from the mesh in 3D space, rather than via a 2D display and a mouse, can help us determine whether the contour detected is suitable. Figure 5 illustrates how user selects the surface contour in the Virtual Environment application. By using this 3D interface, we can easily select the seed point on the volume surface, with a visible 3D stylus. A 3D snap can be applied to the point of selection such that only the mesh point with maximum gradient magnitude would be selected. The size of snap can be defined by the user such that a 'better' edge point can be selected. Figure 6 shows the results. We apply our algorithm on a $128 \times 128 \times 64$ CT(Computed Tomography) data of a human head. The program runs on a Silicon Graphics Onyx Reality Engine and the rendering is done by 3D texture mapping [13]. In the Figure 6, we use our algorithm to cut away the nose and part of the face from the head. The contour of the nose is created by using three seed points selected by the user; *i.e.*, three segments are generated by the algorithm. In the second case, six control points are used. We can see the interior structure of the head after the corresponding part of the face is removed.

Conclusions

In summary, we have introduced a means to select a contour on a volume data surface in a virtual environment. The Intelligent Scissors technique provides an accurate and robust interactive contour detection.

We have also proposed a cutting surface estimation algorithm, by which a region of interest in the volume can be determined by one closed contour on the surface. Dynamic programming reduces the computational complexity.

Acknowledgements

The authors would like to thank for the many constructive comments of the reviewers. Compliments also go to Sau-Ling Chan and John Sum for proof-reading the drafts. This work is supported by RGC Earmarked Grant CUHK4162/97E of Hong Kong and CUHK Direct grant.

References

[1] George Celniker and Dave Gossard. Deformable curve & surface finite elements for free-form shape design. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 257–266, July 1991.

[2] Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1989.

[3] Tinsley A. Galyean and John F. Hughes. Sculpting: An interactive volumetric modeling technique. *Computer Graphics*, 25(4):267–274, July 1991.

[4] Anil K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

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

[6] L. Kobbelt. Interpolatory subdivision on open quadrilateral nets with arbitrary topology. In *Computer Graphics Forum (Proc. EUROGRAPHICS '96), 15(3)*, pages 409–420, 1996. Eurographics '96 issue.

[7] Eric N. Mortensen and William A. Barrett. Intelligent scissors for image composition. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 191–198. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[8] James V. Miller & David E. Breen & William E. Lorensen & Robert M. O'Bara and Michael J. Wozny. Geometrically deformed models: A method for extracting closed geometric models from volume data. In Thomas W. Sederberg, editor, *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 217–226, July 1991.

[9] Timothy Poston. The virtual workbench: a path to use of VR. In J D Irwin, editor, *The Industrial Electronics Handbook*, pages 1390–1393. CRC Press and IEEE Press, 1997.

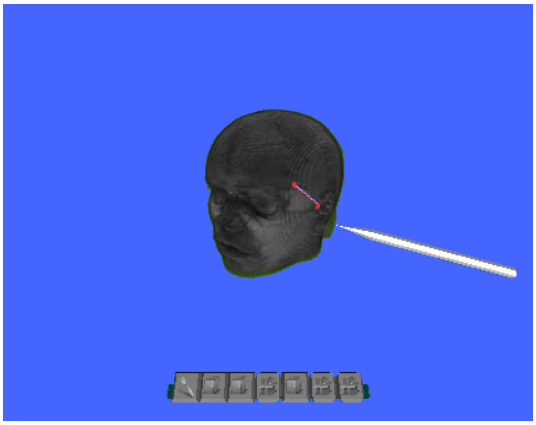
[10] Timothy Poston and Luis Serra. The virtual workbench: Dextrous VR. In *Virtual Reality Software and Technology (Proceedings of VRST'94, August 23-26, 1994, Singapore)*, pages 111–122, Singapore, August 1994. World Scientific Publishing.

[11] Luis Serra, Ng Hern, Chua Beng Choon, and Timothy Poston. Interactive vessel tracing in volume data. In Stephen N. Spencer, editor, *Symposium on Interactive 3D Graphics*, volume 25, pages 131–137, April 1997.

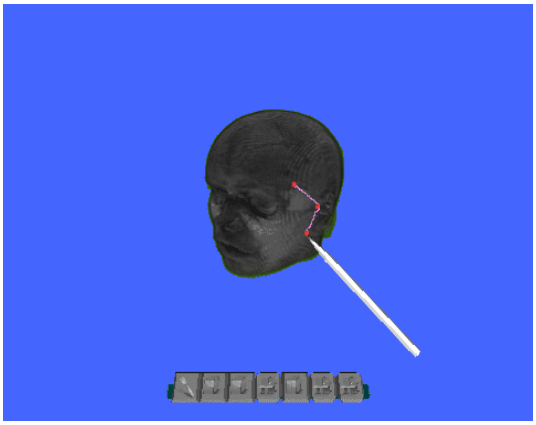
[12] Detlev Stalling and Hans-Christian Hege. Intelligent scissors for medical image segmentation. In B. Arnolds, H. Müller, D. Saupe, and T. Tolxdorff, editors, *Proceedings of 4th Freiburger Workshop* Digitale Bildverarbeitung in der Medizin, *Freiburg*, pages 32–36, March 1996.

[13] Allen Van Gelder and Kwansik Kim. Direct volume rendering with shading via three-dimensional textures. In *1996 Volume Visualization Symposium*, pages 23–30. IEEE, October 1996. ISBN 0-89791-741-3.

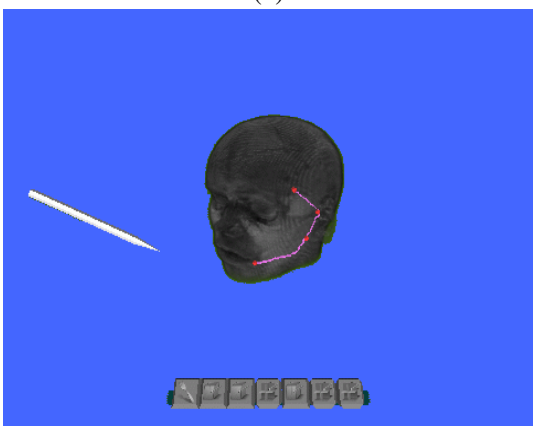
[14] Sidney W. Wang and Arie E. Kaufman. Volume sculpting. In Pat Hanrahan and Jim Winget, editors, *1995 Symposium on Interactive 3D Graphics*, pages 151–156. ACM SIGGRAPH, April 1995. ISBN 0-89791-736-7.



(a)

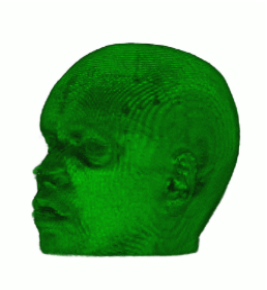


(b)

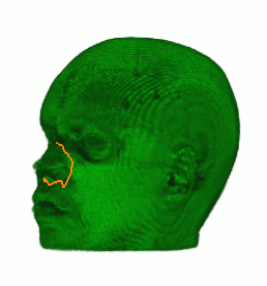


(c)

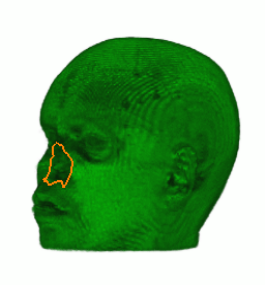
Figure 5: The above figures show how the contour are selected by our tool. (a) By using a 3D stylus, the user can select seed points on the volume surface (indicated by the red points). And then the segment (colored in pink) between two seed points will be estimated by the Intelligent Scissors techniques. (b),(c) The user draws the contour by selecting the seed points one by one.



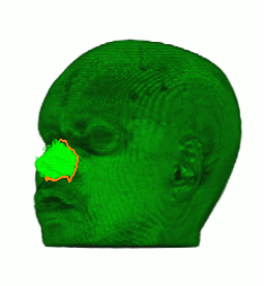
(a)



(b)

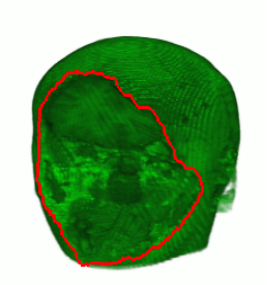


(c)



(d)

(e)



(f)

(g)

Figure 6: (a) The original volume. (b) A contour of the nose is defined by user. (c) The volume without the nose. (d) The nose is highlighted. (e) A closed contour of the front face. (f) The ROI cut away. (g) The volume of interest selected by the contour.