# FAST INTERACTIVE 2D AND 3D SEGMENTATION TOOLS

by

Kevin Chun-Ho Wong

Submitted to the faculty of the Graduate School
in partial fulfillment of the requirements
for the degree
Master of Philosophy
in the Department of Computer Science
The Chinese University of Hong Kong

August 31, 1998

Accepted by the Graduate Faculty, The Chinese University of Hong Kong, in partial fulfillment of the requirements of the degree of Master of Philosophy.

Prof. Pheng-Ann Heng
(Principal Adviser)

Prof. Hanqiu Sun

Prof. Irwin King

Prof. Andrew J. Hanson

August 31, 1998

*To my parents*

# 摘要

長久已來，圖像切割（Image Segmentation）都是電腦繪圖學和圖像處理學的研究熱點。研究人員設計了很多又快又準確的圖像切割技術。可是，這些技術至今仍未能滿足電腦繪圖工業的發展。今時今日，只是一張普通質素的數碼相片已佔用了數以百萬計的記憶位元，而數碼映片和體數據（Volume data）的容量便更加龐大了。傳統的圖像切割技術並不能有效處理這些龐大的圖像數據。因此，設計一些更快更準確的二維及三維的圖像切割技術是很重要的。

本論文主要提出兩項圖像切割技術的新貢獻，第一項是利用二元空間分割技術產生出來的圖來加快智能剪刀（Intelligent Scissors）的運作。主要的技巧是利用前置處理程序把圖像切割運算中所遇到的多餘數據除去。這個前置處理程序在最差情況下時間複雜度是 $O(n^{2/3})$，証明我們建議的方法是非常有效的。

第二項主要貢獻是我們應用二維圖像切割工具來解決三維體數據的切割問題。我們提出一方法利用虛擬真實環境在體數據的表面畫上線段，再利用一個切面評估演算法把適當的體數據取出。透過這工具，我們便能更容易創造更多的體數據模型了。

# Abstract

The research of Image segmentation has long been a difficult problem in the field of Computer Graphics and Image Processing. Many researchers have designed various image segmentation techniques to extract objects accurately and rapidly. However, these algorithms still cannot fulfill the requirement of rapidly developing graphics industries. Nowadays, a digital image with reasonable quality can frequently be very large. The storage requirements of digital video and volume data (3D image) are extremely large. Traditional segmentation algorithms are not efficient enough to handle these huge image data sets. Therefore, it is important to design faster and more accurate 2D and 3D segmentation techniques.

In this thesis, there are two major contributions in terms of 2D and 3D segmentation. One major contribution is that we propose a speedup version of an existing segmentation tool - *Intelligent Scissors*, which exploits a graph generated by the BSP (binary space partition) technique. The key is that a preprocessing scheme is used to filter out the redundant data in order to reduce the computation in the segmentation phase. The preprocessing scheme we propose is very efficient and its worst case time complexity is $O(n^{\frac{3}{2}})$, where $n$ is the number of nodes(pixels) in the graph.

A second significant contribution is that we extend the use of 2D interactive segmentation tools to solve the 3D volume data segmentation problem. We propose a freehand volume cutter to select a contour laying on an iso-surface contained in the volume data within a virtual environment. A cutting surface estimation algorithm is also proposed in order to determine the region of interest in the volume. By using this tool, one can naturally visualize many new volumetric data models.

# Acknowledgements

There are many who have helped me in the writing of this thesis. Prof. Heng Pheng Ann, my principal adviser, have been extremely generous in supporting the development of this thesis and my research project during the past two years. The patience, understanding, encouragement and wise counsel of Prof. Heng are greatly appreciated.

A draft of this thesis was read by Dr. Wong Tien Tsin, who has a special gift for rooting out problems and tenaciously insisting on improvements.

After the initial draft of this thesis was completed it was reviewed by Prof. Hanqiu Sun, Prof. Irwin King and Prof. Andrew J. Hanson. Their many comments and criticisms were of enormous value.

The following people also have made especially valuable suggestions: Mr. Cheung Chi Chiu (CUHK), Mr. Cheung Shing Kwong (CUHK), Mr. Chu Kam Wing (CUHK), Mr. Micheal Fung Ping Fu (CUHK), Mr. Ng Chi Wai (CUHK), Mr. Tommy Siu Yu Heng (CUHK), Mr. Lam Wing Kai (CUHK), Dr. Sum Pui Fai (CUHK) and Mr. Yeun Kin Yan (Poly. HK).

Finally, I want to thank my father, Wong Kwok Wah; my mother, Bonnie Yip Pui Heng; and my aunt, Yip Lai Heng for their encouragement.

# Contents

# List of Figures

# Chapter 1

# Introduction

In computer graphics and image processing, various contour detection techniques have been developed and many researchers have proved their feasibility. Two useful techniques are Active Contours (also called Snakes) [19] and Intelligent Scissors [24] [33].

Active contours proposed by Kass *et al* [19] are deformable polyline models used to locate object boundaries in an image. The contour is first placed near the desired edge manually, then this contour will gradually move to the edge driven by an energy minimization process. The continuity, smoothness and fitness of the contour is supervised by an energy equation. Although, the snake can outline the object's boundary and provide a good result in a noise free image, its initialization-sensitive nature leads to an unstable performance when it is applied to a noisy image.

The Intelligent Scissors [24] proposed by Mortensen and Barrett is another technique that can accurately obtain the object's boundary in the image. However, a certain extent of user intervention is required. The kernel of this algorithm is a shortest path searching procedure. The image is treated as a graph and the pixels are treated as the nodes. In this algorithm, the two end-points of a contour are defined by the user and the resultant contour is computed by finding the shortest path between the given two end points.

In our research, two interesting problems are concerned. One is how to segment the desired object from the given image of natural scene efficiently, accurately and in

a robust manner. Another is how to extract a volume of interest from an existing 3D volume data correctly. Some of our ideas used to solve these problems are inspired by the recent contour detection techniques mentioned above.

At the first stage of our research, we had an indepth review on the current image segmentation techniques. In Chapter 2 and 3, these prior works will be discussed in detail. In Chapter 2, we will have a brief introduction of the image segmentation problem and some of the essential techniques. Chapter 3 introduces the Active Contour Model (Snakes) and its extensions, which has developed rapidly and are broadly applied in various areas in recent years.

At the second stage of our research, we developed a fast interactive image segmentation tool called *fast intelligent scissors*. In Chapter 4 and 5, we introduce this tool and discuss how to extract the object from the given image of natural scene efficiently and accurately. In order to speed up the current interactive segmentation tool, a slimmed graph is proposed to filter out the redundant data. We will describe the generation of the slimmed graph in Chapter 4. In Chapter 5, we discuss how to speed up the Intelligent Scissor Algorithm with the slimmed graph technique.

At the final stage of our research, we extend the idea of 2D intelligent scissor to solve the problem of 3D volume editing. In Chapter 6, we suggest a freehand volume cutter to select a contour on a volume data surface within a virtual environment. The Intelligent Scissors technique we developed provides an accurate and robust tool for interactive contour detection. A cutting surface estimation algorithm is also proposed, by this, a region of interest in the volume can be determined by one closed contour on the surface. We have exploited various algorithms in order to reduce its computational complexity.

# Chapter 2

# Prior Work : Image Segmentation Techniques

One of the vital tasks of our research is to study various image segmentation approaches. In this chapter, a series of segmentation techniques will be discussed. Firstly, we will discuss the image segmentation algorithm which have been developed in early stage such as Region Based Segmentation, Boundary Based Segmentation, Watershed Based Segmentation and Canny Edge Filter with contour following. Secondly, we will discuss some recent approaches such as Fuzzy C-means Segmentation, Pyramid based Fast Curve Extraction and Curve Extraction with Multi-resolution Fourier Transformation. Then, we will introduce some interactive segmentation techniques such as Intelligent Scissors and Magic Wands.

Active Contour Model (Snakes) has been developed rapidly in recent years and there are a great number of publications about this techniques. In Chapter 3, we will discuss this model and its extensions in more detail.

## 2.1 Introduction to Image Segmentation

Image Segmentation is a process of grouping homogeneous pixels in an image. This process can outline the boundary curves of an desired object in a scene and estimates the spline functions of these curves.

In recent few decades, Image segmentation algorithms have been broadly applied in various areas of application such as medical image visualization [34][32][27] (Segmentation of the computed tomography (CT) image or magnetic resonance image (MRI)), motion tracking [11][43] in the movie, image editing [24][12] such as cutting and pasting, volume or 3D data [39][35] segmentation and shape modeling [9][37].

The human visual system has an impressive natural talent to extract an identified object from the image quickly. However, this is a rather difficult task for the computer to handle. The main difficulties include the following points:

- The photo taken from the real world is always degraded due to noise, low contrast and occlusions.
- The image stored in the computer memory is discrete, disconnected and lossy.
- The size of image data involved in the computation is usually large and this leads to a slow performance. In general, the image segmentation is seldom a real time process.
- The image segmentation is difficult to be performed in a fully automatic manner. If no one gives enough information, the computer does not know how detail the segmentation should be. For example, if a human face photo is segmented, computer may consider the eyes as the desired objects, but user may want to extract the whole face.

In the fields of Image Processing and Computer Graphics, various image segmentation and contour detection methodologies have been developed as well as many

researchers have proved their feasibility. In the following sections, these techniques will be discussed in detail.

## 2.2 Region Based Segmentation

### 2.2.1 Boundary Based vs Region Based

In general, most of image segmentation techniques can be classified into two classes, *boundary based approaches* and *region based approaches*. Boundary based approaches segment objects on the basis of their profiles and edge map. Some well known examples are Snakes [20], Contour Following [10][18] and Hough Transform [40]. The other class of approaches is the region based segmentation. In region based analysis, the image will be subdivided into many smaller regions. In each sub-regions, the local feature is sufficiently defined. Then, the object in the image will be extracted by identifying various regions in an image which have similar features.

In some particular problems, the hybrid or mixed methods such as Region-based method with edge detection as well as Watershed based method are designed in order to achieve better segmentation performance.

In this section, we will introduce some region based segmentation techniques as well as its variants.

### 2.2.2 Region growing

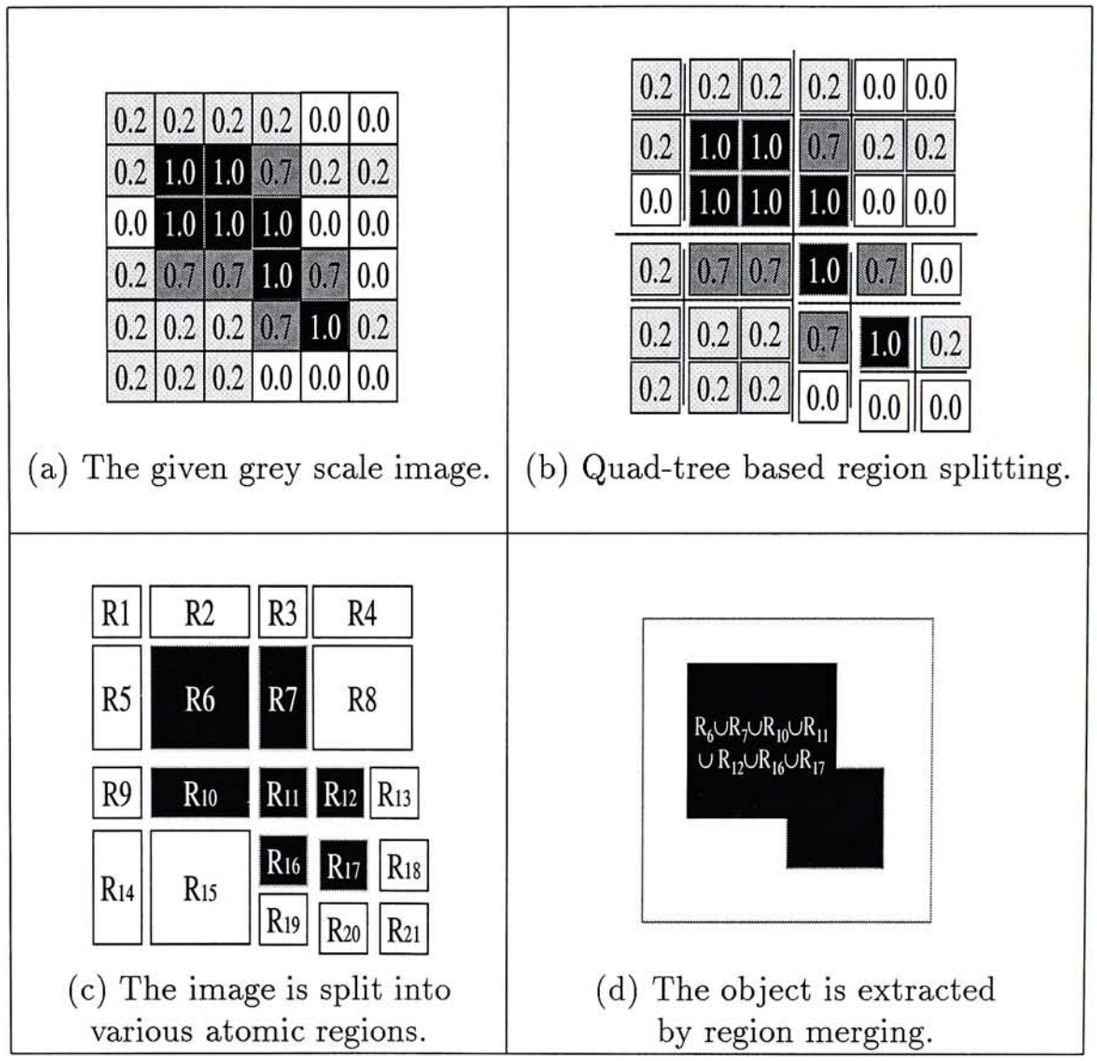
One approach of the region based segmentation techniques is *Region Growing* [7]. There are two key steps in this approach: *region splitting* and *region merging* . In the step of region splitting, the given image will be divided into several atomic regions of pixels with constant grey levels. In the step of region merging, similar adjoined regions will be merged until the adjoined regions become sufficiently distinguishable.

Figure 1(a), 1(b) and 1(c) show how the image is split into atomic regions. The grey value of the pixels is in the range between 0.0 to 1.0. In this example, the quad tree based region splitting is used. If the particular subimage is not a homogeneous region, it will be divided into four parts evenly. This process will go on recursively. By quadtree based region splitting, the atomic regions can be obtained efficiently due to the constant time computation of the position of the split lines. Figure 1(d) shows the step of region merging. Two adjoined regions will be merged if they have similar feature (average grey values). After the region merging, the desired object or different spatial regions in the image can be segmented out.

Comparing with the boundary-based one, region based approach has a larger error tolerance. Nevertheless, the time complexity of the region based algorithms is often quite large. Also, region based approach always produces a over-segmented results.

### 2.2.3 Integrating Region Based and Edge Detection

If a natural scene is treated by region growing algorithm, over segmented results are often obtained. To relieve this problem, *Pavlidis and Liow* [26] proposed a method integrating region growing and edge detection. In the first step, they use region growing to produce a over segmented results. In the second step, the boundaries in the over segmented picture will be cancelled. Then, they attempt to modify the shape of the boundaries to improve the result. They use edge detection and the proposed cost function to control the shape and the strength of the region boundary. The algorithm also penalizes the long straight artifact boundaries. They modified the shape of the object boundary by maximizing the cost function and this process is similar to the energy minimization in the active contour model [20].




(a) The given grey scale image.

(b) Quad-tree based region splitting.

(c) The image is split into various atomic regions.

(d) The object is extracted by region merging.

Figure 1: Explanations of Region Growing algorithm.

### 2.2.4 Watershed Based Methods

The key idea of the watershed algorithm [38][17] is to consider the strength of each edge pixel as an altitude. Hence, the image becomes a 3D surface with the homogeneous regions appearing as basins. Then, the watershed lines will be constructed by flooding this 3D surface. When the floods from adjoined region meet, dams are constructed to prevent merging. After the immersion is completed, the boundary of the segmented regions can be constructed by the dams.

## 2.3 Fuzzy Set Theory in Segmentation

### 2.3.1 Fuzzy Geometry Concept

In many image processing applications, digital geometry often plays a vital role in the calculation of geometrical measures in the image such as area, perimeter, diameter of the objects. However, in some problems of computer vision such as pattern recognition and image segmentation, the object boundaries appear ill-defined and non-crisp in digital geometry. It is much better to consider the geometrical measurements as fuzzy set.

To apply fuzzy set theory into image processing, we should use the membership function to measure the image. Let us use an example to show how to estimate the area under fuzzy geometry. Firstly, we define the membership function $\mu_p(\lambda)$ for the pixels with grey value $p$ as follows:

$$\mu_p(\lambda) = \begin{cases} 1 & p \geq \lambda \\ 0 & p < \lambda \end{cases} \tag{1}$$

The pixel $p$ is active when its grey value is larger than the threshold. By this definition, the object in the image will have different values when different threshold

value are given. (as shown in Figure 2). To describe its area under fuzzy geometry, we use the membership function $A(\lambda)$, where

$$A(\lambda) = \sum_{\forall pixel} \mu_p(\lambda) \tag{2}$$

In the example shown in Figure 2 , $A(\lambda)$ is calculated as follow:

$$A(\lambda) = \begin{cases} 36 & \lambda \leq 0 \\ 27 & 0 < \lambda \leq 0.2 \\ 12 & 0.2 < \lambda \leq 0.7 \\ 7 & 0.7 < \lambda \leq 1.0 \\ 0 & \lambda > 1.0 \end{cases} \tag{3}$$

In next section, we will discuss a general approach in fuzzy segmentation called fuzzy C-means clustering.

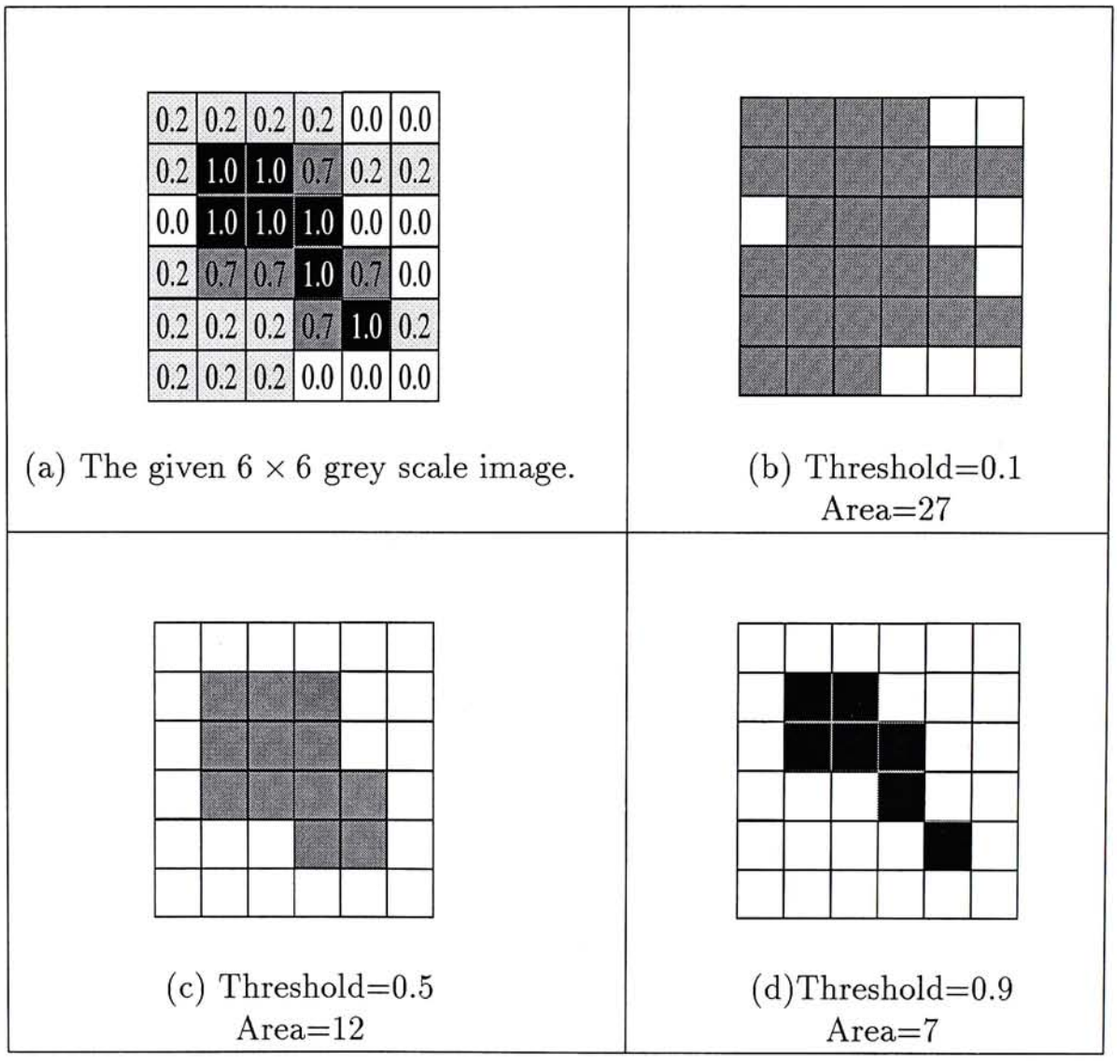
### 2.3.2 Fuzzy C-Means (FCM) Clustering

The segmentation algorithm with fuzzy C-means clustering [22] included two main parts. The first part is to analyze the histogram in order to partition the histogram into various classes of intervals. Then, a great amount of pixels in the image can be grouped according to the partitioned color space. The second part is to classify the remained unclassified pixels by Fuzzy C-means clustering.

The overall segmentation algorithm can be summarized in the following steps:

1. For each color component (R,G,B) of the image, build a histogram.

2. For each histogram, find out their most predominant peaks or valleys by scale-space analysis.




(a) The given 6 × 6 grey scale image.

(b) Threshold=0.1
Area=27

(c) Threshold=0.5
Area=12

(d)Threshold=0.9
Area=7

Figure 2: Area measurement under fuzzy digital geometry.

3. Build an interval tree of zero crossing in second derivative for each histogram so that the x axis along the histogram will be segmented into different classes.

4. Combine the R,G,B histograms to build the segmented color space.

5. If each color component lies within the maxima interval, that pixel is considered classified.

6. Any unclassified pixel will be classified by the fuzzy c-Means techniques.

The fuzzy c-Means technique attempts to cluster a pixel by finding the local minima of the sum of squared error. A pixel is assigned to the closest classified class of which the fuzzy membership has a maximum value.

There are two disadvantages in the FCM segmentation. Firstly, this method always manages with regions which have vague boundaries and shapes. Secondly, FCM segmentation is a time consuming process due to great amount of unclassified pixels.

## 2.4 Canny edge filter with contour following

Canny edge filter is a well-known technique that finds the edge pixels by convolution. It can be implemented easily and its computation is fast when parallel programming is applied. However, the polylines of the boundaries cannot be directly obtained. To find the control points of the boundaries, a further contour following approach [10][18] should be exploited. The boundary is tracked by performing the 8-connectivity test for each edge pixels. A polyline segment will be assigned for each pair of adjoined edge pixels. One of the disadvantages is the curve generated by this technique is not smooth enough, since there are only 8 possible directions of the polyline boundary : 0°,45°,90°,135°,180°,225°,270° and 315°

## 2.5 Pyramid based Fast Curve Extraction

Besides producing a non-smooth curve, contour following method always generates a fragmented resulted curve even though there exists only a little noise along the object boundary. To lessen the noise effect, we can analyze the image in various resolution, Based on this idea, Connelly [5] designed a simple curve extraction process called Fast Curve Extraction.

The curve extraction is performed in the pyramid data structure which is a stack of arrays representing the image at decreasing resolution. A given level of stack has twice the resolution in each dimension as the upper level. To speed up the image analysis, this algorithm is designed as a parallel process. The overall algorithm can be summarized in the following steps:

1. To generate different resolution images from the original one, we can recursively apply the image operations such as sub-sampling or smoothing.

2. We can apply the thinning operator in the edge map of the image. And then, we connect the neighbor points with local maxima to create the trend curve (similar to contour following).

3. For each level of resolution of the image, we retrieve the trend contours in parallel.

4. Combine the trend curves of all level of images to produce the resulted main curve. We can set the weighting values for each level of curves. These weights can be determined in terms of background, homogeneity or the subjective judgements of users.

## 2.6 Curve Extraction with Multi-Resolution Fourier transformation

*Calway* [2] implements another multi-resolution Curve Extraction method called *Curve Extraction with multi-resolution Fourier transformation* .

The key idea of this algorithm is to convert the image into a set of straight line segments which vary in size and orientation. As a result, the boundaries in the image can be clearly outlined by these segments. Of course, we can combine these segment into curves by using simple curvature measurement method.

This method is performed by a fine-to-coarse region subdivision scheme. After the complete subdivision, the image will be converted into a set of square blocks vary in size. Each block contains three local featured parameters to model a straight line. They are scale index (size of the block), orientation (direction of the line model) and offset (offset position of the line). The estimation of these parameters can be achieved by Fourier transformation.

During the subdivision, if a block contains the feature cannot be modeled by a straight line, the block will be subdivided into four sub-parts evenly. This test will be performed recursively until every block contains a simple local feature (a straight line or a homogeneous region).

Since no initialization procedure is involved in this method, the detection is more stable and initialization invariant. However, this analysis is mainly depended on the rectangular grids, so the curve detected is not very smooth.

## 2.7 User interfaces for Image Segmentation

In this part, two famous interactive image segmentation will be discussed. They are intelligent Scissors and Magic Wands.

### 2.7.1 Intelligent Scissors

The Intelligent Scissors [24] is an interactive technique by which the object's boundary in the image can be obtained in an accurate and robust manner. The main kernel of this algorithm is a shortest path searching procedure. The image is considered as a graph and the pixels are considered as nodes. In this algorithm, the two end-points of the boundary contour is defined by the user and the resulted contour is computed by finding the shortest path between the points.

The main kernel of Intelligent Scissors is a global optimal path searching algorithm. It is implemented using the Dijkstra's algorithm [6]. The definition of local costs along the edge depend on the 2D gradient at the pixel positions. In [33], the strength indicator $G(x, y)$ of the pixel at $(x, y)$ is formulated as the following equation:

$$G(x, y) = |\nabla I(x, y)| \tag{4}$$

where $\nabla$ is the gradient operator and the pixel at $(x, y)$ is preferable if its $G(x, y)$ is large enough. Thus the local cost between pixels at $(x_1, y_1)$and $(x_2, y_2)$ can be formulated as

$$Cost((x_1, y_1), (x_2, y_2)) = (max_{\forall x,y} G(x, y)) - \frac{1}{2}(G(x_1, y_1) + G(x_2, y_2)) \tag{5}$$

The Dijkstra algorithm shown in Algorithm 1 is basically a 2D dynamic program which finds the suboptimal paths from all pixels to the seed point. In practical application, the search need not be completed, and it can be stopped whenever a path reaches the position selected by the user (goal point). If the seed point or the cost values on the edges is changed, the search must be executed again.

**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**

Algorithm 1: Dijkstra's algorithm for 2D shortest path searching.

### 2.7.2 Magic Wands

This tool has appeared in several famous image processing application such as Photo-Shop and GIMP for the purpose of the object segmentation. By using Magic Wand, the homogeneous region in the image can be specified through a mouse click to select the seed point.

Magic Wand uses the color similarity to estimate the object boundary. Thus, this tool is not suitable for processing in noisy image. If we can first use the intelligent scissors to outline the partial boundary of an object, and then, use the Magic Wand to estimate the resulted boundary; the accuracy of the resulted outline can be greatly improved.

# Chapter 3

# Prior Work : Active Contours Model (Snakes)

Snakes or active contours proposed by Terzopoulos, Kass and Witkin [20] are deformable contours (a polyline with a set of control points) used to locate object boundaries in an image. The contour is firstly placed near the desired edge by some initialization process, then this contour will gradually move to the edge driven by the external image force. Many previous experiments by other researchers [14] [24] [35] [3] have shown that snake and its variants are powerful tools in segmenting complex contours shape in a noisy image.

## 3.1 Introduction to Active Contour Model

In the paper of *Kass et al* [20], the traditional snake is defined as a curve $v(s) = [x(s), y(s)]$, where the normalized arc-length parameter $s \in [0, 1]$ . The movement of the snake is influenced by minimizing the energy function $E_{snake}$ as

$$E_{snake} = \int_0^1 E_{int}(v(s)) + E_{ext}(v(s))ds$$

where

$$E_{int} = w_1|v'(s)|^2 + w_2|v''(s)|^2$$

$$E_{ext} = -|\nabla(G_\sigma(x, y) * I(x, y))|^2 \quad (6)$$

In Equation 6, $E_{int}$ represents the internal energy of the contour due to bending or discontinuities. $w_1$ and $w_2$ are weighting parameters controlling the elasticity and rigidity of the contour. The external energy $E_{ext}$ is influenced by the image force which is a function of image information such as edges, lines and terminations. A gray level image is denoted by $I(x, y)$ as well as a 2D Gaussian function with the standard deviation $\sigma$ denoted by $G(x, y)$ is used to blur out the noise in the given image information.

The main kernel of active contour model implementation is to find out the shape of contour which contains minimum internal and external energy. There are four steps in the energy minimization algorithm:

- Consider the image domain as a continuous plane and set up a variational integral (energy function).
- Derive the Euler equation from the energy functions

- Discretize the Euler equation by converting the derivates into finite differences.
- Solve the discrete equation iteratively until it converges.

Theoretically, the aim of the above four steps is to minimize a mathematical function. Physically, the contour on the image will move their control points to the positions with lower energy field. Hence, the object on the image can be extracted by this deforming contour.

There are four vital advantages of the original active contour model:

- Certain extend of error tolerance
- Boundary continuity preservation
- Flexibility on the boundary smoothness control
- Good performance in automatic segmentation

Due to these advantages, the snake model can be broadly applied in various applications; especially in segmentation of magnetic resonance images [31] and motion tracking [13].

## 3.2 Variants and Extensions of Snakes

In the following sections, we will introduce some of the significant developments of the active contours model. These techniques included balloons [4], dual active contours [15], gradient vector flow (GVF) snakes [30][44] and active contours with dynamic programming [1][41]. Balloons, dual active contours and GVF snakes aim to achieve a stronger error tolerance in the object segmentation as well as better contour initialization. Dynamic programming aims to improve the stability and efficiency in the energy minimization process. The motivations and main features of these techniques will be discussed.

### 3.2.1 Balloons

#### Aims and Motivations

In 1991, *Cohen* [4] proposed a new active contour model called **Balloons**. When this model is applied to a 2D image, its contour will inflate gradually and being attracted to the high gradient pixels. By its inflating feature, balloon model can achieve a result that is much less sensitive to the initial conditions comparing to the traditional active contour model.

#### Main idea

In general, the active contours model may shrink its boundary shape into a point due to the minimization of the internal energy term $E_{int}$. To avoid this shrinking property, an internal pressure pushing the contour outward is involved. *Cohen* has suggested this pressure force as $F_{pressure}$

$$F_{pressure} = k_1 \mathbf{n}(\mathbf{s}) \tag{7}$$

where $\mathbf{n}(\mathbf{s})$ is outward pointing unit normal vector to the contour at point $v(s)$, $k_1$ is the magnitude of this force which is usually positive. The internal pressure force prevent the curve from shrinking into a point and being trapped by isolated edge points caused by noise.

#### Advantages and Drawback

By the using the pressure force on the curve, balloon model enables the user to achieve the promising results even though the given initial guess of the curve is far from the desired solution. However, if the image force is not strong enough, the curve will not be attracted by the edge point due to the additional pressure. Hence, the user should select the parameters of the image force and internal pressure in the energy equation carefully.

### 3.2.2 Robust Dual Active Contour

#### Aims and Motivations

Bad choices of an initial contour and other parameters will cause an incorrect segmentation results. In order to increase the robustness of the active contour model and relieve the problem of sensitive initialization, *Gunn* and *Nixon* [15] propose a dual active contour algorithm which contains both contracting and inflating contours.

#### Main idea

In dual active contour model, there are one contour contracting from outside and one contour inflating from inside of the desired boundary. These contours will refine their shapes under the minimization process of the scale invariant internal energy function which retains the continuity and smoothness constraints. When two contours become stationary, the contour with the higher energy will move to the other contour with an adaptive driving force. This process will go on until these two contours reach equilibrium.

#### Advantages and Drawback

By the dual active contour, the sensitivity to initialization can be greatly reduced. Comparing the two contours during energy minimization, the weak minima can be rejected easily and the more accuracy results can be obtained in a robust manner. On the other hand, by the use of the adaptive driving force, the contour model can extract different degrees of minima (potential resulted curves) without the need to modify its parameters.

Since two initial contours (contracting and inflating) are used, the dual active contour works ambiguously when some special topological problems are treated. For examples, using the dual contours model to detect the open boundary will results in a closed loop. Moreover, the wrong selection of an inner contour will lead to a false result.

### 3.2.3 Gradient Vector Flow Snakes

#### Aims and Motivations

In [30], *Xu* and *Prince* indicate two important difficulties in the design and implementation of active contour model. First, the initial contour given by the users must, in general, be close to the true boundary otherwise it will likely converge to an undesired result. Second, active contours have poor convergence to boundary concavities. To relieve these problems, a new active contour model with Gradient Vector Flow is proposed.

#### Main idea

Gradient Vector Flow (GVF) is a new type of external force for active contour model. This flow is computed over the image domain as a diffusion of the gradient vectors of a grey-level edge map. A GVF field $V(x, y)$ is defined as the equilibrium solution of the following system of partial differential equations

$$\frac{\partial V(x,y,t)}{\partial t} = \mu_{smooth}\nabla^2 V - (V - \nabla f)|\nabla f|^2 \tag{8}$$

where $\frac{\partial V(x,y,t)}{\partial t}$ denotes the partial derivative of $V(x, y, t)$ with respect to time $t$. The coefficient $\mu_{smooth}$ governs the level of smoothing of the field near the boundaries. $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is the Laplacian operator. $f$ is the edge map derived from the grey level image $I(x, y)$ and its values can be computed by conventional edge detector.

In the original energy minimization process, the following Euler equation derived from Equation 6 must be satisfied.

$$w_1 v''(s) - w_2 v''''(s) - \nabla E_{ext} = 0 \tag{9}$$

To apply GVF in the active contour model, we only need to replace the terms $-\nabla E_{ext}$ by the GVF field $\frac{\partial V(x,y,t)}{\partial t}$ shown in Equation 8. Then the Euler equation will become

$$w_1 v''(s) - w_2 v''''(s) - \mu_{smooth}\nabla^2 V + (V - \nabla f)|\nabla f|^2 = 0 \tag{10}$$

To find the solution of the curve $v(s)$, we just consider it as a function of time $v(s,t)$ and solve the following partial differential equation:

$$\frac{\partial}{\partial t}v(s,t) = w_1 \frac{\partial^2}{\partial s^2}v(s,t) - w_2 \frac{\partial^4}{\partial s^4}v(s,t) - \mu_{smooth}\nabla^2 V + (V - \nabla f)|\nabla f|^2 \tag{11}$$

Refer to [20], this equation can be discretized and solved iteratively until it converges using some numerical methods or dynamic programming approaches mentioned in [1].

**Features**

GVF snake works especially well when it is applied to fit the boundary with high concavity. Its extension, generalized GVF snakes [44] , also improves its convergence into long, thin boundary indentations and shows the superior performance on medical images segmentations.

### 3.2.4 Energy Minimization using Dynamic Programming with presence of hard constraints

**Aims and Motivations**

In the energy minimization of active contour, the following problems always appear:

- Since there is no hard constraint for the interdistance of the points on the contours, it is not surprised to see points along the contour clustering into a dense structure at certain places.

- In the original minimization, there is a need to calculate the high -order derivatives with the discrete image information. Errors often occur and may lead the contours to have unstable behavior.

To solve these problems and improve the performance of the algorithm, Amini and *et al.*[1] present a stable energy minimization algorithm based on dynamic programming with the presence of hard constraints.

**Main idea**

The proposed dynamic programming is accomplished by filling the 3D array $E_t(i,j,k)$ iteratively. The computation of the array values is based on the following equations which is derived from the original active contours energy function (Equation 6).

$$\begin{aligned} E_t(i+1,j,k) &= min_{0\leq m\leq N}\{E_t(i,k,m)\} + E_{ext}(v_i \oplus k) \\ &+ \tfrac{1}{2}(w_{1,i}|v_i \oplus k - v_{i-1} \oplus m|^2) \\ &+ w_{2,i}|v_{i+1} \oplus j - 2v_i \oplus k + v_{i-1} \oplus m| \end{aligned} \tag{12}$$

where

$v_i$ : the positions of the control points of the active contours in $i^{th}$ iteration.

$\oplus$ : the operator for $v_i$, $v_i \oplus k$ indicates the $k^{th}$ neighbor choice (possible moves) of $v_i$.

$N$ : the number of possible moves at each iteration and $0 \leq j, m, k \leq N$.

$E_{ext}$ : the external energy term.

$w_1$, $w_2$ : weighting parameters controlling the elasticity and rigidity of the contour.

As $i$ increases, the values of $E_t$ will tend to the minimum energy value. On the other hand, to avoid the control points of the contour being placed too close, some

hard constraints can be used to control the lower bound of their interdistances. In each iteration, if there does not exist a minimum satisfying the constraint, then the algorithm will terminate.

## 3.3 Conclusions

In these two chapters (chapter 2 and 3), we have discussed various image segmentation techniques. The purpose of developing these techniques is to improve the performance, accurate and robustness of the object extraction. From the review, we can easily notice that extracting the object interactively can provide the most accurate segmentation result. However, most of these traditional techniques are not fast enough to be applied in interactive tools since they always require exhaustive computation. One of the bottleneck is that these techniques always treat the image as a huge 2D array and analyze the image pixel by pixel. In order to relieve this problem, we propose a new data structure called Slimmed Graph to represent an image instead of 2D array. Fast segmentation can be achieved when the boundary computation is performed on this slimmed data structure. In the following chapter, we describe this new data structure in detail.

# Chapter 4

# Slimmed Graph

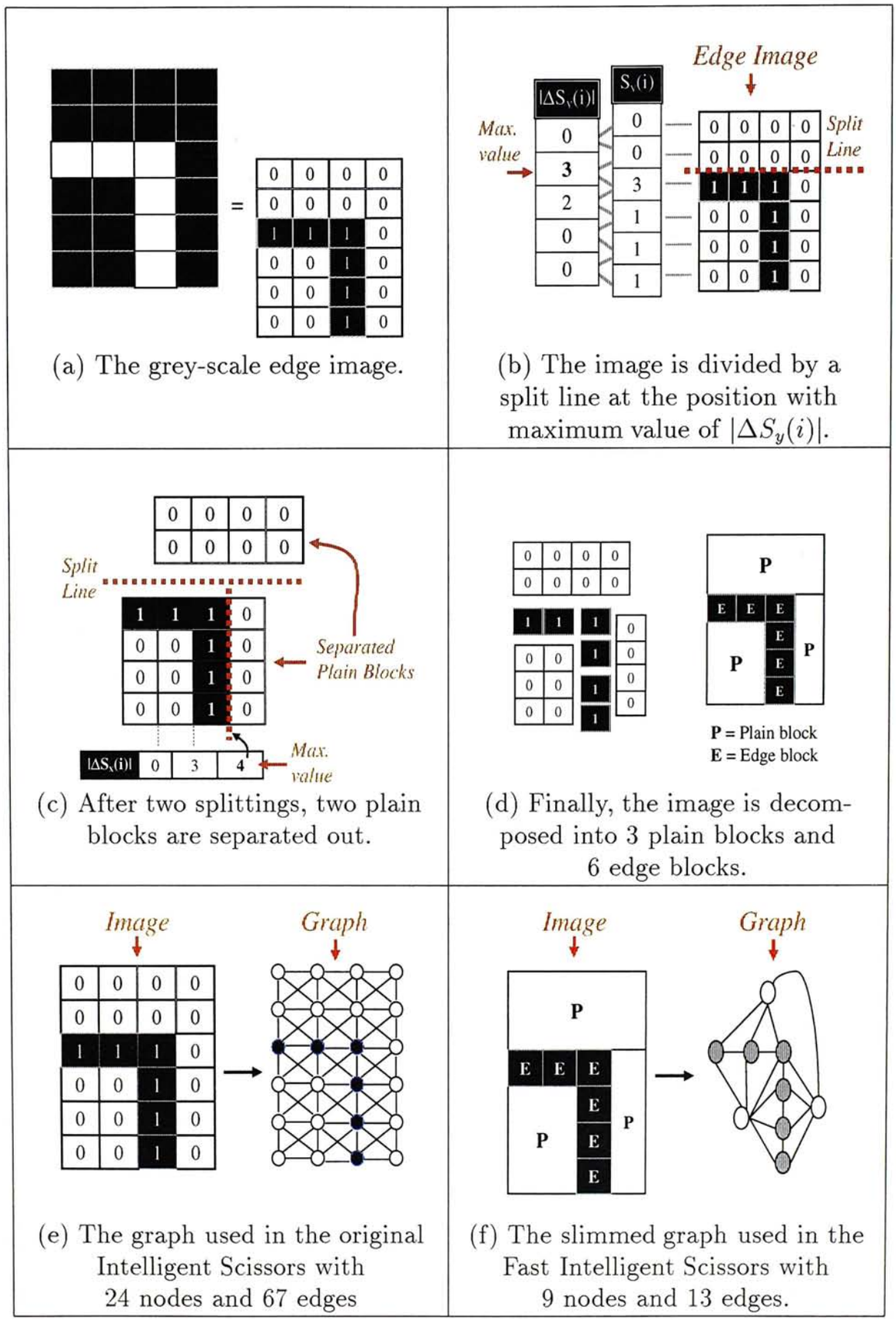
Slimmed Graph provides a new data structure to represent the profile of the image. By this, the computation complexity involved in the image analysis can be greatly reduced. In this chapter, a slimmed graph generation algorithm with worst case time complexity of $O(n^{\frac{3}{2}})$ is introduced.

In the slimmed graph, each node represents a sub-region of the image. The set of nodes representing the adjoined plain regions will be grouped, and more nodes are needed to represent an edge region (a region with high gradient).

The node distribution can be achieved by the BSP (Binary Space Partition) mentioned in Section 4.1. In this process, a certain amount of split lines will be used to separate the image into blocks. With the split lines selection scheme proposed in Section 4.2 and 4.3, more tiny blocks will be generated near the region with higher gradient. Another problem in the slimmed graph generation is how to add the edges between two neighbor nodes and assigning the cost value to the edge. Section 4.4 discusses how to determine whether two nodes are neighbors. In section 4.5, the overall slimmed graph generation algorithm is shown.

## 4.1 BSP-based image analysis

Using BSP-based (Binary Space Partition) image analysis, we separate the image into many smaller sub-regions by vertical and horizontal split lines. Eventually, a block diagram (a diagram of segmented regions) can be generated (Figure 3a-d). The aim of the splitting process is to find out the plain block and the edge block of the image. Plain block is the rectangular region with less gradient, e.g. a region containing same pixel values. An edge block is a region with high image gradient, e.g. a region containing sharp edges. Then, a slimmed graph can be created (Figure 3-f) by associating each block with a node and connecting each pair of neighbor blocks with an edge.




(a) The grey-scale edge image.

(b) The image is divided by a split line at the position with maximum value of $|\Delta S_y(i)|$.

(c) After two splittings, two plain blocks are separated out.

(d) Finally, the image is decomposed into 3 plain blocks and 6 edge blocks.

(e) The graph used in the original Intelligent Scissors with 24 nodes and 67 edges

(f) The slimmed graph used in the Fast Intelligent Scissors with 9 nodes and 13 edges.

Figure 3: A simple example of slimmed graph generation.

## 4.2 Split Line Selection

Function $G(x, y)$ is the pixel gradient at pixel $(x, y)$. The value of $G(x, y)$ is between 0.0 to 1.0. $M$ and $N$ are the width and height of the interested sub-region respectively. $S_x$ and $S_y$ are the sum of the grey values along the vertical and horizontal lines respectively.

$$S_x(i) = \sum_{j=0}^{N-1} G(i, j) \tag{13}$$

$$S_y(i) = \sum_{j=0}^{M-1} G(j, i) \tag{14}$$

The position of the split line is determined by the following criteria.

If $M > N$, the vertical split line from $(x_s, 0)$ to $(x_s, N)$ is selected.

where

$$x_s = argmax_i(|\Delta S_x(i)|) + 1 \tag{15}$$

$argmax_i(f(i))$ returns the argument $i$ if $\forall j, f(i) \geq f(j)$.

Otherwise, a horizontal split line from $(0, y_s)$ to $(M, y_s)$ is selected.

where

$$y_s = argmax_i(|\Delta S_y(i)|) + 1 \tag{16}$$

$x_s$ is between 1 and $M - 1$ while $y_s$ is between 1 and $N - 1$. Figure 3-b shows how the split line is chosen by this criteria.

## 4.3 Split Line Selection with Summed Area Table

A faster computation of $\Delta S_x(i)$ and $\Delta S_y(i)$ can be achieved by the summed area table techniques. Let $A(x, y)$ be the precomputed summed area function where

$$A(x,y) = \begin{cases} \sum_{i=0}^{x}\sum_{j=0}^{y} G(i,j) & 0 \le x < W,\ 0 \le y < H \\ 0 & otherwise \end{cases} \tag{17}$$

Suppose that the sub-region $D$ is considered. The top-left and bottom-right corner positions of $D$ are $(left_D, top_D)$ and $(right_D, bottom_D)$. Then,

$$S_x(i) = A(i, bottom_D) + A(i-1, top_D - 1) - A(i-1, bottom_D) - A(i, top_D - 1) \tag{18}$$

$$S_y(i) = A(right_D, i) + A(left_D - 1, i - 1) - A(right_D, i - 1) - A(left_D - 1, i) \tag{19}$$

And

$$\begin{aligned} \Delta S_x(i) = \quad & A(i+1, bottom_D) + 2A(i, top_D - 1) + A(i-1, bottom_D) \\ & -A(i+1, top_D - 1) - 2A(i, bottom_D) - A(i-1, top_D - 1) \end{aligned} \tag{20}$$

$$\begin{aligned} \Delta S_y(i) = \quad & A(right_D, i+1) + 2A(left_D - 1, i) + A(right_D, i-1) \\ & -A(left_D - 1, i+1) - 2A(right_D, i) - A(left_D - 1, i-1) \end{aligned} \tag{21}$$

As the values of $A(x,y)$ have been stored in a 2D array, the computation of $A(x,y)$ can be performed in constant time. Hence, the functions $\Delta S_x(i)$ and $\Delta S_y(i)$ can also be computed in constant time.

## 4.4 Neighbor blocks

The slimmed graph is then generated by associating each block with a node. For each pair of neighbor blocks, an edge is added to connect them. Any two blocks with shared boundary are considered as neighbor. Let $(x_{i,left}, y_{i,top})$ and $(x_{i,right}, y_{i,bottom})$ be the corner positions of block $i$. Whether or not block $i$ and block $j$ are neighbor is determined by the following criteria.

$$x_{min} = min(x_{i,left}, x_{j,left})$$

$$x_{max} = max(x_{i,right}, x_{j,right})$$

$$y_{min} = min(y_{i,top}, y_{j,top})$$

$$y_{max} = max(y_{i,bottom}, y_{j,bottom}) \tag{22}$$

**If**

$$x_{max} - x_{min} \leq (x_{i,right} - x_{i,left}) + (x_{j,right} - x_{j,left}) \tag{23}$$

**and**

$$y_{max} - y_{min} \leq (y_{i,bottom} - y_{i,top}) + (y_{j,bottom} - y_{j,top}) \tag{24}$$

**then** the block $i$ and block $j$ are neighborhood.

## 4.5 Slimmed Graph Generation

Our slimmed graph generation algorithm is outlined in Algorithm 3. In this graph construction procedure, if all pairs of blocks are considered in the neighborhood checking, the time complexity is $O(n^2)$, where $n$ is the number of nodes in the slimmed graph. Nevertheless, if the edges are added during the block decomposition, by our graph generation algorithm, the worst case time complexity is $O(n^{\frac{3}{2}})$.

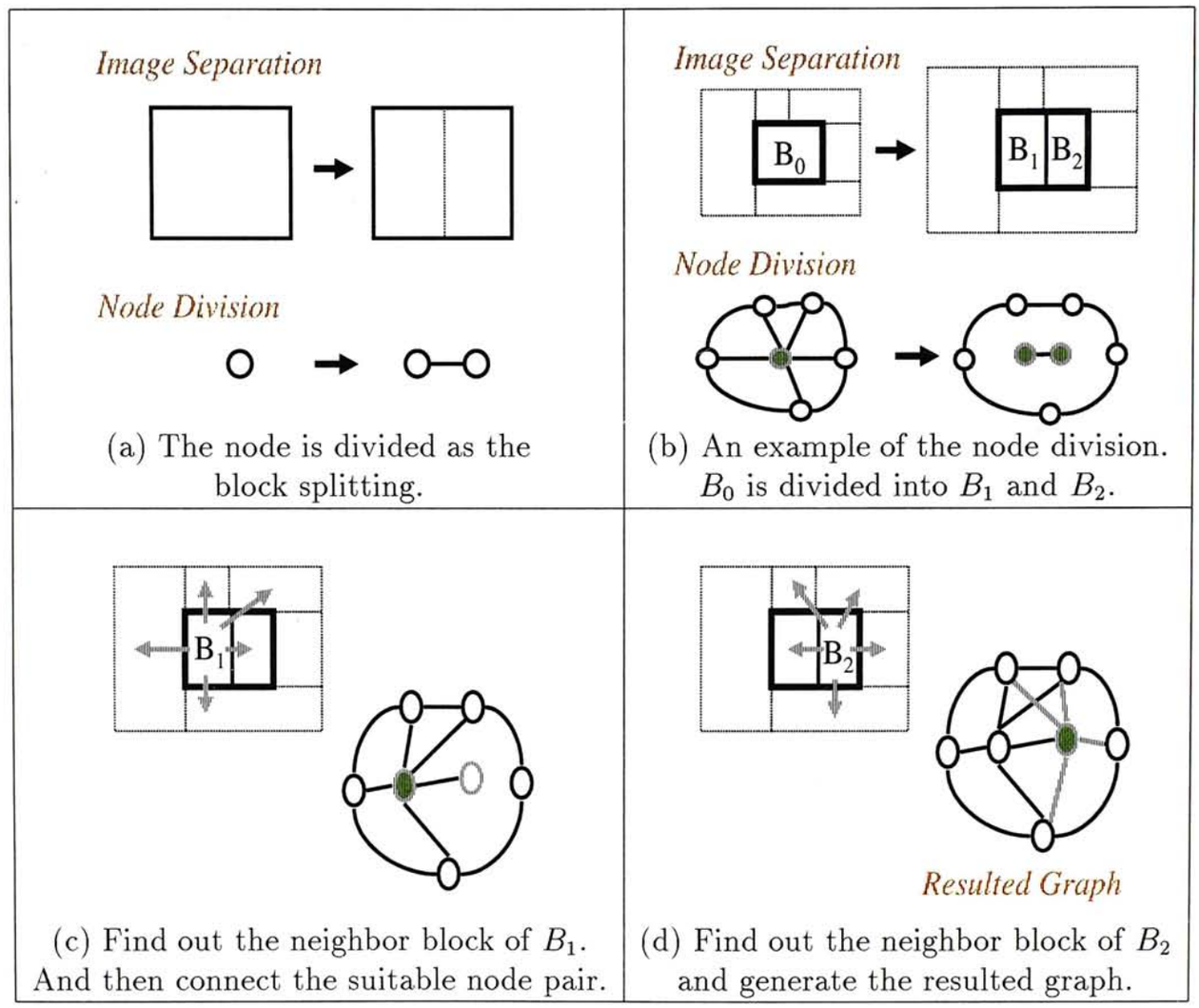
There are two threshold values $\lambda$ and $\theta$ used in the block decomposition. Threshold $\lambda$ is the maximum area of each edge block. Threshold $\theta$ is the maximum on the total sum of gradient of a plain block. The nature of a block can be classified by the algorithm shown in Algorithm 2. The size of the slimmed graph can be controlled by $\lambda$ and $\theta$. When the value of $\lambda$ or $\theta$ increases, a slimmer graph can be obtained. However, the fitness and accuracy of the boundary line will be reduced in this case.

In our algorithm, a new node is generated in each splitting. The Figure 4-a and 4-b illustrate how a node is converted into two connected nodes during the block splitting. The Figure 4-c and 4-d illustrate how to complete the node division by adding the edges from its neighbors to these new nodes.

▷There are three possible types for the given block B : **plain**, **edge** and **unclassified**.

**Algorithm** *Block-Classification(***Block** B*)*
  **If** ($\forall$ grey value in B $< \theta$)
    B is a **plain** block.
  **elseif** (the area of B $< \lambda$)
    B is an **edge** block.
  **else**
    B is **unclassified** and
    will have a further subdivision.

Algorithm 2: the algorithm of Block Classification.




(a) The node is divided as the block splitting.

(b) An example of the node division. $B_0$ is divided into $B_1$ and $B_2$.

(c) Find out the neighbor block of $B_1$. And then connect the suitable node pair.

(d) Find out the neighbor block of $B_2$ and generate the resulted graph.

Figure 4: An example of node division while splitting the blocks.

**Notation**

$\mathcal{G}$ is the graph.
$\mathcal{T}$ is a list of blocks to be classified.
$N_r(n)$ is a list of adjoined nodes of node $n$.
$B(n)$ is the corresponding block of node $n$.
$n(B)$ is the corresponding node of block $B$.
$B_0$ is the initial block representing the whole image.

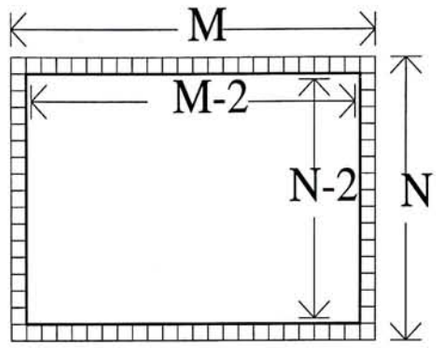
**Algorithm** *Slimmed-Graph-Generation*

```
Append T with B0.
Add n(B0) to graph G.
while (T≠empty)
  ▷ Pop b from T
  if b is neither plain nor edge then   // Check with Algorithm 2.
     Pop b from T.
     Split block b into b1, b2.      // Use the split line selection method
                                     // mentioned in Section 4.2 and 4.3.
     n_r ←N_r(n(b))
     Append T with b1, b2.
     Remove n(b) from graph G.
     Add n(b1), n(b2) to graph G.
     Connect n(b1), n(b2) with an edge.
     for all node p in node list n_r
       if B(p) and b1 are neighbor then      // Use the neighborhood
         Connect p, n(b1) with an edge.      // checking method mentioned
       end if                                // in Section 4.4 .
       if B(p) and b2 are neighbor then
         Connect p, n(b2) with an edge.
       end if
     end for
   end if
 end while
```

**Time Complexity**

**Append, Add, Pop, Remove, Connect, if...then** : $O(1)$.
**Split** : $O(R)$. where $R = max(W, H)$,
$W$ and $H$ are the width and height of the given image.
**for, while** : depend on number of iterations.

Algorithm 3: the algorithm of Slimmed Graph Generation.




In this case, the $(M-2) \times (N-2)$ block is surrounded by $2(M+N-2)$ pieces of $1 \times 1$ tiny blocks. Its neighborhood number reachs maximum.

Figure 5: A block with maximum neighborhoods.

## 4.6 Time Complexity

The time complexity of our slimmed graph generation algorithm in the worst case is $O(n^{\frac{3}{2}})$. Referring to the Algorithm 3, we know that the time complexity of this algorithm is $O((c+R)n)$ where $c$ is the maximum number of the neighborhood among the blocks, $n$ is the number of nodes in the graph, $W$ and $H$ are the image width and height respectively and $R$ is equal to $max(W, H)$. In the worst case (Figure 5), a $(M-2) \times (N-2)$ block adjoins with many $1 \times 1$ tiny blocks. In this case, $c$ will equal to $2(M-2+N-2)+4$.

Since $n < W \times H$ ($W$ and $H$ are the image width and height), the upper bound of $n$ is $R^2$. Hence, the time complexity of our slimmed graph generation algorithm in the worst case is

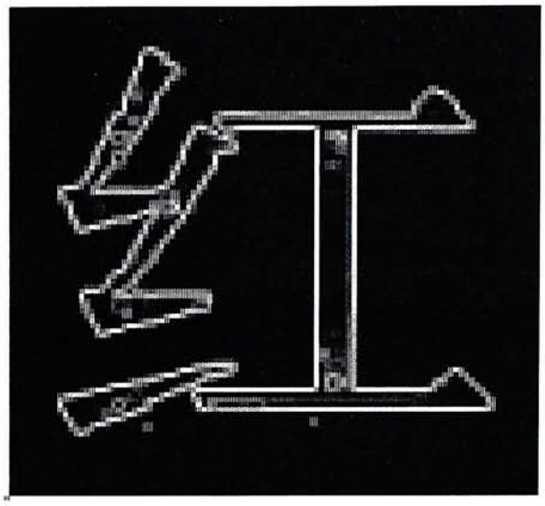
$$\begin{aligned} O((c+R)n) &= O(\{[2(M+N-4)+4]+R\} \times R^2) \\ &= O(R^3) \\ &= O(n^{\frac{3}{2}}) \end{aligned}$$

## 4.7 Results and Conclusions

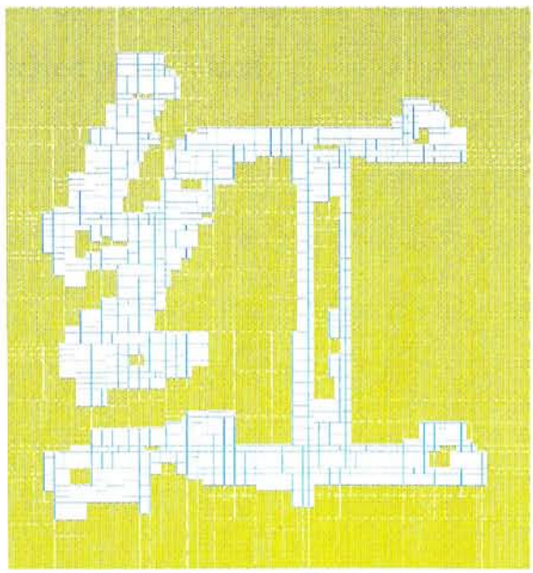
Figure 6 shows a simple example of slimmed graph generation of a chinese character image. This image contains 10000 pixels. For the slimmed graph, only 1043 nodes and 3619 edges are used to represent it.

In summary, a slimmed graph generation algorithm with time complexity $O(n^{\frac{3}{2}})$ is proposed. This technique can be exploited in various real-time image processing tools to improve their performance. In next chapter, we will attempt to extend our idea on the Intelligent Scissors.

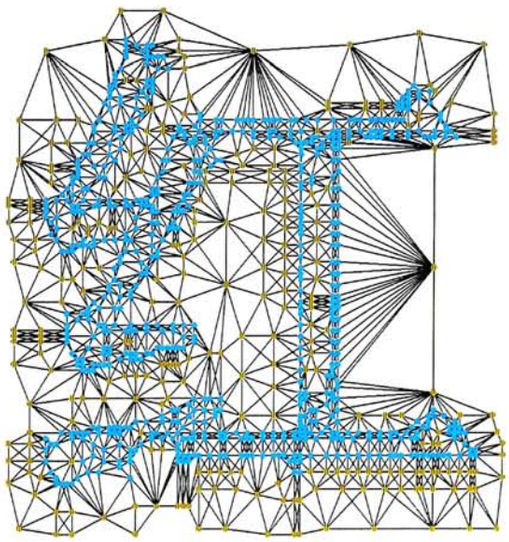
(a) A 100×100 image
of a chinese character 'Red'.



The grey-scale edge image.



(c) The BSP block analysis.
The blue block is edge block
and the orange one is plain.



(d) The slimmed graph.

Figure 6: A slimmed graph generation example of a chinese character image.

# Chapter 5

# Fast Intelligent Scissor

Using Intelligent Scissors [24], user can accurately extract an object from the digitized image using simple intuitive gesture. However, this tool suffers from slow performance when large images are treated. Moreover, selecting a new seed point requires a noticeable amount of time in path searching. By observation, the pixels within the non-edge regions are rarely involved in the boundary computation. If these nodes can be removed before extracting the boundary, the performance of the intelligent scissors can be sped up. In this part, we introduce a technique which utilizes the mentioned observation.

## 5.1 Background

Accurate object extraction from the digitized image has long been an open problem in the field of Computer Graphics and Computer User Interface. *Mortensen and Barrett* [24] introduced an interactive tool known as Intelligent Scissors to segment the desired object out of a given image. *Stalling and Hege* [33] further extended the use of this tool for the medical image segmentation.

The kernel of Intelligent Scissors is the Dijkstra [6] shortest path searching algorithm. After the seed point and the goal point are selected by the user, a boundary line between these two points can be obtained by computing the global optimal path. In the graph representation of the image, each pixel is mapped to a node connecting with its 8 neighbor pixels (nodes) by the edge. As the example shown in Figure 3(e), an image of $W \times H$ pixels contains $W \times H$ nodes and $4WH - 3(W + H) + 2$ edges.

## 5.2 Motivation of Fast Intelligent Scissors

In the original Intelligent Scissors algorithm, all potential shortest paths have to be recomputed whenever a new seed point is chosen. This computation is performed using a dynamic programming which has the time complexity of $O(n)$, where $n$ is the number of nodes(pixels) in the graph. Hence, the performance of Intelligent Scissors is highly depended on the size of the graph. By observation, there is a great portion of nodes within the non-edge regions which are rarely involved in the boundary computation. If these nodes are pruned before running the algorithm, the performance should be significantly improved. For a real time interactive application, its speed is critical. Fast intelligent scissors provide the user a fluent control as well as an immediate preview while dragging the mouse to select the goal point. Moreover, if the tool can show the resulting image boundary right after altering the

parameter, the user can have a comfortable fine tuning of the boundary so as to obtain a more accurate segmentation. A faster cost recomputation for the fine tuning can be achieved by reducing the number of nodes and edges in the graph. Our motivation is to reduce the size of the graph in order to achieve real time interactive feedback.

For more information about the original idea of Intelligent Scissors, please refer to Section 2.7.1 in Chapter 2. Furthermore, the detail of computation of the Slimmed Graph have been discussed in the previous chapter. In this chapter, We will give the discussion of how to speed up the Intelligent Scissors by applying the Slimmed Graph in the shortest path searching. The implementation and results of fast intelligent scissor is shown in Section 5.4. Conclusions are drawn and future directions are discussed in Section 5.5.

## 5.3 Main idea of Fast Intelligent Scissors

The main idea of our fast Intelligent Scissors is to perform the boundary searching on a slimmed graph instead of the original complete image graph. In this application, we put the slimmed graph generation as a preprocessing scheme. When the image is completely loaded, the slimmed graph will be computed immediately.

When the user selects a seed point $s$, the block enclosing this point $B(s)$ will be located. The corresponding node $n(B(s))$ will be used as the starting node. After the user has selected the goal point $g$, the boundary path between the nodes $n(B(s))$ and $n(B(g))$ would be calculated by the 2D dynamic programming algorithm shown in Algorithm 1. In our application, this algorithm works with the slimmed graph proposed in the previous section. Whenever the parameters or the costs have been altered, the costs in all edges and the boundary path would be recomputed.

### 5.3.1 Node position and Cost function

A 2D position $(x_{n_k}, y_{n_k})$ is assigned to each node in the slimmed graph to represent the group of pixels within the block $B_k$. Position $(x_{n_k}, y_{n_k})$ is found by the weighted average functions shown in Equations 25 and 26.

$$x_{n_k} = \frac{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} G(i,j)i}{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} G(i,j)} \tag{25}$$

$$y_{n_k} = \frac{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} G(i,j)j}{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} G(i,j)} \tag{26}$$

For each block, $l(n_k)$ is the average grey value of the edge block.

$$l(n_k) = \frac{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1} G(i,j)}{MN} \tag{27}$$

Since the topology of the slimmed graph is no longer a grid in which each pair of neighbor nodes has the same inter-distance, we need to include the effect of distance between the nodes into the local cost function calculation. Hence, the cost function of the edge between two nodes can be defined as the following equation:

$$Cost(n_1, n_2) = \alpha\sqrt{(x_{n_1} - x_{n_2})^2 + (y_{n_1} - y_{n_2})^2} + \beta[1.0 - l(n_1)][1.0 - l(n_2)] \tag{28}$$

In the original Intelligent Scissors, Dijkstra algorithm is applied to search for the shortest path. In order to keep the Dijkstra algorithm work well, the cost value along each edge must be positive value. The cost function (Equation 28) is designed under this consideration.

The two parameters $\alpha$ and $\beta$ are the positive weights to control the smoothness and fitness of the boundary line respectively. In practical application, the new resultant

| | | Slimmed Graph Thresholds | | The graph in Original I.S. | | Slimmed graph in Fast I.S. | | % of size reduced | |
|---|---|---|---|---|---|---|---|---|---|
| Figure | Size(pixels) | $\theta$ | $\lambda$ | # of node | # of edge | # of node | # of edge | node | edge |
| 6 | 100×100 | 0.08 | 5 | 10000 | 39402 | 1043 | 3619 | 89.57 | 90.82 |
| 7 | 488×335 | 0.08 | 5 | 163480 | 651449 | 24582 | 84216 | 84.96 | 87.07 |
| 8 | 320×240 | 0.08 | 5 | 76800 | 305522 | 8134 | 27391 | 89.41 | 91.03 |
| 9 | 512×512 | 0.08 | 7 | 262144 | 1045506 | 17825 | 59848 | 93.20 | 94.28 |
| 10 | 100×100 | 0.02 | 5 | 10000 | 39402 | 1343 | 4566 | 86.57 | 88.41 |
| 11 | 416×600 | 0.12 | 7 | 249600 | 995394 | 18370 | 60514 | 92.64 | 93.92 |
| 12 | 600×415 | 0.12 | 7 | 249000 | 992957 | 16505 | 54705 | 93.37 | 94.49 |

Table 1: This table shows the size of the slimmed graph in various fast intelligent scissors testings.

boundary should be displayed right after the user changes these parameter values.

## 5.4 Implementation and Results

In our implementation, LEDA and ImageVision libraries are used. LEDA [25] is a C++ library of data types and algorithms of combinatorial computing. It provides some useful classes and methods such as graph and Dijkstra algorithm for our implementation. We have also used the classes from the ImageVision library [8] to perform some image processing operations.

Our algorithm have been executed under the Silicon Graphics Onyx equipped with Reality Engine. Different types of images such as digitized image captured from video camera, medical image and synthetic image are tested. Table 1 summarizes the size of the slimmed graph in these experiments. The statistics in the table indicate our slimmed graph generation algorithm can reduce the size of graphs significantly.

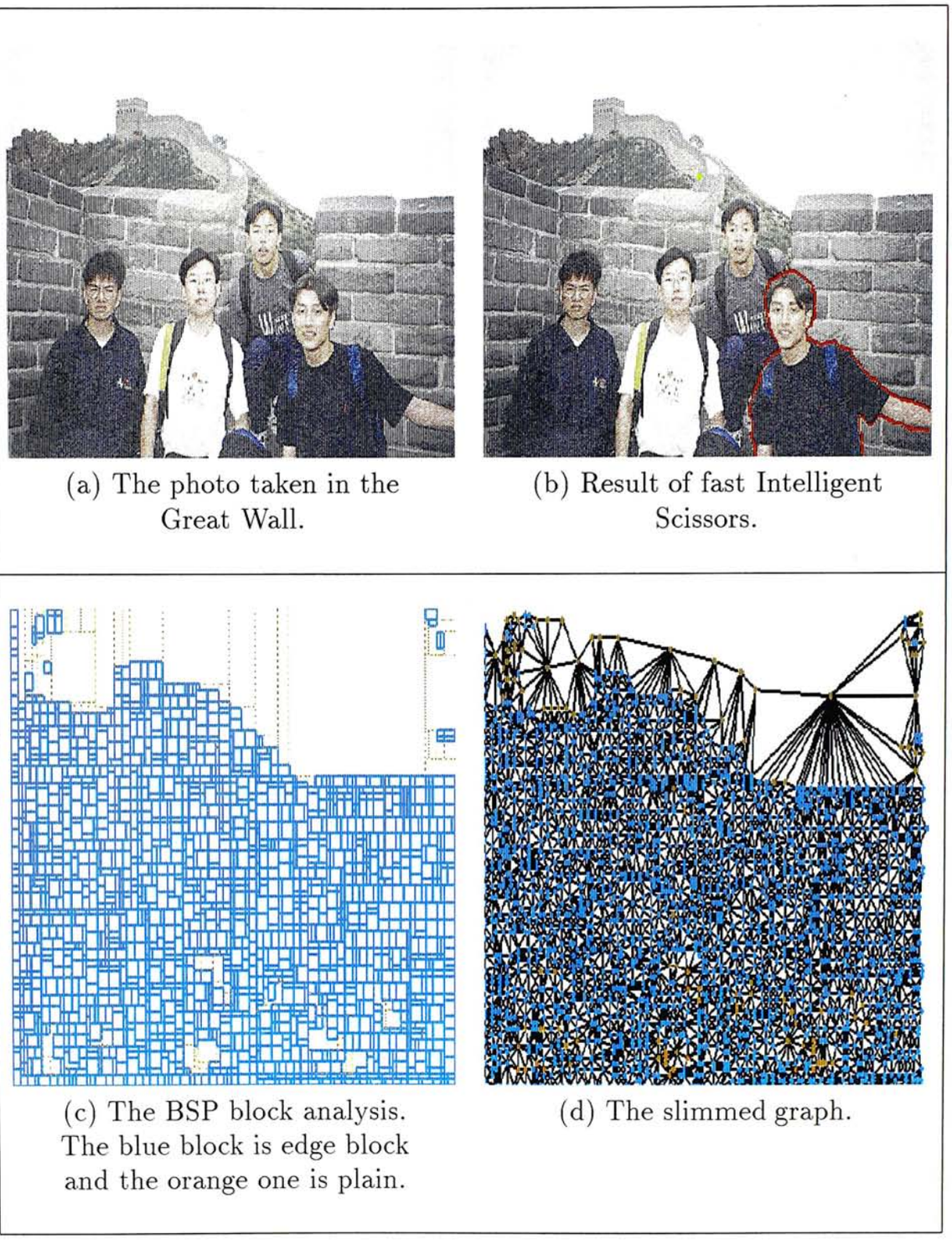
Figure 7 demonstrates how we use the fast Intelligent Scissors to extract the object from the photo taken in the Great Wall. The results of blocks decomposition and the slimmed graph are shown in Figure 7-c and 7-d. Figure 8 shows the result of the fast Intelligent Scissors applied to a low quality image captured from video camera. Figure

9 demonstrates that the fast intelligent scissors can work well on medical image. In Figure 10, we segment the cube out from the synthetic image. Figure 11 and 12 illustrate two more photos taken in Korea. The desired regions in these images can be outlined accurately.

## 5.5 Conclusions

In summary, we precompute a slimmed graph to speed up the Intelligent Scissors algorithm. By our fast Intelligent Scissors, user can segment the desired object from the given image accurately and efficiently .

Since the cost recomputation has been sped up by reducing the edge and node quantity in the graph, the user can interactively fine tune the boundary shape by altering the parameters such as $\alpha$ and $\beta$. With a real time fine tuning of these parameters, a more accurate result can be obtained.



(a) The photo taken in the Great Wall.

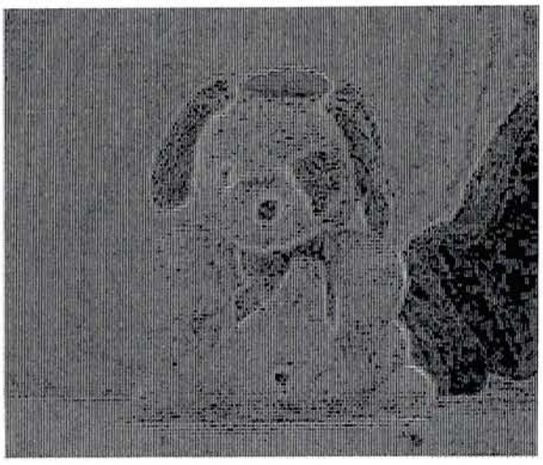
(b) Result of fast Intelligent Scissors.

(c) The BSP block analysis. The blue block is edge block and the orange one is plain.

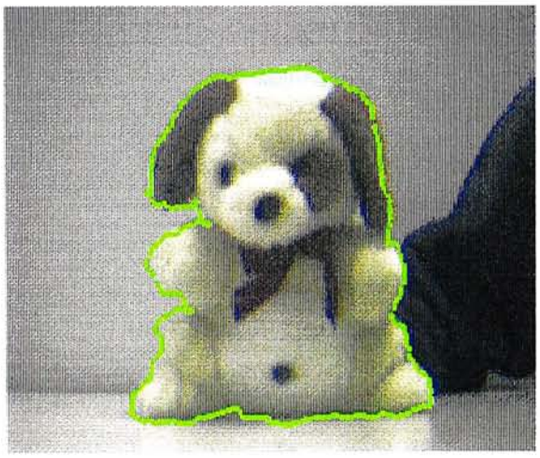
(d) The slimmed graph.

Figure 7: An application of the Fast Intelligent Scissors on a photo taken in China.

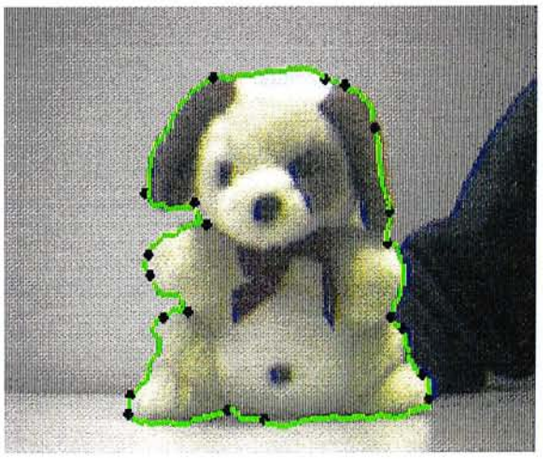
(a) The image captured from a video camera.



(b) The grey-scale edge image (The quality of this edge image is poor.)

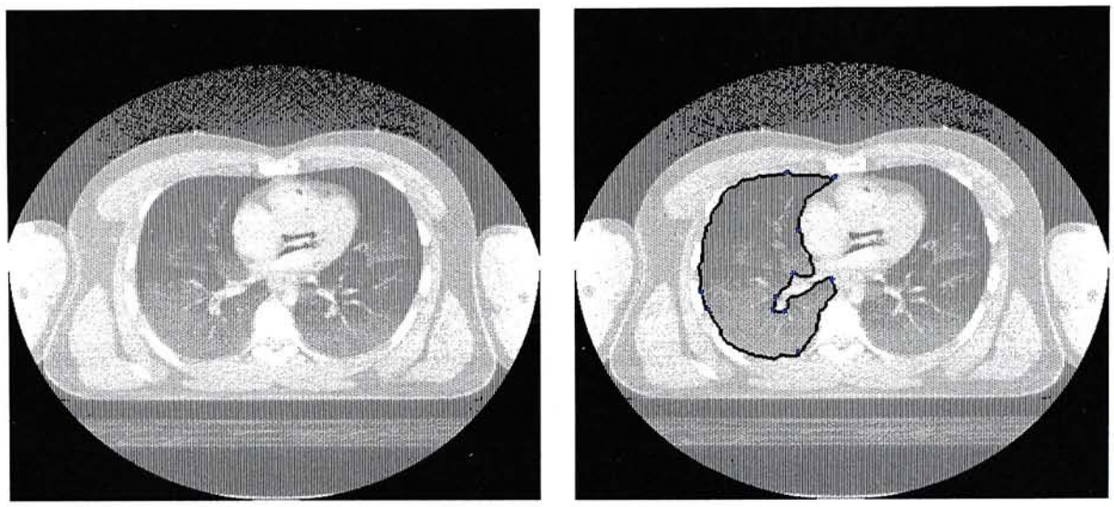


(c) Result of fast intelligent scissors.



(d) 20 seed points are needed to segment out the doll.

Figure 8: An application of the Fast Intelligent Scissors on an image captured from a video camera.

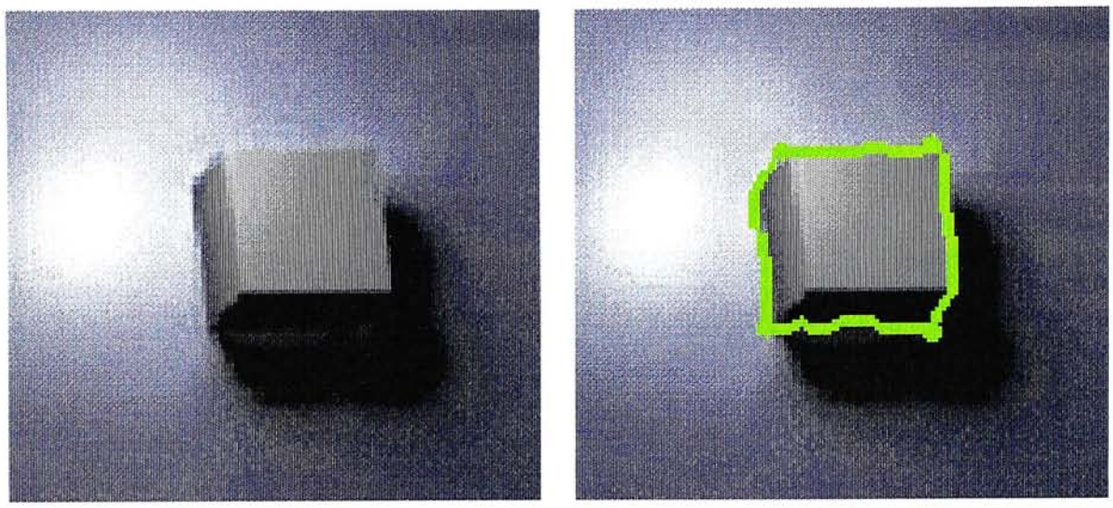


(a) (b)

(a) The left figure is a 512 × 512 CT image which shows the cross-section of a male chest.

(b) In the right image, the left lung has been segmented out by fast Intelligent Scissors. 11 seed points are used to outline the boundary.

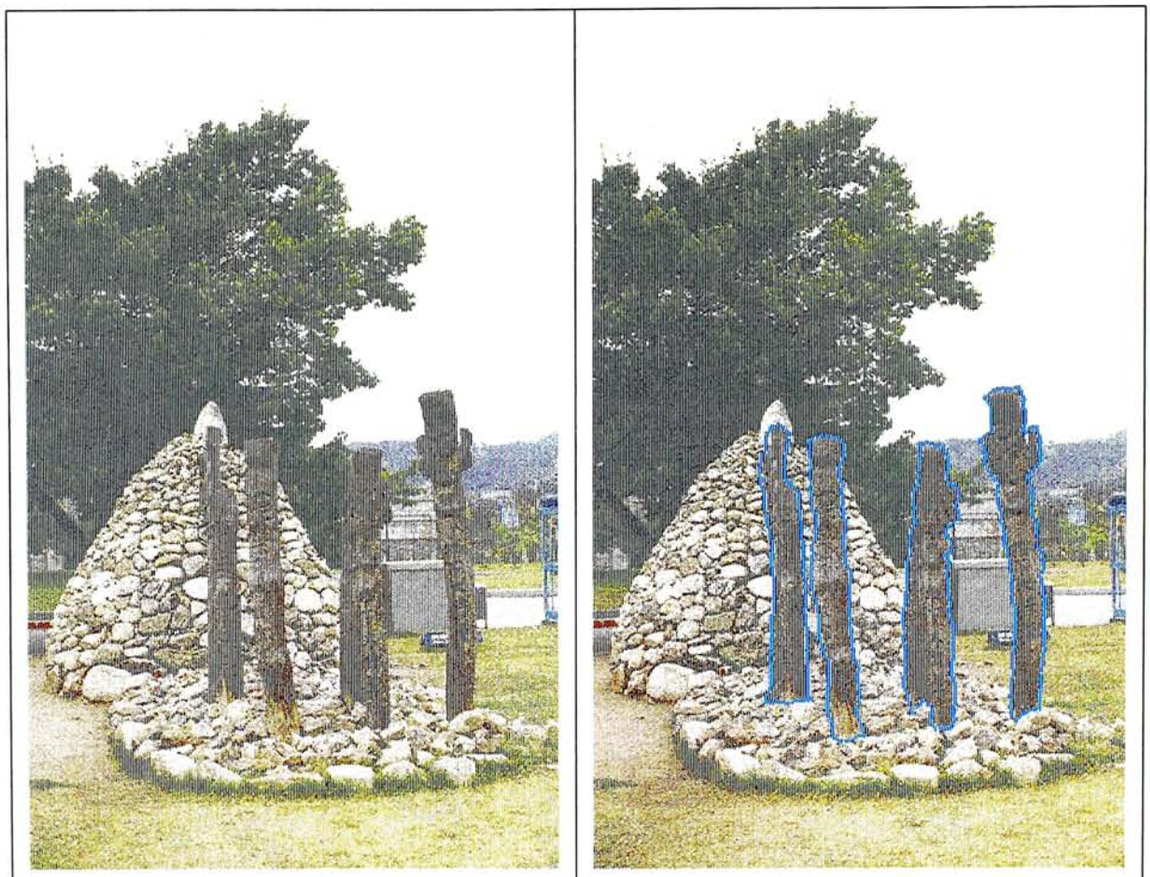
Figure 9: An application of the Fast Intelligent Scissors on the medical image.



(a) (b)

(a) The left figure shows a synthetic image.
(b) The cube is segmented out by fast Intelligent Scissors.

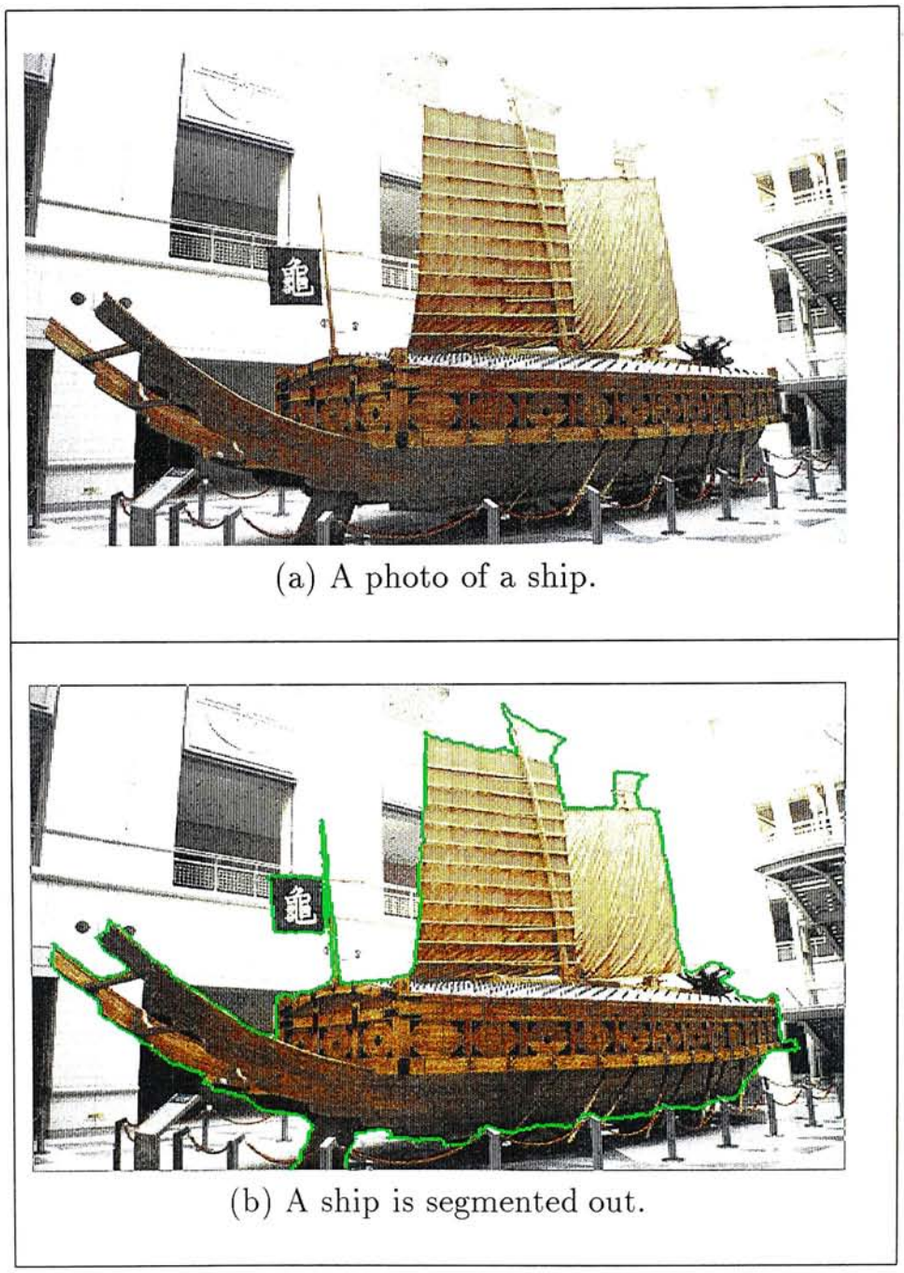
Figure 10: An application of the Fast Intelligent Scissors on the Computer Generated image.



(a) A photo taken in Korea.

(b) The four wooden rods is segmented out by fast Intelligent Scissors.

Figure 11: An application of the Fast Intelligent Scissors on the natural scene taken in Korea.



(a) A photo of a ship.

(b) A ship is segmented out.

Figure 12: An application of the Fast Intelligent Scissors for segmenting the ship in the natural scene.

# Chapter 6

# 3D Contour Detection: Volume Cutting

Intelligent Scissors is an efficient interactive tool for 2D 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 chapter, 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 on the volumetric data surface 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.

## 6.1 Interactive Volume Cutting with the intelligent scissors

Interactive volumetric data manipulation is an important focus in recent Visualization and Graphics interface research. *Galyean and Hughes*[12] developed a desk-top Polhemus-based system for manipulating a volumetric model interactively. *Serra et al.*[32] proposed a general system for free-form creation of 3D objects with given volume data. *Wang and Kaufman*[39] 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*[24] 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.

In this chapter, we will discuss how to detach a three dimensional ROI from a given

volume intuitively, correctly and efficiently. The next section (Section 6.2) addresses some basic issues concerning contour detection on a volume surface, and introduces the use of Intelligent Scissors on the volume model surface. Section 6.3 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 6.4, and give conclusions in Section 6.5.

## 6.2 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 [18]. We apply it to 3D iso-contour detection, and implement it in the Virtual WorkBench[28, 29], a virtual environment with 3D display and input (see Figure 17). 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.

### 6.2.1 3D Intelligent Scissors

In 2D image segmentation, 'Intelligent Scissors' has been proved to be an efficient method. In the context of 3D 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 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

$$\text{cost}(u,v) = \|p_u - p_v\| \left(w_g * f_g(u,v) + w_n * f_n(u,v)\right) \tag{29}$$

$$f_g(u,v) = 1 - \frac{G(u) + (G(v)}{2\max(G)} \tag{30}$$

$$f_n(u,v) = \frac{1 - Nu \cdot Nv}{2} \tag{31}$$

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.

### 6.2.2 Dijkstra's algorithm

We use the Dijkstra algorithm [6] 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 re-calculated. 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 (Algorithm 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 $\mathrm{c}(u, v)$ can be preprocessed and the only changed function is $\mathrm{B}(u)$, which indicates the path from $u$ to $s$.

## 6.3 3D 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 13 shows how to select it using contours.

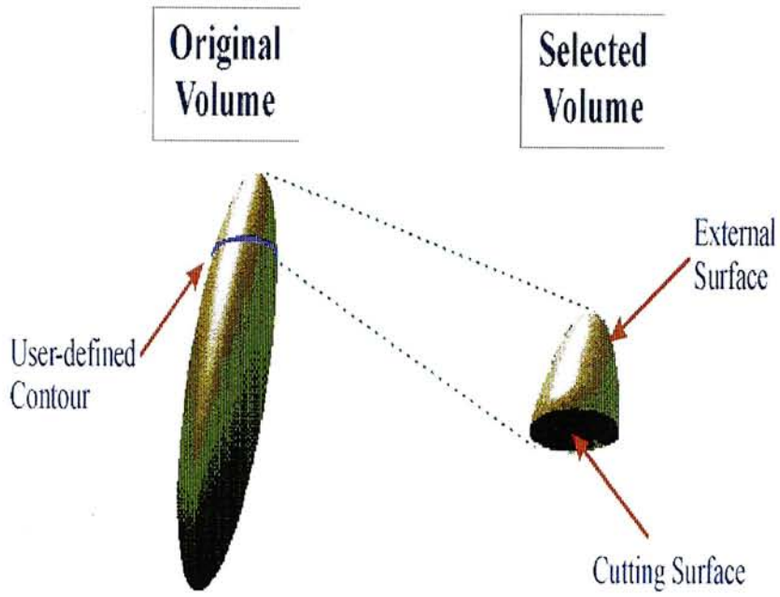
The *external surface* is the part of the whole volume surface, which can be estimated by any mesh generation algorithm [23] and surrounds the interested volume when combined with the *cutting surface.* The shape of this cutting surface is influenced by the shape of the specified contour (boundary), the gradient of the voxels near this surface, and its smoothness.

For surface modeling, a deformable mesh [3, 21] 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, each junction can be bijectively mapped to an unique grid within the interior region of the projected contour (Figure 14). 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.

### 6.3.1 Cost function for the cutting surface

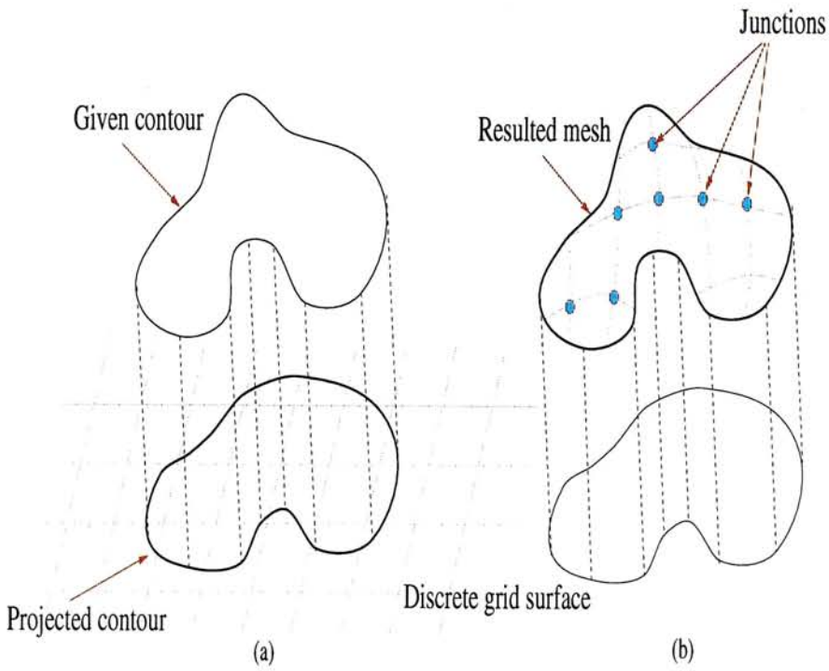
Two vital factors, the continuity between junctions and the voxel gradient near them, are considered for defining the cost function of the mesh. The first factor makes the surface smooth and flat. The second factor drives the surface to fit the iso-surface in the volume. Its function is similar to the image force used with an active contour [20] to fit the edge in a 2D image. The cost function $C(x, y, z)$ for a voxel with the intensity $I(x, y, z)$ can be defined by

$$C(x, y, z) = -\alpha \left| \nabla I(x, y, z) \right| + \beta S(x, y, z), \tag{32}$$



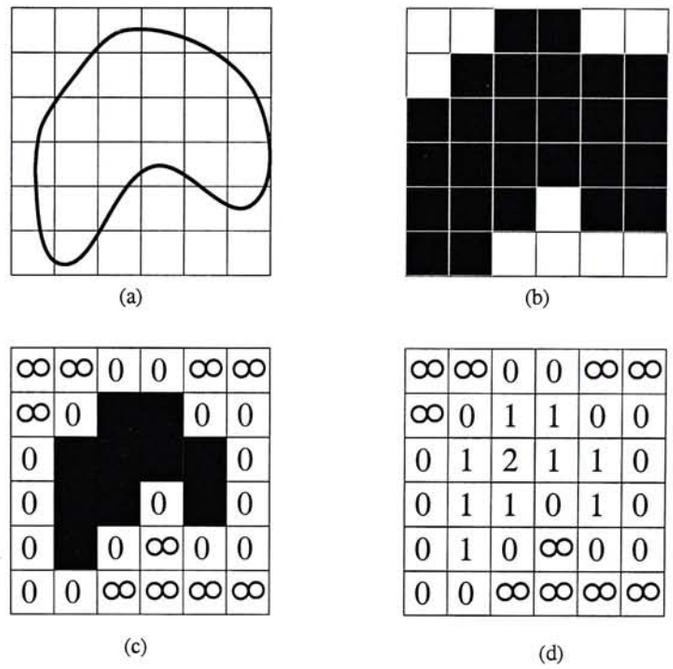

The selected volume is bounded by two surfaces
(external and cutting surfaces).
Their common edge is defined by the user.

Figure 13: The external surface and the cutting surface.




(a) A contour is projected on a discrete grid plane.
(b) Then, the junctions of the surface mesh is generated.

Figure 14: Generation of the deformable mesh.




(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.

Figure 15: Calculation of $L(x, y)$.

where $\alpha$ and $\beta$ are the weighting parameters for the iso-surface fitting factor and the surface continuity respectively, $\nabla$ is the gradient operator, and $S(x, y, z)$, discussed in next section, 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:

$$C_{\mathcal{M}} = \sum_{(x,y,z)\in\mathcal{M}} C(x, y, z) \tag{33}$$

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.

### 6.3.2 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 15, these contours are considered as pixel chains on the discrete grid surface. In our algorithm, the continuity function $S(x, y, z)$ 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 Algorithm 4.

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

Algorithm 4: The algorithm to calculate the map of $L(x,y)$.

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

$$\mathcal{A}_{(x,y,z)} = \mathcal{N}_{(x,y,z)} \cap \mathcal{B}_{(L(x_p,y_p)-1)} \cap \mathcal{G} \tag{34}$$

$$\mathcal{N}_{(x,y,z)} = \left\{ (x',y',z') \mid \left\| \left(x'_p, y'_p\right) - (x_p, y_p) \right\| < \varepsilon \right\} \tag{35}$$

$$\mathcal{B}_k = \{ (x,y,z) \mid L(x_p,y_p) = k \} \tag{36}$$

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 $\varepsilon$ 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

$$S(x,y,z) = \frac{\sqrt{\sum_{\vec{v} \in \mathcal{A}_{(x,y,z)}} \|\vec{v} - (x,y,z)\|^2}}{|\mathcal{A}_{(x,y,z)}|} \tag{37}$$

### 6.3.3 Finding the cutting surface

The mesh defining the cutting surface can be computed by minimizing the cost function $C_{\mathcal{M}}$ (equation 33). This process can be performed efficiently by dynamic-programming techniques. Our algorithm (Algorithm 5) has the following properties:

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

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 $\mathcal{G}$.

### 6.3.4 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 16a 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 16b).

Another kind of problem is that mesh junctions cannot be evenly generated by a simple planar surface. Figure 16c,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.

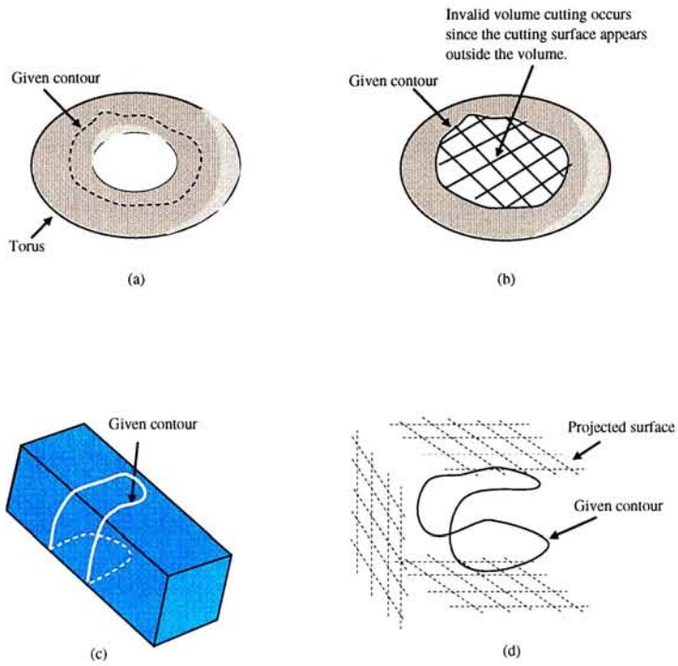
### 6.3.5 Assumptions for the well-conditional contour used in our algorithm

**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 Algorithm 6 to check whether the given contour satisfies Assumption 1.




(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. (c) shows a case of volume cutting with a C-shaped contour. (d) illustrates that three projected planes are used in order to make the junctions on the mesh distributed evenly.

Figure 16: Illustrations of some topological problems.

## 6.4 Implementation and Results

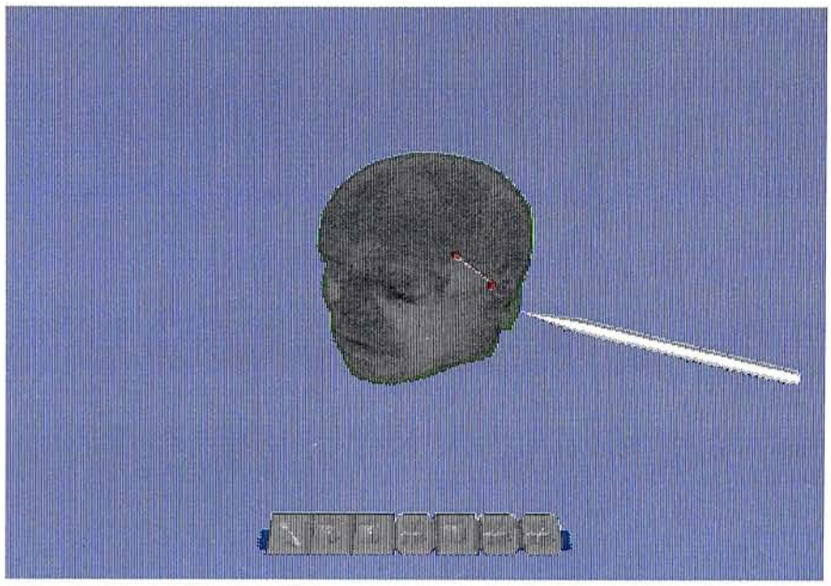
We have implemented the algorithm on the Virtual WorkBench [29, 28], a general purpose reach-in 3D interface that supports stereo display with shutter glasses or mirrors, and 3D input with a 6DOF stylus. We choose this virtual environment because we found that a 2D display of volumetric data is seldom 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 17 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 19 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 equipped with Reality Engine and the rendering is done by 3D texture mapping [36]. In Figure 19, 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.

## 6.5 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.

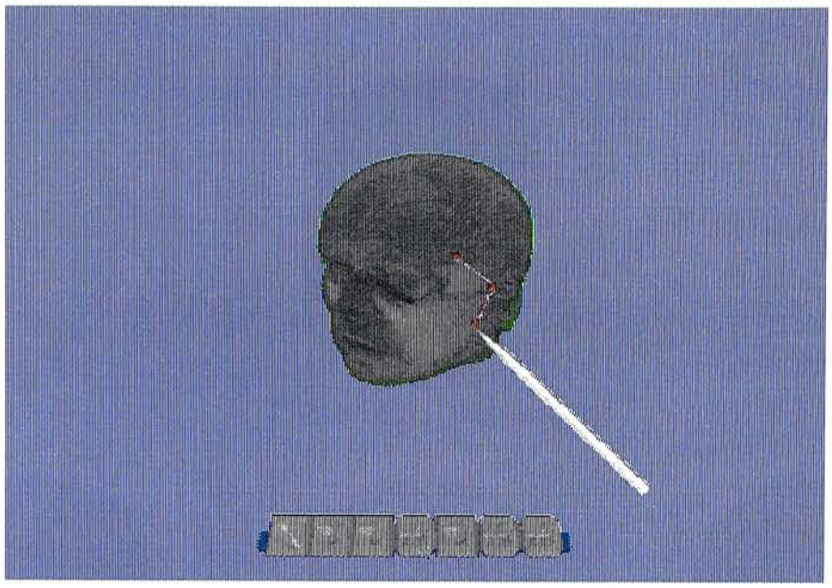
We have also proposed a cutting surface estimation algorithm, by which a region of interest in the volume can be determined by one closed contour on the surface. Dynamic programming reduces the computational complexity.



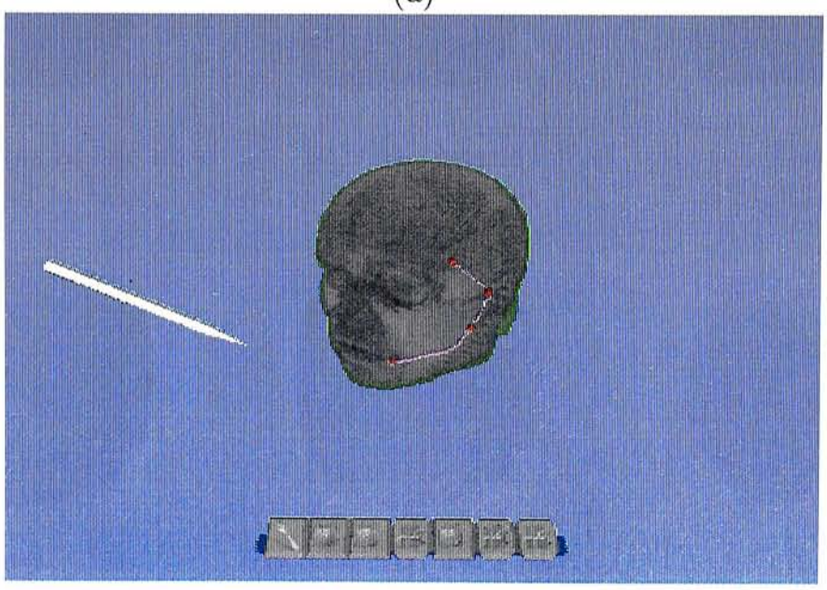
(a)

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.

Figure 17: Drawing a curve segment with a 3D stylus.

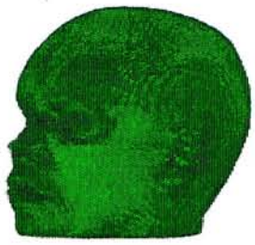


(a)

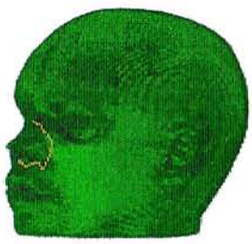


(b)

Figure 18: Drawing the contour by selecting the seed points one by one.

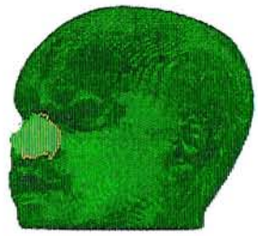


(a)
The original volume.

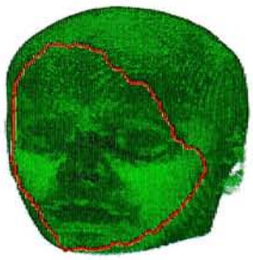


(b) A contour of the nose is defined by user.

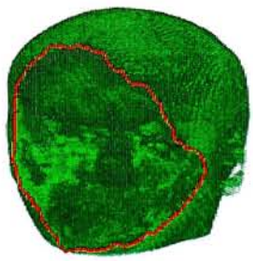
(c)The volume without the nose.



(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.

Figure 19: Volume Cutting results.

| step | instruction |
|---|---|
| 1. | $\mathcal{G} \leftarrow \mathcal{C} \cap \mathcal{B}_0$<br>*// $\mathcal{C}$ is the set of voxels touching the given contour.*<br>*// and $\mathcal{G}$ will be used to find $\mathcal{A}$ by equ. 34.*<br>$k \leftarrow 1$<br>$\forall i, j\ ,\ V[i,j] \leftarrow null$<br>*// $V[i,j]$ is the array of 3D vectors which*<br>*// defines the shape of resulting cutting surface.*<br>$\forall i, j\ ,\ c[i,j] \leftarrow \infty$ |
| 2. | **For each** $(x,y,z) \in \mathcal{C}$<br>$c[x_p, y_p] \leftarrow 0$<br>$V[x_p, y_p] \leftarrow (x,y,z)$<br>*// where $(x_p, y_p)$ is the projected point*<br>*// of $(x,y,z)$ on the discrete grid surface.*<br>**end for** |
| 3. | $\mathcal{D} \leftarrow \mathcal{B}_k$<br>**if** $\mathcal{D} = \oslash$ **then** exit. |
| 4. | **For each** $(x,y,z) \in \mathcal{D}$<br>Compute $\mathcal{A}_{(x,y,z)}$ by equ.34.<br>$m_{tmp} \leftarrow$<br>$C(x,y,z) + \frac{\sum_{(x',y',z') \in \mathcal{A}_{(x,y,z)}} C(x',y',z')}{\|\mathcal{A}_{(x,y,z)}\|}$<br>**if** $m_{tmp} < c[x_p, y_p]$ **then**<br>$c[x_p, y_p] \leftarrow m_{tmp}$<br>*// Here, $x_p, y_p$ are rounded off.*<br>$V[x_p, y_p] \leftarrow (x,y,z)$<br>**end if**<br>**end for** |
| 5. | $\mathcal{G} \leftarrow \bigcup_{i,j} V[i,j] - \{null\}$<br>$k \leftarrow k+1$<br>**Goto** step 3. |

Algorithm 5: The algorithm to compute the cutting surface.

| 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**<br>  **return** false  *// The cut is invalid*<br>**else**<br>  **return** true  *// The cut is valid* |

Algorithm 6: This algorithm checks if a given contour can cut one volume into two sub-volumes.

# Chapter 7

# Conclusions

In this thesis, we have proposed the new solutions to solve the existing 2D and 3D image segmentation problems. In this section, we will have a summary to state our main contributions and the future directions of our research.

## 7.1 Contributions

There are two main contributions in our research:

- We develop the fast intelligent scissor to segment the desired object from the given image of natural scene efficiently, accurately and in a robust manner. The main idea of this technique is to use a slimmed graph to filter out the redundant data in order to speed up the computation in the segmentation phase.

- We develop a freehand volume cutter to segment the interesting partition of a given 3D volume data naturally. Firstly, we select a contour on a volume data surface within a virtual environment. The above proposed tools have been presented on Graphics Interface'98 [42] and published in the Journal of Computer Science and Technology [16]. Moreover, we proposed a cutting surface estimation algorithm to determine the interesting sub-volume by using the closed contour drawn on the iso-surface.

## 7.2 Future Work

The following are some possible applications and extensions of the techniques proposed in this thesis.

### 7.2.1 Real-time interactive tools with Slimmed Graph

Slimmed Graph mentioned in Chapter 4 is a general technique to speed up the image segmentation computation. In current stage, we only applied it on the Intelligent Scissors. In future, we will extend the idea of slimmed graph on the other interactive segmentation tools such as Magic Wand and Active Contour Model [24].

### 7.2.2 3D slimmed graph

The size of the volume data is usually extremely large. The slimmed graph we proposed can also be applied to speed up the 3D volume data analysis. The kernel of the 2D slimmed graph is block decomposition. If we extend this idea to decompose the cube in the 3D space, a 3D slimmed graph for the volume data can be generated easily. Using 3D slimmed graph, we can have the following applications:

- Speed up the Volume Cutting proposed in Chapter 6.
- Generate different level-of-detail of the volume model.
- Perform volume rendering with 3D slimmed graph.

### 7.2.3 Cartoon Film Generation System

The cost of animation production is high because a large amount of drawings from artists are always required. It is a labor-intensive work to create high quality picture

even for a professional artist. Moreover, producing high quality animation is a time consuming task.

With the accurate image segmentation tools, we can extract the profiles and the critical edge data from the natural image. Using these tools, we can develop a computer-based system to convert the real-world photo or film into high quality cartoon animation with a little user-intervention.

The cartoon generation system should contain the following functions:

1. It can accurately extract the profiles from the real-world photo image.

2. It can generate the interpolated images between two key frames with extracted profile.

3. It can render the cartoon-like picture from the results of profile extraction.

# Bibliography


[1] AMINI, A. A., TEHRANI, S., AND WEYMOUTH, T. E. Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints. In *1988 Second International Conference on Computer Vision* (1988), pp. 95–99.

[2] CALWAY, A. D., AND WILSON, R. Curve extraction in images using a multiresolution framework. *Computer Vision, Graphics, and Image Processing. Image Understanding 59*, 3 (May 1994), 359–366.

[3] CELNIKER, G., AND GOSSARD, D. Deformable curve and surface finite elements for free-form shape design. In *Computer Graphics (SIGGRAPH '91 Proceedings)* (July 1991), T. W. Sederberg, Ed., vol. 25, pp. 257–266.

[4] COHEN, L. D. On active contour models and balloons. *Computer Vision, Graphics, and Image Processing. Image Understanding 53*, 2 (Mar. 1991), 211–218.

[5] CONNELLY, S., AND ROSENFELD, A. A pyramid algorithm for fast curve extraction. *Computer Vision, Graphics, and Image Processing 49*, 3 (Mar. 1990), 332–345.

[6] CORMEN, T., LEISERSON, C., AND RIVEST, R. *Introduction to Algorithms.* The MIT Press, Cambridge, MA, 1989.

[7] C.R.BRICE, AND C.L.FENNEMA. Sence analysis using regions. In *Computer Methods in Image Analysis.* (Los Angeles, June 1977), IEEE Computer Society.

[8] ECKEL, G., NEIDER, J., AND BASSLER, E. *ImageVision Library Programming Guide.* Silicon Graphics Inc., 2011 N. Shoreline Blvd., Mountain View, 1996.

[9] FALCIDIENO, B., ORGOLESU, S., PIZZI, C., SANGUINETI, A., AND SPAGNUOLO, M. High fidelity digital terrain modelling for the reconstruction of the Antarctic sea floor. *The Journal of Visualization and Computer Animation 7*, 3 (July–Sept. 1996), 177–187 (or 177–188??).

[10] FREEMAN, H., AND DAVIS, L. A corner-finding algorithm for chain-coded curves. *IEEE Transactions on Computers C-26* (1977), 297–303.

[11] FUJIMURA, K., YOKOYA, N., AND YAMAMOTO, K. Motion tracking of deformable objects based on energy minimization using multiscale dynamic programming. In *International Conference on Pattern Recognition* (1992), pp. 83–86. Vol. A.

[12] GALYEAN, T. A., AND HUGHES, J. F. Sculpting: An interactive volumetric modeling technique. *Computer Graphics 25*, 4 (July 1991), 267–274.

[13] GEIGER, D., GUPTA, A., COSTA, L., AND VLONTZOS, J. Dynamic-programming for detecting, tracking, and matching deformable contours. *PAMI 17*, 3 (March 1995), 294–302.

[14] GLEICHER, M. Image snapping. In *SIGGRAPH 95 Conference Proceedings* (Aug. 1995), R. Cook, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 183–190. held in Los Angeles, California, 06-11 August 1995.

[15] GUNN, S. R., AND S.NIXON, M. A robust snake implementation; a dual active contour. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 1 (1997), 63–68.

[16] HENG, P.-A., WONG, K. C.-H., SIU, Y.-H. S., QING, F. J., AND SUN, H. An application of intelligent scissors in interactive volume manipulation. In *Journal of Computer Science and Technology* (June 1998), pp. –.

[17] JACKWAY, P. T. Gradient watersheds in morphological scale-space. In *1995 IEEE Workshop on Nonlinear Signal and Image Processing* (20-22 June 1995), I. Pitas, Ed., vol. I, IEEE, pp. 22–25.

[18] JAIN, A. K. *Fundamentals of Digital Image Processing.* Prentice-Hall, Englewood Cliffs, NJ, 1989.

[19] KASS, M., WITKIN, A., AND TERZOPOULOS, D. Snakes: Active contour models. In *First International Conference on Computer Vision, (London, England, June 8–11, 1987)* (Washington, DC., 1987), IEEE Computer Society Press, pp. 259–268.

[20] KASS, M., WITKIN, A., AND TERZOPOULOS, D. Snakes: Active contour models. *International Journal of Computer Vision 1*, 4 (1988), 321–331.

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

[22] LIM, Y. W., AND LEE, S. U. On the color image segmentation algorithm based on the thresholding and the fuzzy c-means techniques. *Pattern Recognition 23*, 9 (1990), 935–952.

[23] MILLER, J. V., BREEN, D. E., LORENSEN, W. E., O'BARA, R. M., AND WOZNY, M. J. Geometrically deformed models: A method for extracting closed geometric models from volume data. In *Computer Graphics (SIGGRAPH '91 Proceedings)* (July 1991), T. W. Sederberg, Ed., vol. 25, pp. 217–226.

[24] MORTENSEN, E. N., AND BARRETT, W. A. Intelligent scissors for image composition. In *SIGGRAPH 95 Conference Proceedings* (Aug. 1995), R. Cook, Ed., Annual Conference Series, ACM SIGGRAPH, Addison Wesley, pp. 191–198. held in Los Angeles, California, 06-11 August 1995.

[25] NÄHER, S., AND UHRIG, C. Leda user manual (version r 3.2). Research Report MPI-I-95-1-002, Max-Planck-Institut für Informatik, Im Stadtwald, D-66123 Saarbrücken, Germany, June 1995.

[26] PAVLIDIS, T., AND LIOW, Y. T. Integrating region growing and edge detection. In *CVPR'88 (IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Ann Arbor, MI, June 5–9, 1988)* (Washington, DC., June 1988), Computer Society Press, pp. 208–214.

[27] PHAM, D. L., PRINCE, J. L., DAGHER, A. P., AND XU, C. An automated technique for statistical characterization of brain tissues in magnetic resonance imaging. *International Journal on Pattern Recognition and Artificial Intelligence 11*, 8 (1997), 1189–1211.

[28] POSTON, T. The virtual workbench: a path to use of VR. In *The Industrial Electronics Handbook* (1997), J. D. Irwin, Ed., CRC Press and IEEE Press, pp. 1390–1393.

[29] POSTON, T., AND SERRA, L. The virtual workbench: Dextrous VR. In *Virtual Reality Software and Technology (Proceedings of VRST'94, August 23-26, 1994, Singapore)* (Singapore, Aug. 1994), World Scientific Publishing, pp. 111–122.

[30] PRINCE, J. L., AND XU, C. A new external force model for snakes. In *1996 Image and Multidimensional Signal Processing Workshop* (1996), pp. 30–31.

[31] RUECKERT, D., AND BURGER, P. A multiscale approach to contour fitting for MR images. In *SPIE Conference on Medical Imaging: Image Processing* (Feb. 1996), M. H. Loew and K. M. Hanson, Eds., vol. 2710, SPIE, pp. 289–300.

[32] SERRA, L., HERN, N., CHOON, C. B., AND POSTON, T. Interactive vessel tracing in volume data. In *Symposium on Interactive 3D Graphics* (Apr. 1997), S. N. Spencer, Ed., vol. 25, pp. 131–137.

[33] STALLING, D., AND HEGE, H.-C. Intelligent scissors for medical image segmentation. In *Proceedings of 4th Freiburger Workshop* Digitale Bildverarbeitung in der Medizin, *Freiburg* (Mar. 1996), B. Arnolds, H. Müller, D. Saupe, and T. Tolxdorff, Eds., pp. 32–36.

[34] STALLING, D., AND HEGE, H.-C. Intelligent scissors for medical image segmentation. In *Proceedings of 4th Freiburger Workshop* Digitale Bildverarbeitung in der Medizin, *Freiburg* (Mar. 1996), B. Arnolds, H. Müller, D. Saupe, and T. Tolxdorff, Eds., pp. 32–36.

[35] TERZOPOULOS, D., WITKIN, A., AND KASS, M. Constraints on Deformable Models: Recovering 3D Shape and Nonrigid Motion. *Artificial Intelligence 36*, 1 (1988), 91–123.

[36] VAN GELDER, A., AND KIM, K. Direct volume rendering with shading via three-dimensional textures. In *1996 Volume Visualization Symposium* (Oct. 1996), IEEE, pp. 23–30. ISBN 0-89791-741-3.

[37] VEMURI, B. C., AND RADISAVLJEVIC, A. Multiresolution stochastic hybrid shape models with fractal priors. *ACM Transactions on Graphics 13*, 2 (Apr. 1994), 177–207.

[38] VINCENT, L., AND SOILLE, P. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE PAMI, 1991 13*, 6 (1991), 583–598.

[39] WANG, S. W., AND KAUFMAN, A. E. Volume sculpting. In *1995 Symposium on Interactive 3D Graphics* (Apr. 1995), P. Hanrahan and J. Winget, Eds., ACM SIGGRAPH, pp. 151–156. ISBN 0-89791-736-7.

[40] WANG, Y., AND LEE, O. Active mesh - A video representation scheme for feature seeking and tracking. In *Proceedings SPIE Visual Communication and Image Processing '93* (Cambridge, Massachusetts, USA, 8-11 Nov. 1993), vol. 2094, pp. 1558–1569.

[41] WILLIAMS, D. J., AND MUBARAK, S. A fast algorithm for active contours and curvature estimation. *Computer Vision, Graphics, and Image Processing. Image Understanding 55*, 1 (Jan. 1992), 14–26.

[42] WONG, K. C.-H., SIU, Y.-H. S., HENG, P.-A., AND SUN, H. Interactive volume cutting. In *Graphics Interface* (June 1998), pp. –.

[43] WONG, W. H., SHEN, D., AND IP, H. H. S. Affine invariant shape matching for content-based retrieval. In *Proceedings of ICARCV* (1996).

[44] XU, C., AND PRINCE, J. L. A generalized gradient vector flow for active contour models. In *1997 Conf. Info. Sci. Syst., Johns Hopkins University* (Mar. 1997), pp. 885–890.