Word Grouping In Imaged Documents Using Voronoi Tessellation

WANG ZHE

NATIONAL UNIVERSITY OF SINGAPORE

2004

Word Grouping In Imaged Documents Using Voronoi Tessellation

WANG ZHE

(B.Comp.(Hons.), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2004

ACKNOWLEDGEMENTS

I must give special thanks to the following people, who provided valuable helps throughout the duration of my master study. Their assistance has enabled me to successfully complete this thesis:

A/P Tan Chew Lim, my supervisor, for allowing me to work in his laboratory and providing me with the excellent facilities, and his ingenious supervision.

Dr. Lu Yue, my mentor, for his consistent help and guidance in image processing, and for reading and commenting on the thesis.

Mr. Zhang Zheng, my colleague in CHIME (Center for Information Mining and Extraction), deserves warm thanks for his unconditional support.

Finally, I am deeply grateful to my father **Mr. Wang Zhengxiang** and mother **Mrs. Zhang Hao** for their love and care over the years.

TABLE OF CONTENTS

SUMMARY

In this thesis, a Voronoi tessellation based method is presented for word grouping in imaged documents. Voronoi tessellation of image elements provides an intuitive and appealing definition of proximity, which has been suggested as an effective tool for the description of relations among the neighboring objects in a digital document image. The Voronoi tessellation generated from the input image enables us to obtain neighbor relations between connected components. Based on the neighbor relations, the task of word extraction becomes the problem of selecting appropriate Voronoi edges separating connected components in the same word and then merging those components. For this purpose, we define four characteristic features and we would rely on the features to examine each pair of neighboring connected components locally, and then judge whether we should perform character merging or not. The proposed method has been evaluated on a variety of document images. The experimental results show that it has achieved promising results with a high accuracy, and is robust to various fonts, styles, sizes, as well as different text arrangements.

The thesis is organized into the following chapters: Chapter 1, which is an introductory chapter, provides an overview of the whole project. Some background knowledge and motivation of the project are also discussed. Chapter 2 will describe the processing on eight-neighbor connected components generated from the input document image; this chapter also includes noise removal and special symbol detection. Chapter 3 will give a detailed introduction to Voronoi tessellation, including point Voronoi tessellation and area Voronoi tessellation. Chapter 4 explains why the task of word extraction can be transferred to the problem of selecting appropriate Voronoi edges; the definitions of crucial features and the criteria for Voronoi edge selection are also presented in this chapter. Chapter 5 illustrates experimental results and the performance of word extraction using our system, and the analysis on the error cases. Chapter 6, which is the last chapter, will give a final conclusion.

Chapter 1:

Introduction

1.1 Background of Document Image Processing

Traditionally, transmission and storage of information have been by paper documents. In the past few decades, documents increasingly originate on the computer. It is strongly desirable for computers to deal with paper documents as they deal with other forms of computer media. That is, paper documents should be digitized and could be read by both the computer and human being.

The most widespread format for digital documents is the text in which the characters of the documents are represented by the machine-readable codes (e.g. ASCII codes). The majority of the newly generated documents are in the text format. On the other hand, to make billions of volumes of traditional paper documents available and accessible in the format of digital domain, they are scanned and converted to digital images by the digitization equipments.

The objective of document image analysis is to recognize the text and graphics components in those digital images and extract the intended information as a human would. There are two categories of document processing: 1) textual processing, dealing with the text components of a document image; 2) graphics processing, dealing with the

non-textual components. Figure 1.1 shows a hierarchy of document image processing categories.

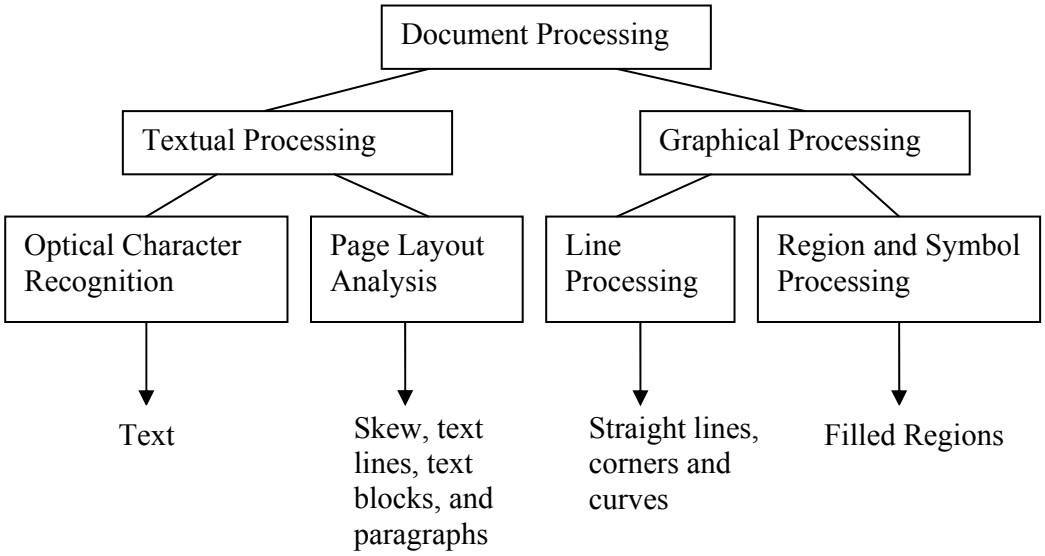


Figure 1.1 Hierarchy of Document Image Processing Categories

Although the technology of Optical Character Recognition (OCR) may be utilized to automatically transfer the digital images of these documents to their machine-readable text format, the OCR has still its inherent weaknesses in the recognition ability, especially for the poor quality document images. Many commercial Optical Character Recognition (OCR) systems, which claim to achieve the recognition rate of above 99% for single characters, have been introduced to the market during the past few decades. However, when confronted with degraded images, the performance of these systems deteriorates severely because all of them rely on the segmentation procedure that is prone to recognition failure at the presence of image noise and poor printing quality. In particular, when adjacent characters are joined or fused, an OCR system must perform the delicate task of separating them. However, under some conditions perfect segmentation would be

impossible. It has been estimated that half of the errors in the character recognition occur during segmentation in locating the individual characters within a word.

Generally speaking, manual correction/proofing of the OCR results are usually unavoidable, which is typically not cost effective for transferring a huge amount of paper documents to their text format. Moreover, the technology of layout analysis is still immature for the documents with complicated layouts. As a result, storing these documents in the image format has become an alternative way in many cases. Nowadays, many digital documents are in the image format. It is therefore of significant meaning to study the strategies of retrieving information from these document images.

1.2 Motivation of the Project

In the field of image processing, the problem of text extraction from a document image remains an important issue. As we know, a document structure analysis system converts a scanned document page or a document encoded by a Page Description Language (PDL), such as Postscript and Portable Document Format (PDF), into a well partitioned hierarchical representation that reliably identifies the basic document components – text words, text lines, and text blocks. Thus, extracting (segmenting) word objects from document images is an essential component for most document analysis and recognition systems. Its goal is to group a set of image elements of characters, touched characters, or character portions into a word. The accuracy of the word grouping greatly influences the performance of the systems. If word grouping is incorrectly done, serious irreparable

errors could occur in the subsequent processing. This is especially true for most word-based recognition systems and word spotting systems. However, it is not trivial to develop a word segmentation method with not only high accuracy but also robustness to various documents.

The digital library of NUS maintains a collection of historical documents such as a century-old set of Chinese newspapers and past students' theses for public access on the web. These documents were scanned in as images without any conversion to text due to the prohibiting cost of optical character recognition and manual correction. In order to make use of these imaged documents, we need to develop an efficient technique to extract words from them.

1.3 Existent Word grouping Techniques and Our Method

Many methods for word grouping in document images have been suggested by researchers. Fletcher and Kasturi [4] described a method in which Hough transform was used to group characters into words. Hough transform is applied to the centroids of the rectangles enclosing each connected component; the collinear connected components are thus located. The positional relationships between the collinear connected components are then examined to locate the words. This method, however, is not capable of dealing with the documents with various sizes of characters, because the process depends on the average height of all connected components. Jain and Bhattacharjee [7] considered text image as textured objects and used Gabor filtering for text analysis. Obviously, this

method is sensitive to font sizes and styles, and it is generally time-consuming. Ittner and Baird[6] assumed that the distribution of scaled inter-symbol distance parallel to text-line orientation is bimodal with one model for inter-symbol spaces and the other mode for inter-word spaces. Wang et al.[16] presented a statistical-based approach to text word grouping that takes a set of bounding boxes of glyphs and their associated text lines of a given document and partitions the glyphs into a set of text words. The probabilities, estimated off-line from a document image database, drived all decisions in the on-line text word grouping. An accuracy of about 97% was reported. In [12], Park introduced a 3D neighborhood graph model which can group words in inclined lines, intersecting lines, and even curved lines. Sobottka [13] proposed an approach to automatically extract text from colored books and journal covers. Tan and Ng [14] gave a method using irregular pyramid structure. The uniqueness of this algorithm is its inclusion of strategic background information in the analysis.

A crucial step for word grouping is to find the neighbors in proximity of a particular image element. A naive approach is to compare the distances of the element to all the others in the image, and the neighbors are then defined by those with shorter distances. Such definition is not always accurate, because an element with a short distance is not a real neighbor sometimes. Voronoi tessellation (also named as Voronoi tessellation) provides a useful tool which is capable of generating minimal in the number but complete neighbors of an element, i.e. only those elements that are closest are obtained, but all are included. The Voronoi tessellation of a collection of geometric objects is a partition of space into cells, each of which consists of all the points closer to one particular object than

to any others. It divides the continuous space into mutually disjoint subspace according to the nearest neighbor rule. In the past decades, increasing attentions have been paid to the use of Voronoi tessellation for various applications.

The most important and significant contribution of the Voronoi tessellation to image analysis is that it introduces neighboring relations into a set of elements (e.g. connected components) on a digital image. In particular, it enables us to obtain neighbors without recourse to predetermined parameters. In recent years, there are some reports in the literature about applying Voronoi tessellation to document image analysis. For instance, Ittner and Baird [6] applied the Delaunay triangulation, dual of the Voronoi tessellation, to detect the orientation of text lines in a block, based on the assumption that most Delaunay edges lie within rather than between text lines. Xiao and Yan[17] described a method of text region extraction using the Delaunay tessellation. In both of the above two methods, the connected components in a document image are represented by their centroids. Such simplification is inappropriate in some cases, because the centroid is a poor representation of shapes for non-round elements. Since a document image generally contains various characters of different sizes and different intercharacter gap, the approximation of each element as a single point is too imprecise, and it does not adequately represent the spatial structure of the page image. Therefore, the point Voronoi tessellation is unsuitable for some applications.

Considering the complex shapes of image elements, the use of area Voronoi tessellation has been investigated for document image analysis. For example, Wang et al. [15] applied

area Voronoi tessellation for segmenting characters connected to graphics, based on the observation that area Voronoi tessellation represents the shape of connected components better than the bounding box does. Kise et al. employed area Voronio tessellation to perform page segmentation [10] and text-line extraction [9].

Word grouping would evidently benefit from the information provided by the Voronoi tessellation. However, the research on this topic has not been extensively studied so far, except Burge and Monagan's work [2] which made an attempt using the Voronoi tessellation for grouping words and multi-part symbols in a map understanding system. An obvious shortcoming of their method is that it requires the necessary information of the resolution at which the processed image was scanned. In most general sense, the image resolution is unknown for a document analysis system in most cases.

Based on the area Voronoi tessellation, a method for grouping the image elements to word objects is proposed in this thesis. No priori knowledge such as character font, character size or inter-character spacing is required for the proposed method, and no special word orientations are assumed. Experimental result on real document images shows that more than 99% of words are successfully extracted.

Figure 1.2 shows the procedures of our algorithm. It consists of totally five steps. Firstly, the input document image is processed to find all the connected components inside it. Secondly, we perform some preprocessing tasks, such as noise removal and special symbol detection. Next, an area Voronoi tessellation will be generated using the connected

components found in the last step as generators. Fourthly, we will select those appropriate Voronoi edges, which separate connected components potentially in a word object, within the newly generated area Voronoi tessellation. The final step involves merging connected components which are separated by those newly selected Voronoi edges in Step Four.

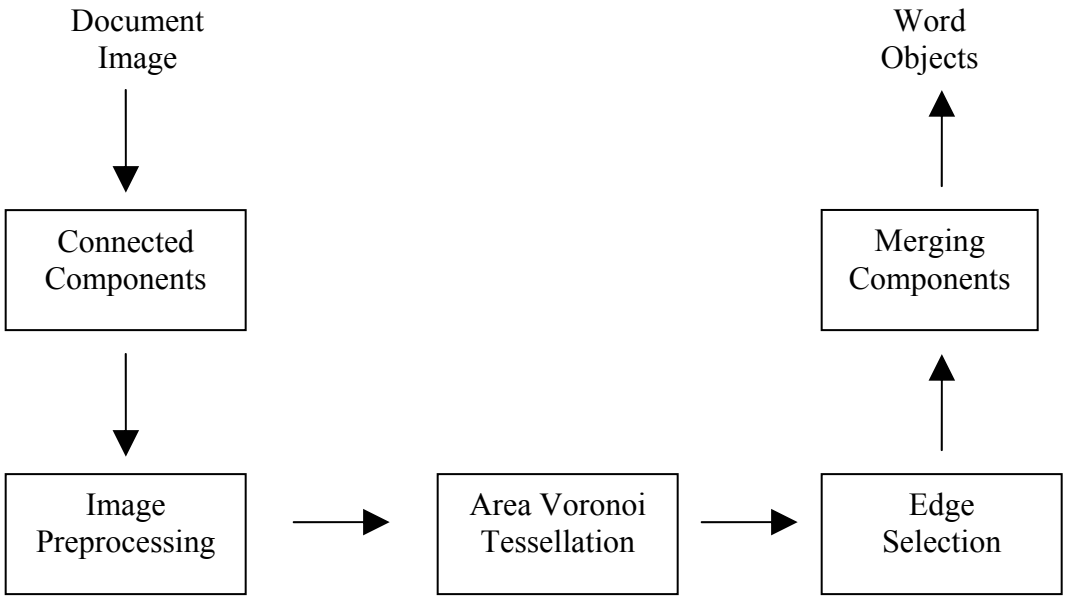


Figure 1.2 Overview of Word grouping Algorithm

1.4 Thesis Outline

The thesis is organized into the following chapters.

- Chapter 1 provides an overview of the whole project. Some background knowledge and motivation of the project are discussed here.

- Chapter 2 will describe the processing on eight-neighbor connected components generated from the input document image. This chapter also includes merging overlapped components and special symbol detection.

- Chapter 3 will give a detailed introduction to Voronoi tessellations, including Voronoi region, Voronoi edge, Voronoi point, point Voronoi tessellation and area Voronoi tessellation.

- Chapter 4 explains why the task of word grouping can be transferred to the problem of selecting appropriate Voronoi edges. Also we will present the criteria for edge selection.

- Chapter 5 illustrates experimental results and the performance of word grouping using this system, and also the analysis on the error cases.

- Chapter 6 will give a conclusion and present the advantages of our method; also we will do comparisons between our method and other existent ones.

Chapter 2:

Image Preprocessing

2.1 Finding Connected Components

In most image processing systems, the foremost step to be done is to find out the connected components. The images to be processed are assumed to be binary images with black and white pixels, so a connected component is a set of pixels whose value is black and all the pixels are connected to each other as a part. Finding the connected component is the most common step that needs to be done firstly. Our objective here is to find out the position of the characters so that later on we can do some merging on these characters to form words.

In the process of finding the connected components, we need to use concept of "neighbors of a pixel". Neighbor pixels indicate those that are so called connected to the examined pixel. In document analysis we normally use 8-neighbor or 4-neighbor. 4-neighbors are the 4 pixels that are left to, right to, above and below the pixel. 8-neighbors are the 8 pixels that are around the pixel. 8-neighbor is used in the present project.

The traditional way of finding the connected components is using backtracking. It checks the pixels one by one from left to right and from top to bottom. We only consider the black pixels. If the pixel is not connected to the connected components currently marked

(by checking if its neighbors are marked) then this pixel will be marked as a new connected component (by giving a label to the corresponding matrix entry). Otherwise it will check if the neighboring black pixels of this pixel are marked as the same connected component. If all marked neighboring pixels are the same, then this pixel is marked the same value (connected component labeling) as its marked neighboring pixels. If the marking values of the neighboring pixels are not the same, then it now notes down the values of the marked neighboring pixels. Then do the backtracking and for all the pixels that have the value that is the same with a value that has been noted down it will reset the pixel's value to be the smallest value that have been noted down. This method needs too much memory and computational time because of the backtracking.

We use a component-oriented method to solve the problem. In this method, the main idea is that when we meet a pixel in a new connected component (the pixel is not marked) then we keep searching and marking all the neighbors starting from this pixel until we mark all the pixels in this connected component to have the same value. But in actual programming since we are only interested in the characters' positions rather than those marked connected components, we mark down the range when we find a new connected component. That is, when we find a new black pixel, we will firstly store all its neighboring black pixels and set this pixel to be white (because we have finished processing this pixel). Then for all the neighboring pixels we have store we do the same checking for their neighboring pixels until all the neighboring pixels are white. During this searching process, we will use a rectangle to save the range of this connected component. After searching all the pixels in this connected component, the range of this connected

component is given out by the rectangle and also all the pixels in this connected component become white pixels now so that later we don't need to check these pixels when we will find a new connected components. So now for each pixel we only do one operation, so the memory usage are smaller than the traditional way and the speed is also faster than that way.

2.2 Noise Removal

After finding the connected components, now we have an array of rectangles BlockBox[]. Each rectangle contains one connected component which we call it a block. But not all the blocks are meaningful for the word grouping. Some blocks may contain noise, or graphics and tables in the document image. But we are not interested in these connected components when we do word grouping. So the simplest way that can remove the noise is deleting those blocks that are very high. The reason why we only delete very high blocks is that, there can be very long block if the image quality is not so good and the characters in a word are connected with each other. We don't delete those very small blocks at the moment because some very small rectangles are meaningful like the dots in "i" and "j". For some very small noise, we can use a simple way to remove them. For any two blocks, if one block is in the other, the smaller block is deleted. Now we can remove the noise that is inside the character blocks.

2.3 Merging Overlapping Component

It may happen that two blocks partially overlap with each other. Just like the case in Figure 2.1(a), “f” and “r” overlap and have a common area. For the need of future computation, we should merge them two into one block. As a result, all of the blocks in the document image are non-overlapping, as shown in Figure 2.1(b).

(a) (b)

Figure 2.1 Merge Overlapped Components

2.4 Special Symbol Detection

In English, there are two types of special symbols. The first one is for the special symbols which are elongated horizontally such as dash (‘-’) or tilde (‘~’). The second one is for the vertically elongated symbols such as various kinds of parentheses ‘{’, ‘}’, ‘[’, ‘]’, ‘(’, or ‘)’. These special symbols are not considered as an element in a word, so we need to detect them using a ruled-based algorithm without involving the recognition of any symbols, and then eliminate them from the block array BlockBox[].

Refer to Figure 2.2, the dash symbol between “Shang” and “hae” pointed by arrow 1 belongs to special symbol type one; those parentheses (round brackets and square

brackets) pointed by arrows 2 belong to special symbol type two. In order to detect all of them, we propose the following rule-based algorithm with reference to [8]:

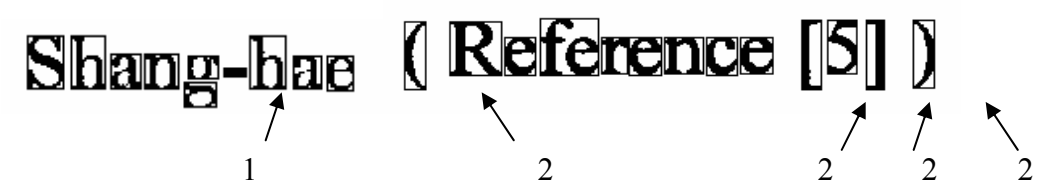


Figure 2.2 Special Symbols

(Rule 1)

For a connected component with width W_i and height H_i respectively, if it satisfies both the following two conditions, it is regarded as a special symbol of the first type:

(1) $W_i > 2 \times H_i$

(2) $H_i < T_1$

where T_1 is 30% of the median height among all connected components

(Rule 2)

On the other hand, if one connected component satisfies the following four conditions, it is regarded as a special symbol of the second type:

(1) $H_i > 2 \times W_i$

(2) $D_i < 0.75 \times W_i \times H_i$

where D_i is the number of black pixels of the connected component

(3) The upper and lower parts of the connected component are symmetric and the left and right parts are not symmetric

The above two rules are effective to remove most of the special symbols, therefore greatly increase the accuracy of word grouping.

Chapter 3:

Voronoi Tessellation

In this chapter, we give a brief review of the definitions of Voronoi tessellation. For further detail, please refer to [11].

3.1 Introduction

Voronoi tessellation is a geometric structure which is probably the most famous in computational geometry. Intuitively speaking, a Voronoi tessellation divides the available space spanned by a number of given locations (called sites), according to the nearest-neighbor rule: each site P gets assigned the region (of the plane, say) which is closest to P. Refer to Figure 3.1 below; a honeycomb-like structure is obtained.

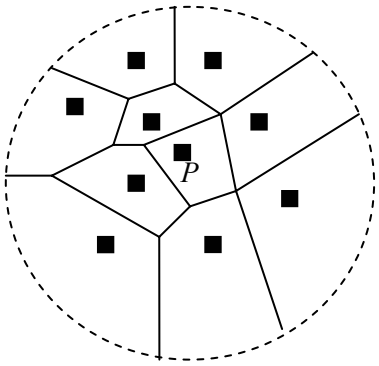


Figure 3.1 Voronoi Regions

Imagine the given sites are post offices, then, for any chosen location of a customer, the containing Voronoi region makes explicit the post office closest to him/her. More abstractly, by performing point location in the data structure 'Voronoi tessellation', the nearest neighbor site of any query point can be retrieved quickly (in fact, in $O(\log n)$ time). This is optimal by a matching information-theoretic bound). Similarly, sites may represent department stores, and Voronoi neighborhood - witnessed by diagram edges - will indicate stores in strongest mutual influence (or competition). Apart from economics, this basic finding has far-reaching applications in biological and physico-chemical systems.

On the other hand, Voronoi vertices are places where the influence from sites reaches a local minimum – a fact of interest in facility location. Similar observations apply to the robotics scenery: sites are snapshots of moving robots, and danger of collision is most acute for the closest pair. Moreover, when planning a collision-free motion of a robot among a given set of obstacle sites, sticking to the boundaries of Voronoi regions (that is, moving along Voronoi edges and vertices) will keep the robot off the obstacles in a best possible way. (This is known as the retraction approach in motion planning.)

After the Russian mathematician George Voronoi - believed to be the first to formally introduce it - this diagram has been reinvented and used in the past century in various different sciences. Area-specific names like Wigner-Seitz zones (chemical, physics), domains of action (crystallography), Thiessen polygons (geography), and Blum's transform (biology) document this remarkable fact. Voronoi tessellation has proved to be a powerful tool in solving seemingly unrelated computational questions, and efficient and

reasonably simple techniques have been developed for their computer construction and representation. Moreover, Voronoi tessellation has surprising mathematical properties and is related to many well-known geometric structures. Finally though, human intuition is guided by visual perception: if one sees an underlying structure, the whole situation may be understood at a higher level.

Nowadays, in the field of image processing, including document image analysis, Voronoi tessellation has sometimes been used as the representation of input images. In the following sections, we will present a description of two special members in the Voronoi tessellation family – point Voronoi tessellation and area Voronoi tessellation. The latter one plays a curial role in our system.

3.2 Definitions and Point Voronoi Tessellation

Let $P = \{p_1,..., p_n\}$ be a set of points (or generators) in the two-dimensional plane (the input document image), and $d(p_1, p_2)$ be the Euclidean distance between points p_1 and p_2. By $\overline{p_1 p_2}$ we denote the line segment from p_1 to p_2. The closure of a set A will be denoted by $\overline{A}$.

For $p_1, p_2 \in P$, let

$$B(p_1, p_2) = \{x \mid d(p_1, x) = d(p_2, x)\} \qquad (3.1)$$

be the bisector of p_1 and p_2. $B(p_1, p_2)$ is the perpendicular line through the center of the line segment $\overline{p_1 p_2}$. It separates the halfplane

$$D(p_1, p_2) = \{x \mid d(p_1, x) < d(p_2, x)\} \tag{3.2}$$

containing p_1 from the halfplane $D(p_2, p_1)$ containing p_2. We call

$$VR(p_1, P) = \bigcap_{p_2 \in P, p_2 \neq p_1} D(p_1, p_2) \tag{3.3}$$

the Voronoi region of p_1 with respect to P.

The ordinary Voronoi tessellation generated from a set of points P is given as a set of Voronoi regions

$$V(P) = \{VR(p_1, P), ..., V(p_n, P)\} \tag{3.4}$$

We call this the ***point Voronoi tessellation***.

3.3 Divide & Conquer Algorithm of Constructing Point Voronoi Tessellation

There are several ways of computing the point Voronoi tessellation, such as incremental construction, divide & conquer, sweep and lifting to 3-space. In this section, we will introduce one of them, which is implemented in our system --- divide & conquer algorithm.

It is the first deterministic worst-case optimal algorithm for computing the Voronoi

tessellation. In the divide & conquer approach, the set of point sites, S, is split by a dividing line into subsets L and R of about the same sizes. Then, the Voronoi tessellations $V(L)$ and $V(R)$ are computed recursively. The essential part is in finding the split line, and in merging $V(L)$ and $V(R)$, to obtain $V(S)$. If these tasks can be carried out in time $O(n)$ then the overall running time is $O(n \log n)$. During the recursion, vertical or horizontal split lines can be easily found if the sites in S are sorted by their x- and y-coordinates beforehand. The merger step involves computing the set $B(L,R)$, the bisector of L and R, of all Voronoi edges of $V(S)$ that separate regions of sites in L from regions of sites in R.

Suppose that the split line is vertical, and that L lies to its left. Thus, $V(S)$ can be obtained by gluing together $B(L,R)$, the part of $V(L)$ to the left of $B(L,R)$, and the part of $V(R)$ to its right. Please refer to Figure 3.2, where $V(R)$ is depicted by dashed lines.

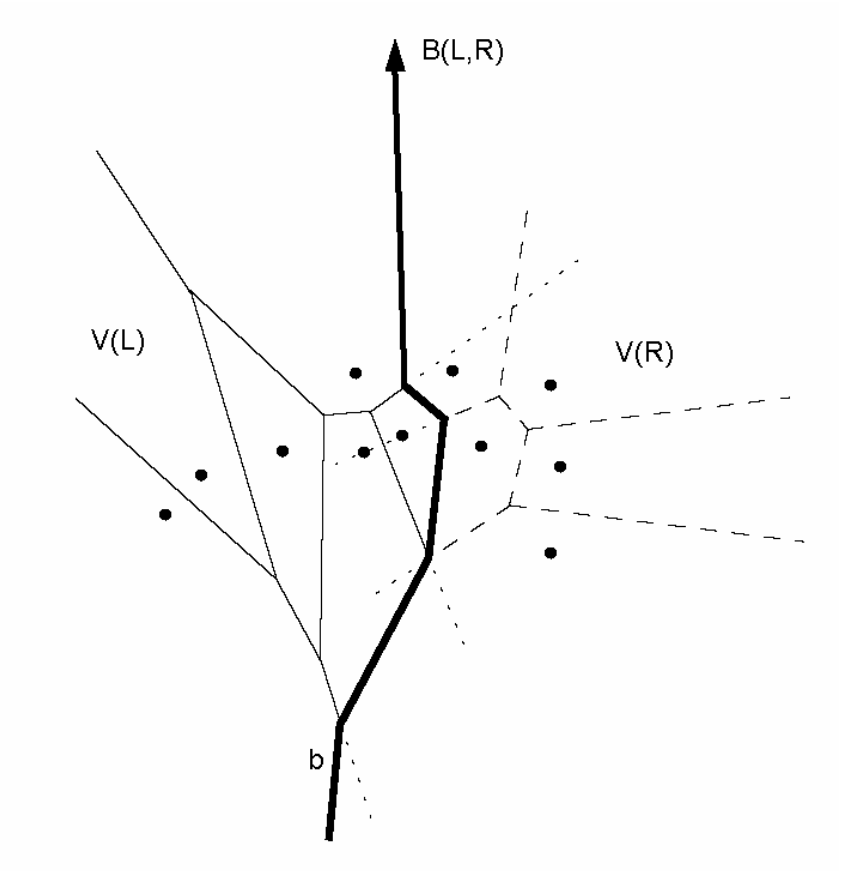


Figure 3.2 Merging $V(L)$ and $V(R)$ into $V(S)$

The polygonal chain $B(L,R)$ is constructed by finding a starting edge at infinity, and by tracing $B(L,R)$ through $V(L)$ and $V(R)$. An unbounded starting edge of $B(L,R)$ can be found in $O(n)$ time by determining a line tangent to the convex hulls of L and R respectively. Here we describe an alternative method. The unbounded regions of $V(L)$ and $V(R)$ are scanned simultaneously in cyclic order. For each non-empty intersection $VR(l,L) \cap VR(r,R)$, we test if it contains an unbounded piece of $B(l,r)$. If so, this must be an edge of $B(L,R)$. Since $B(L,R)$ has two unbounded edges, this search will be successful. It takes time $|V(L)| + |V(R)| = O(n)$.

Now we describe how $B(L,R)$ is traced. Suppose that the current edge b of $B(L,R)$ has just entered the region $VR(l,L)$ at point v while running within $VR(r,R)$, refer to Figure 3.3. We determine the points v_L and v_R where b leaves the regions of l resp. of r. The point v_L is found by scanning the boundary of $VR(l,L)$ counterclockwise, starting from v. In our example, v_R is closer to v than v_L, so that it must be the endpoint of edge b. From v_R, $B(L,R)$ continues with an edge b_2 separating l and r_2. Now we have to determine the points $v_{L,2}$ and $v_{R,2}$ where b_2 hits the boundaries of the regions of l and r_2. The crucial observation is that $v_{L,2}$ cannot be situated on the boundary segment of $VR(l,L)$ from v to v_L that we have just scanned. Therefore, we need to scan the boundary of $VR(l,L)$ only from v_L on, in counterclockwise direction. The same reasoning applies to $V(R)$; only here, region boundaries are scanned clockwise.

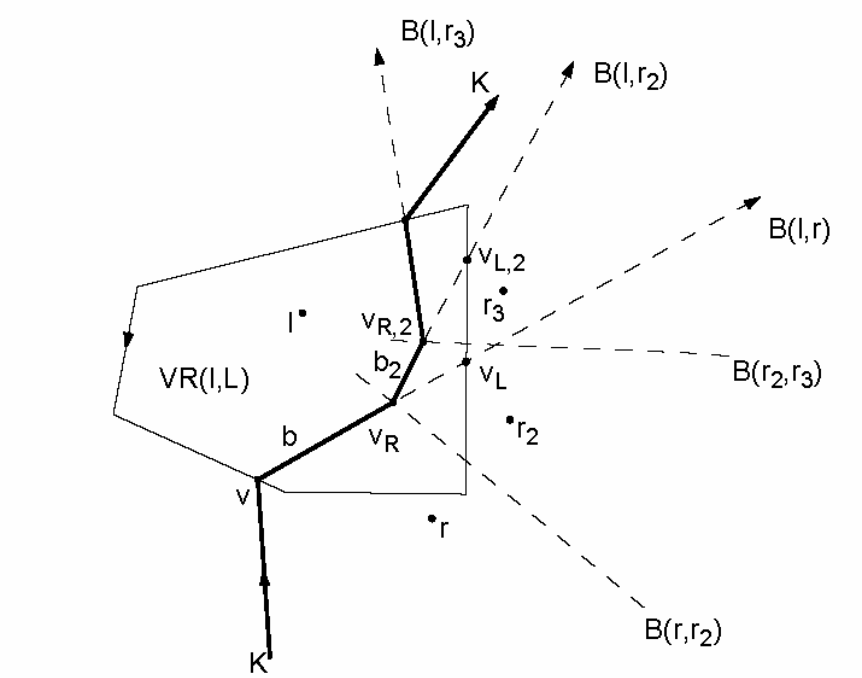


Figure 3.3 Computing the Chain $B(L,R)$

3.4 Area Voronoi Tessellation

The point Voronoi tessellation has been generalized in many directions. We focus here on the generalization of generators from points to figures of arbitrary shape. Such a Voronoi tessellation is sometimes called the area Voronoi tessellation.

Let $G = \{g_1,...,g_n\}$ be a set of non-overlapping figures (reason for merging overlapping components in section 2.3) in the two-dimensional plane, and let $d(p,g_i)$ be the Euclidean distance between a point p and a figure g_i defined as

$$d(p, g_i) = \min_{q \in g_i} d(p, q) \qquad (3.5)$$

where q is a point in g_i . Then the Voronoi region $V(g_i)$ and the ***area Voronoi tessellation*** $V(G)$ are defined as

$$V(g_i) = \{ p \mid d(p, g_i) \le d(p, g_j), \forall j \ne i \ \} \qquad (3.6)$$

$$V(G) = \{ V(g_1), ..., V(g_n) \ \} \qquad (3.7)$$

The boundaries of Voronoi regions may consist of line segments, half lines, or infinite lines, which are called ***Voronoi edges***. A ***Voronoi point*** indicates a point where Voronoi edges come in contact.

It is known that the area Voronoi tessellation can be approximately constructed by applying an algorithm for the point Voronoi tessellation in the following manner.

Step 1. Let $P_i = \{ p_{i1}, ..., p_{im_i} \ \}$ be a set of points lying on the boundary of a figure g_i .

Step 2. Generate the point Voronoi tessellation from the generators $P = \{ P_1 \cup ... P_n \ \}$.

Step 3. For all i, j, k , delete Voronoi edges generated from points p_{ij} and p_{ik} , i.e., Voronoi edges generated from points on the same figure.

Although a Voronoi edge of the area Voronoi tessellation can be a complex curve, the approximated version consists of line segments, which are the approximation of the curve. In what follows, the approximated version is referred to as the area Voronoi tessellation for simplicity.

3.5 Area Voronoi Tessellation Used in Our System

We first construct the point Voronoi tessellation from a document image by, for instance, using the centroids of connected components. However, the point Voronoi tessellation is unsuitable for the task of word grouping, since the approximation of each connected component as a single point is too imprecise. Fortunately, we can also construct the area Voronoi tessellation in a straightforward manner using the algorithm discussed in section 3.4. The generator, which is a set of non-overlapping figures $\{g_1,...,g_n\}$, is now the array of connected components BlockBox[], i.e. $g_1,...,g_n$ are all set to be blocks. Figure 3.4 shows an example of area Voronoi tessellation built from a document image.

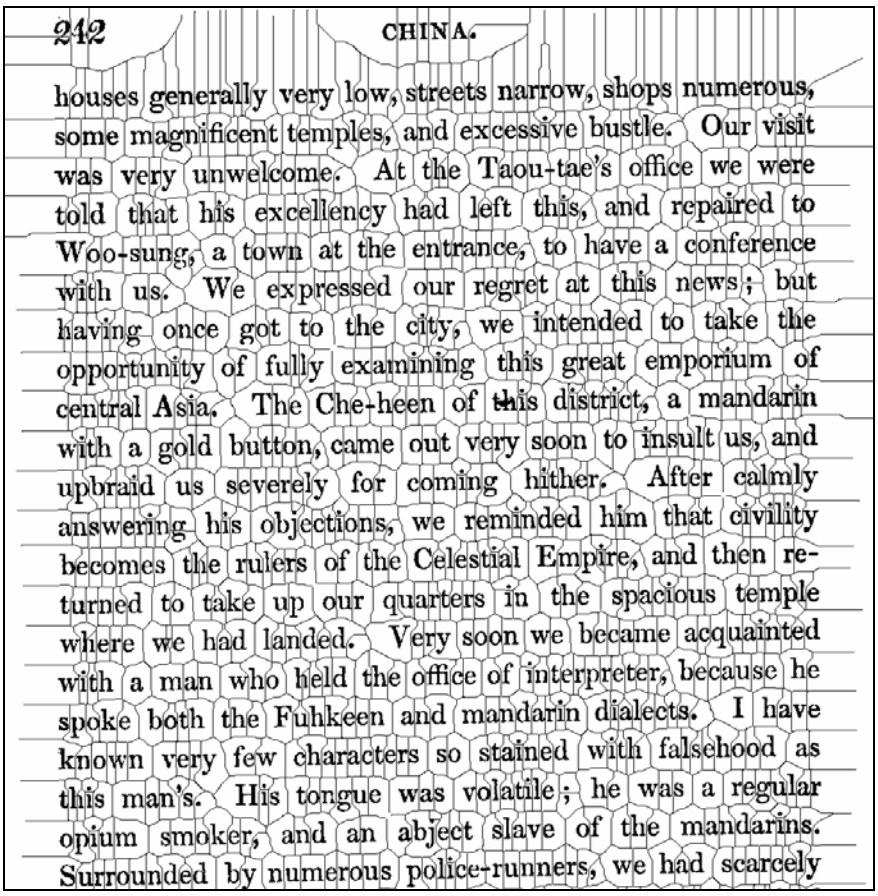


Figure 3.4 Area Voronoi Tessellation Example

Chapter 4:

Word Grouping

Now we are in the stage of grouping words from the area Voronoi tessellation. Basically, word grouping is equivalent to finding those connected components in the same word and then merge them into one. However, the problem is that, without any hint, we are unable to tell which components are in the same word and should be grouped together. Fortunately, area Voronoi tessellation is a powerful weapon in our hand, and it can help us smooth this obstacle.

4.1 Voronoi Edge Selection

Let's put our sight back to Figure 3.4, observe that each Voronoi edge lies between two adjacent Voronoi regions, which are generated by respective connected components. In other words, in between any two neighboring connected components, there must be a Voronoi edge separating them. Therefore, finding connected components in the same word can be realized by the selection of Voronoi edges which separate two connected components potentially in the same word. This is actually a feasible task, because, as we described before, the area Voronoi tessellation is a superior computational geometry structure such that it contains sufficient information for us to employ.

It can be summarized as follows: every word in the document image is represented as a set of connected components which are adjacent with one another; finding these components can be done by Voronoi edge selection; hence, the problem of word grouping is transferred to be the problem of selecting appropriate Voronoi edges which separate connected components in the same word.

4.2 Neighborhood and Distance

A test image, as in Figure 4.1, with skewed text of different font styles and sizes, is used to demonstrate the performance of our proposed word grouping method based on the analysis of Voronoi neighborhoods. As shown in Figure 4.1(a), Voronoi edges lie between any two adjacent connected components (elements). In other words, every word component is represented as a set of Voronoi tessellations which are adjacent with one another. If two elements e_i and e_j share parts of their Vornoroi edges, they are said to be Voronoi neighbors each other.

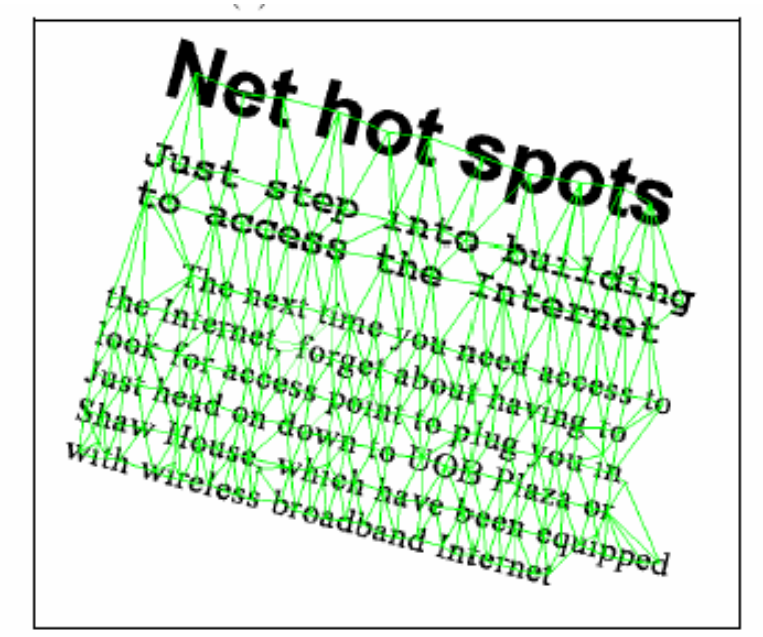
From Figure 4.1(b), we can find that the Delaunay triangulation, the dual of the Voronoi tessellation, shows us the relations among Voronoi neighbors. Each edge of the triangles connects two Voronoi neighbors. As a result, a neighborhood graph can be constructed from the area Voronoi tessellation. In the neighborhood graph, each node represents an image element, and each edge is a connection to its neighboring element. The distance between two Voronoi neighbors is treated as the weight of the edge connecting them. Such an edge is represented using a 3-tuple:

$$\xi_{ij} = \{e_i, e_j, d_{ij}\} \qquad (4.1)$$

where e_i and e_j are two Voronoi neighbors and the distance d_{ij} is defined as follow. Suppose $L_{ij} = \{l_1,\ldots,l_m\}$ be the Voronoi edge between the two elements e_i and e_j, where l_k is a point on L_{ij}. Note that $d(l_k, e_i) \equiv d(l_k, e_j)$ according to the definition of Voronoi edge, but in digitized space $d(l_k, e_i) = d(l_k, e_j) \pm 1$ is also a possible case. We define the d_{ij} as the shortest distance summation of the distance from the Voronoi edge L_{ij} to the elements e_i and e_j.

$$d_{ij} = \min_{1 \le k \le m}(d(l_k, e_i) + d(l_k, e_j)) \qquad (4.2)$$

(a) Voronoi tessellation



(b) Delaunay triangulation

Figure 4.1 Voronoi tessellation of a document image.

Table 4.1 lists the distances of some neighbor pairs generated from Fig. 4.1. Then the goal of grouping elements to words becomes a goal to search the neighborhood graph to generate sub graphs, so that connections among elements from different words are deleted, but connections among elements belonging to same word objects remain. The process of word grouping is, therefore, considered to be the selection of the edges in the neighborhood graph, which connects two elements potentially in the same word. To this end, we need criteria for deciding which connecting lines should be deleted, and which should remain.

We can see from Fig. 4.1 and Table 4.1, that an element generally has more than three neighbors. However, in most cases, only one or two of the neighbors are within the same

word that the element belongs to. For example, in the case that two neighbors belong to the same word, one is its preceding character; the other is its succeeding character.

neighbor pair	distance	neighbor pair	distance	neighbor pair	distance
(1,2)	9	(2,4)	8	(3,5)	10
(1,8)	58	(2,1)	9	(3,4)	33
(1,9)	63	(2,13)	60	(3,18)	63
(1,12)	63			(3,25)	63
				(3,15)	70
(4,2)	8	(5,6)	8	(6,5)	8
(4,3)	33	(5,3)	10	(6,10)	27
(4,17)	64	(5,20)	59	(6,29)	64
(4,15)	76	(5,27)	66	(6,24)	70
		(5,25)	84	(6,20)	73
(7,11)	8	(8,9)	1		
(7,10)	11	(8,16)	19		
(7,36)	36	(8,19)	37		
(7,33)	59	(8,1)	58		
(7,30)	71				
…	…	…	…	…	…

Table 4.1 Neighbor pairs and their distances

Therefore, we take into account only the two nearest neighbors. One is of the shortest distance and the other is of the second shortest distance from the element. An example is shown in Figure 4.2, where only the two nearest neighbors with the shortest distance are selected, whereas the others are excluded from further processing, i.e. we need only consider whether the two nearest neighbors should be grouped with the element in the subsequent process. With this, most of the connections between text lines are effectively eliminated.

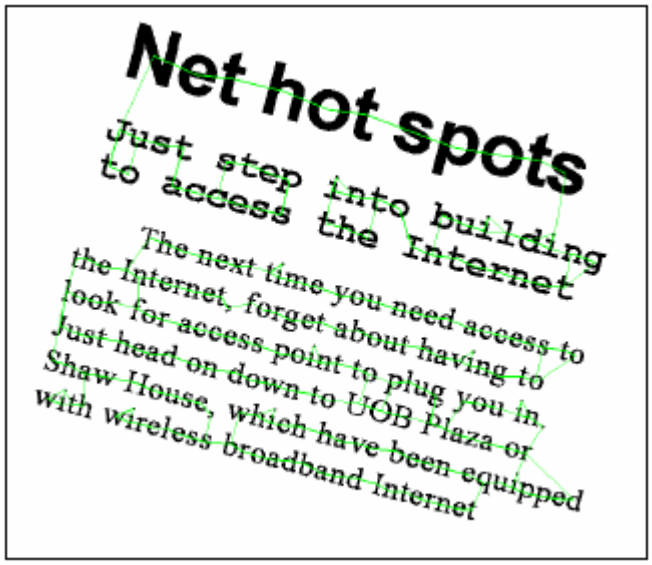


Figure 4.2 Two selected nearest neighbors with shortest distances

4.3 Features and Rules for Word Grouping

Now the task we are facing is how to know a remainder connection between the elements is within a word or between words. To solve this problem, we employ four features defined as follows. Suppose the element e_k has two most nearest neighbors. They are e_f with the most shortest distance d_{kf} , and e_s with the second most shortest distance d_{ks}. The characteristics of them are as follows: the heights, widths and areas of e_k, e_f , e_s are (h_k, w_k, a_k), (h_f , w_f , a_f) and (h_s, w_s, a_s), respectively.

The four features are defined as:

$$f_1 = \frac{d_{kf}}{\min\left(\left(h_k + w_k\right)/2, \left(h_f + w_f\right)/2\right)} \qquad (4.3)$$

It is the normalized distance from e_k to its most nearest neighbor e_f. The intuition of defining this feature is that, although the majority of the pairs (e_k, e_f) are indeed within same words, however, exceptions do exist. One common exception is in single-character words like the indefinite article "a". Obviously, the single-letter word, "a", should not be grouped with characters. Therefore, we define this feature f_1 and set a threshold T_{f1}, as shown in Fig. If $f_1 < T_{f1}$, then e_k and e_f should be grouped.

The use of MIN value at the denominator is in accordance with the definition of d_{ij} as in formula (4.2) where we will see that d_{ij} is based on the minimum distance between d (l_k , e_i) and d (l_k , e_j). Thus in the case of feature f_1, we should choose the corresponding MIN value, because d_{kf} in the equation of f_1 is based on the minimum distance to e_k or to e_f. If e_k is closer, then we should choose the height and width of e_k to normalize, and the same argument goes if e_f is closer.

$$f_2 = \frac{d_{ks}}{\min\left(\left(h_k + w_k\right)/2, \left(h_s + w_s\right)/2\right)} \qquad (4.4)$$

Feature f_2 is the normalized distance from e_k to its second most nearest neighbor e_s. The intuition of defining this feature is as follows: In real time, most of the pairs (e_k, e_s) are within same words. However, some (e_k, e_s) pairs are indeed from different words. For example, the last character of a word and the first character of its succeeding word generate such a special (e_k, e_s) pair. Therefore, we define feature f_2, together with feature f_3 (will be introduced very soon) to avoid this kind of pairs to be merged.

Here, we use MIN value at the denominator by the similar reason of f_1 case.

$$f_3 = \frac{d_{ks} - d_{kf}}{d_{ks}} \qquad (4.5)$$

Feature f_3 is the normalized difference between the distances d_{ks} and d_{kf}. If a character is located in the middle of a word (i.e. neither the first character nor the last character), the value of its f_3 feature will be small. Otherwise, the value will be large. We combine features f_2 and f_3 to produce another rule: If f_2 is less than threshold value T_{f2} and f_3 is less than threshold value T_{f3}, then e_k and e_s are grouped, and naturally e_k and e_f should be grouped as well.

To investigate the features f_1, f_2 and f_3, 100 real document images with varying character fonts, sizes and different inter-character, inter-word and inter-textline gaps, are utilized to obtain the statistical characteristics of them. The statistical results of occurrence frequencies are demonstrated in Figure 4.3 to 4.5, respectively.

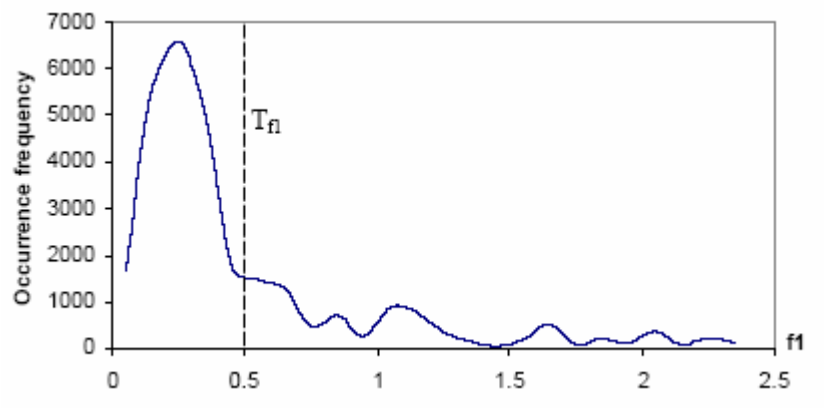


Figure 4.3 Occurrence frequency vs. f1

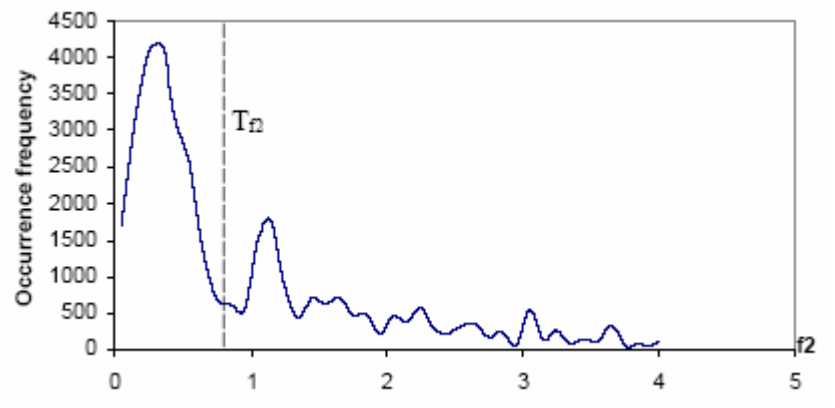


Figure 4.4 Occurrence frequency vs. f2

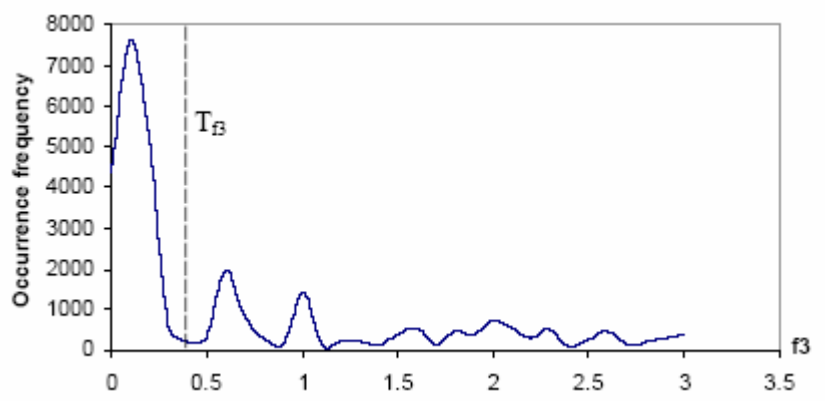


Figure 4.5 Occurrence frequency vs. f3

Then, two criteria can be summarized as follows:

Rule 1: if $f_1 < T_{f1}$, then e_k and e_f are grouped.

Rule 2: if $f_2 < T_{f2}$ and $f_3 < T_{f3}$, then e_k, e_f and e_s are grouped.

Based on the above process, there are two problems left. One is that many dots of characters ‘i’ and ‘j’ are not grouped to the corresponding words. The other is that some punctuation marks are erroneously grouped with words. From the properties of the

elements, it is difficult to distinguish the dots on the characters ‘i’ and ‘j’ from the punctuation like commas and full stops. Anyway, they have the same characteristic of relative small areas. We therefore employ Feature f_4, the area ratio, to identify them from others. If f4 of an element is less than a predefined threshold T_{f4} (set 0.25 empirically), it undergoes a further process then.

$$f_4 = \frac{a_k}{a_f} \qquad (4.6)$$

Our observations find that the two nearest neighbors of the dots on ‘i’ and ‘j’ come from the same word in general. On the other hand, for an element of punctuation, its most nearest neighbor is generally the last character of its preceding word, whereas its second nearest neighbor is the first character of its succeeding word. As a result, its f_3 normally has a larger value. We then utilize the following criteria to estimate them:

Rule 3: if $f_3 < T_{f3}$ and $f_4 < T_{f4}$, then e_k and e_f are grouped.

Rule 4: if $f_2 > T_{f2}$, $f_3 > T_{f3}$ and $f_4 < T_{f4}$, then e_k cannot be grouped with e_f.

4.4 Processing Punctuations

Generally speaking, commas and full stops are commonly used punctuation marks in documents. Rule 4 can effectively detect them. It is also worth noting that, some special

symbols such as dash (‘-’), tilde (‘~’), and various kinds of parentheses ‘{’, ‘}’, ‘[’, ‘]’, ‘(’, ‘)’, should be detected and excluded from word grouping. Punctuations should be excluded from word grouping.

Normally the punctuations come after a word. And these punctuations are very close to the words before them. So when we merge the blocks that are close to each other, we will merge the punctuations to be a part of the word. This is the reason why there is a need to remove the punctuations from the character blocks.

At the current state, we can only handle the comma “,” and the full stop “.”. For other punctuations like “! ? ”, our system can not handle them. Luckily the most normal punctuations in documents are commas and full stops. To handle them, we need to study the property of comma and full stop. For each comma or full stop, it is considered as a connected component on the area Voronoi tessellation. By observation we can always find a Voronoi edge which is in between of punctuation and the neighboring character.

Easily we can notice that these two punctuations have a significant position property. That is, they are at the right bottom corner of the previous character block. So from this property, the following judgment is used to check whether a character block is actually a comma or a full stop:

If each block has attributes *top, bottom, left, right and area*, and the following table names all the attributes of punctuation block and its neighboring character.

	top	bottom	left	right	area
punctuation	T_p	B_p	L_p	R_p	A_p
character	T_c	B_c	L_c	R_c	A_c

If all the following conditions are satisfied, it is regarded as punctuation:

1. $4 \times A_p < A_c$
2. $T_p < B_c$
3. $T_p > T_c + 0.3 \times (B_c - T_c)$
4. $L_p > R_c$

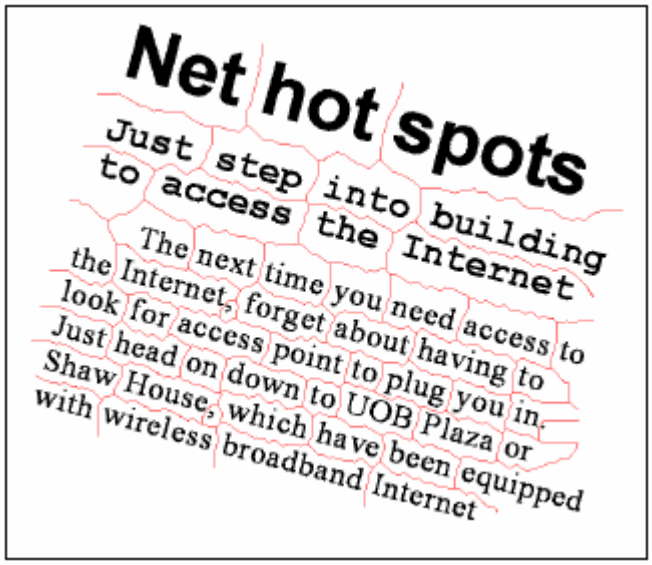
The first condition is to make sure the punctuation block is much smaller than its neighboring block. The second condition reflects the fact that the top of punctuation is above to the bottom of the character. The third condition is to ensure the top of punctuation is lower than that of the character. Then the last condition is to make sure the punctuation is on the right hand side of the character.

If the above conditions are satisfied, then the block will be considered to be a comma or a full stop and then it will be prevented from merging with its neighboring character.

Figure 4.6(a) shows the processing results of Figure 4.2, and Figure 4.6(b) gives the

corresponding Voronoi edges separating the grouped words.

(a)Neighbors within words



(b)Voronoi edges among words

Figure 4.6 Word grouping results of Figure 4.2.

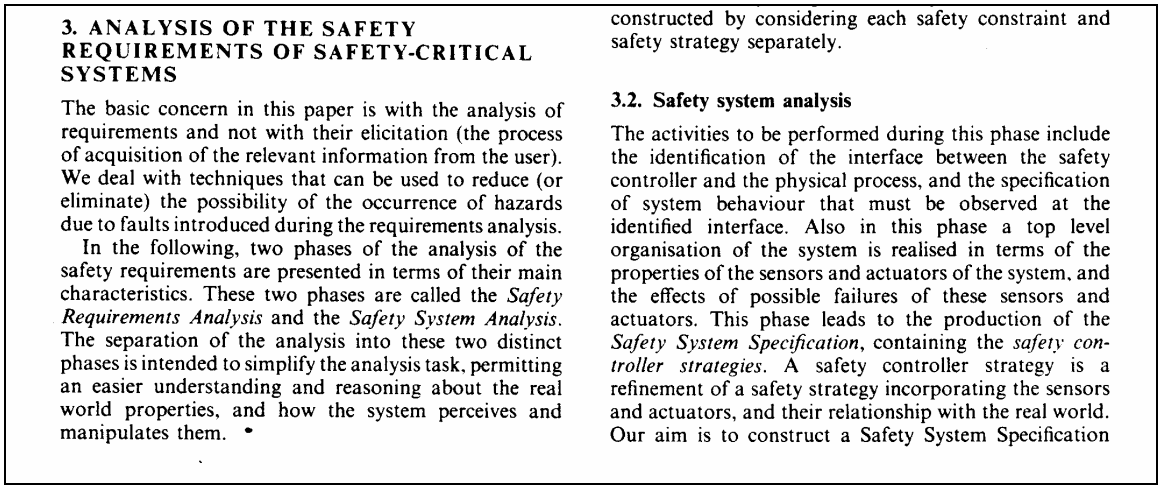
Chapter 5:

Experimental Results and Discussion

In this chapter, we first show two sample documents to illustrate the operation of the system, and then we will discuss the performance of the system and also analyze the erroneous cases.

5.1 Testing Results

To evaluate the performance of the proposed method, the document image, from different sources, with various characters of sizes and fonts, are used for the test. They are 100 page document images selected from the UW document image database, and the images provided by the Digital Library of our university including 328 document images of scanned books and 476 document images of scanned outdated student theses. Each document has more than 200 characters and is scanned as black and white image. Here we will show two of the tests. The input document in Figure 5.1 contains words of different fonts and sizes. The input document in Figure 5.2 is an article with tilted text lines. We will use these two images to illustrate the area Voronoi tessellation generation and word grouping done by the system.



3. ANALYSIS OF THE SAFETY REQUIREMENTS OF SAFETY-CRITICAL SYSTEMS

The basic concern in this paper is with the analysis of requirements and not with their elicitation (the process of acquisition of the relevant information from the user). We deal with techniques that can be used to reduce (or eliminate) the possibility of the occurrence of hazards due to faults introduced during the requirements analysis.

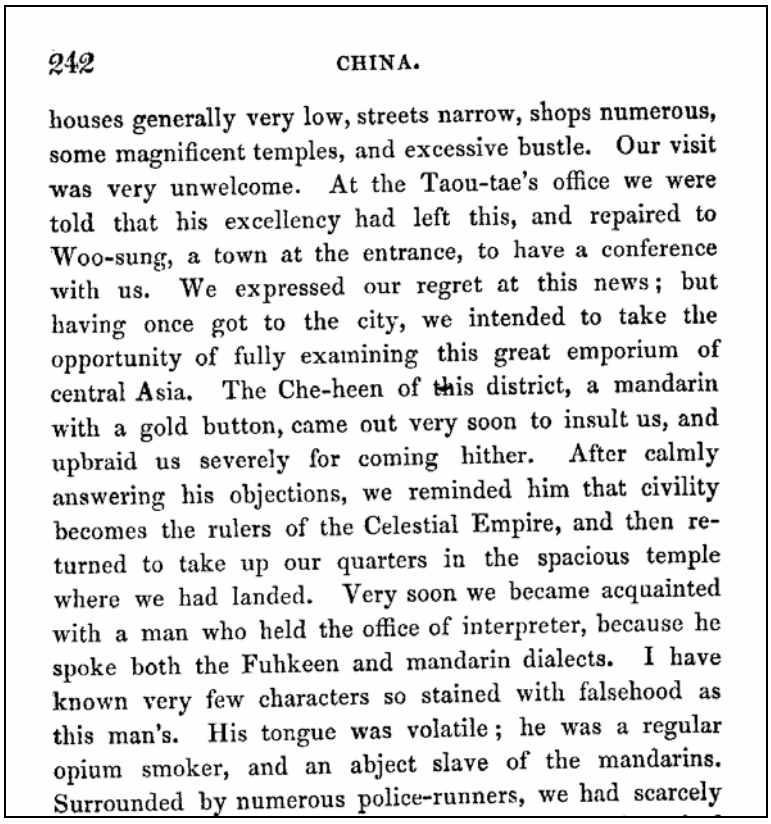
In the following, two phases of the analysis of the safety requirements are presented in terms of their main characteristics. These two phases are called the *Safety Requirements Analysis* and the *Safety System Analysis*. The separation of the analysis into these two distinct phases is intended to simplify the analysis task, permitting an easier understanding and reasoning about the real world properties, and how the system perceives and manipulates them.

constructed by considering each safety constraint and safety strategy separately.

3.2. Safety system analysis

The activities to be performed during this phase include the identification of the interface between the safety controller and the physical process, and the specification of system behaviour that must be observed at the identified interface. Also in this phase a top level organisation of the system is realised in terms of the properties of the sensors and actuators of the system, and the effects of possible failures of these sensors and actuators. This phase leads to the production of the *Safety System Specification*, containing the *safety controller strategies*. A safety controller strategy is a refinement of a safety strategy incorporating the sensors and actuators, and their relationship with the real world. Our aim is to construct a Safety System Specification

Figure 5.1 Input Document 1 (UW document image)

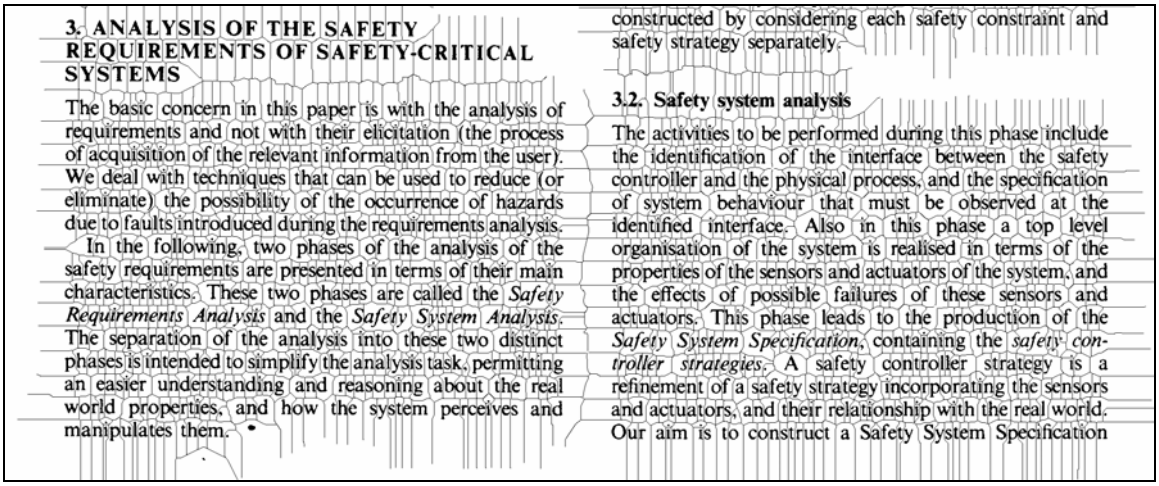


242 CHINA.

houses generally very low, streets narrow, shops numerous, some magnificent temples, and excessive bustle. Our visit was very unwelcome. At the Taou-tae's office we were told that his excellency had left this, and repaired to Woo-sung, a town at the entrance, to have a conference with us. We expressed our regret at this news; but having once got to the city, we intended to take the opportunity of fully examining this great emporium of central Asia. The Che-heen of this district, a mandarin with a gold button, came out very soon to insult us, and upbraid us severely for coming hither. After calmly answering his objections, we reminded him that civility becomes the rulers of the Celestial Empire, and then returned to take up our quarters in the spacious temple where we had landed. Very soon we became acquainted with a man who held the office of interpreter, because he spoke both the Fuhkeen and mandarin dialects. I have known very few characters so stained with falsehood as this man's. His tongue was volatile; he was a regular opium smoker, and an abject slave of the mandarins. Surrounded by numerous police-runners, we had scarcely

Figure 5.2 Input Document 2 (Scanned Book)

After finding connected components, noise removal and special symbol detection, we managed to construct two area Voronoi tessellations using blocks as generators. They are shown in Figure 5.3 and Figure 5.4.



3. ANALYSIS OF THE SAFETY REQUIREMENTS OF SAFETY-CRITICAL SYSTEMS

The basic concern in this paper is with the analysis of requirements and not with their elicitation (the process of acquisition of the relevant information from the user). We deal with techniques that can be used to reduce (or eliminate) the possibility of the occurrence of hazards due to faults introduced during the requirements analysis.

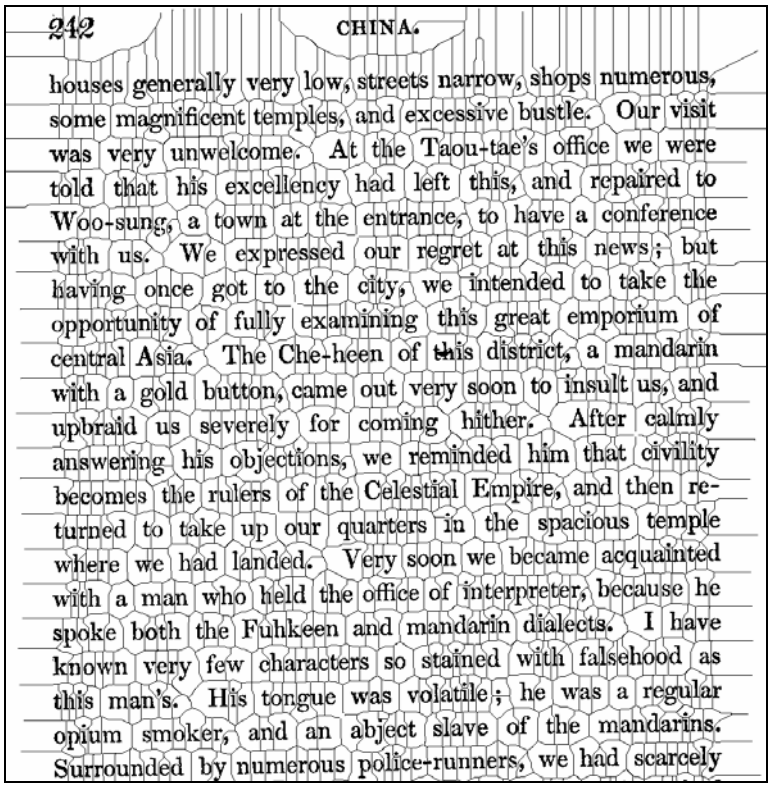
In the following, two phases of the analysis of the safety requirements are presented in terms of their main characteristics. These two phases are called the *Safety Requirements Analysis* and the *Safety System Analysis*. The separation of the analysis into these two distinct phases is intended to simplify the analysis task, permitting an easier understanding and reasoning about the real world properties, and how the system perceives and manipulates them.

constructed by considering each safety constraint and safety strategy separately.

3.2. Safety system analysis

The activities to be performed during this phase include the identification of the interface between the safety controller and the physical process, and the specification of system behaviour that must be observed at the identified interface. Also in this phase a top level organisation of the system is realised in terms of the properties of the sensors and actuators of the system, and the effects of possible failures of these sensors and actuators. This phase leads to the production of the *Safety System Specification*, containing the *safety controller strategies*. A safety controller strategy is a refinement of a safety strategy incorporating the sensors and actuators, and their relationship with the real world. Our aim is to construct a Safety System Specification

Figure 5.3 The Area Voronoi tessellation of Document 1

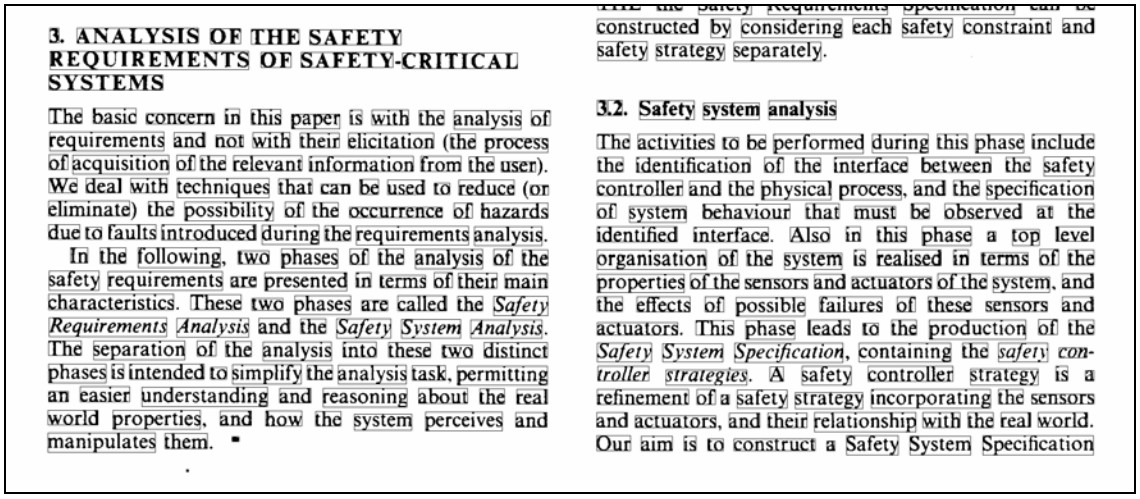


242 CHINA.

houses generally very low, streets narrow, shops numerous, some magnificent temples, and excessive bustle. Our visit was very unwelcome. At the Taou-tae's office we were told that his excellency had left this, and repaired to Woo-sung, a town at the entrance, to have a conference with us. We expressed our regret at this news; but having once got to the city, we intended to take the opportunity of fully examining this great emporium of central Asia. The Che-heen of this district, a mandarin with a gold button, came out very soon to insult us, and upbraid us severely for coming hither. After calmly answering his objections, we reminded him that civility becomes the rulers of the Celestial Empire, and then returned to take up our quarters in the spacious temple where we had landed. Very soon we became acquainted with a man who held the office of interpreter, because he spoke both the Fuhkeen and mandarin dialects. I have known very few characters so stained with falsehood as this man's. His tongue was volatile; he was a regular opium smoker, and an abject slave of the mandarins. Surrounded by numerous police-runners, we had scarcely

Figure 5.4 The Area Voronoi tessellation of Document 2

Figure 5.5 and 5.6 show the final output documents. The words are identified using rectangles. We notice that in Figure 5.5, almost all the words are correctedly extracted although they are in tilted text lines; in Figure 5.6, the words in title and text body are different in font, size and inter-character distance, but they are both successfully extracted.



3. ANALYSIS OF THE SAFETY REQUIREMENTS OF SAFETY-CRITICAL SYSTEMS

The basic concern in this paper is with the analysis of requirements and not with their elicitation (the process of acquisition of the relevant information from the user). We deal with techniques that can be used to reduce (or eliminate) the possibility of the occurrence of hazards due to faults introduced during the requirements analysis.

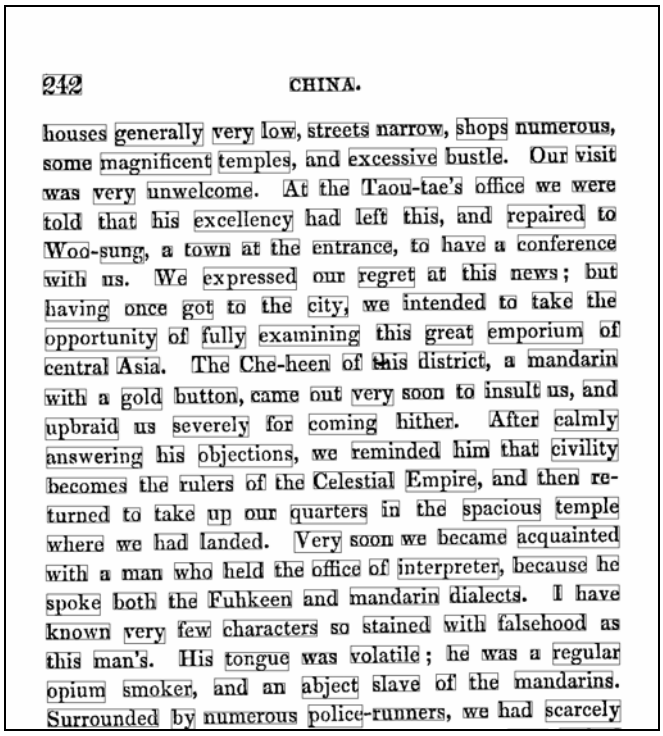
In the following, two phases of the analysis of the safety requirements are presented in terms of their main characteristics. These two phases are called the *Safety Requirements Analysis* and the *Safety System Analysis*. The separation of the analysis into these two distinct phases is intended to simplify the analysis task, permitting an easier understanding and reasoning about the real world properties, and how the system perceives and manipulates them. ▪

constructed by considering each safety constraint and safety strategy separately.

3.2. Safety system analysis

The activities to be performed during this phase include the identification of the interface between the safety controller and the physical process, and the specification of system behaviour that must be observed at the identified interface. Also in this phase a top level organisation of the system is realised in terms of the properties of the sensors and actuators of the system, and the effects of possible failures of these sensors and actuators. This phase leads to the production of the *Safety System Specification*, containing the *safety controller strategies*. A safety controller strategy is a refinement of a safety strategy incorporating the sensors and actuators, and their relationship with the real world. Our aim is to construct a Safety System Specification

Figure 5.5 Word grouping Result of Document 1



242 CHINA.

houses generally very low, streets narrow, shops numerous, some magnificent temples, and excessive bustle. Our visit was very unwelcome. At the Taou-tae's office we were told that his excellency had left this, and repaired to Woo-sung, a town at the entrance, to have a conference with us. We expressed our regret at this news; but having once got to the city, we intended to take the opportunity of fully examining this great emporium of central Asia. The Che-heen of this district, a mandarin with a gold button, came out very soon to insult us, and upbraid us severely for coming hither. After calmly answering his objections, we reminded him that civility becomes the rulers of the Celestial Empire, and then returned to take up our quarters in the spacious temple where we had landed. Very soon we became acquainted with a man who held the office of interpreter, because he spoke both the Fuhkeen and mandarin dialects. I have known very few characters so stained with falsehood as this man's. His tongue was volatile; he was a regular opium smoker, and an abject slave of the mandarins. Surrounded by numerous police-runners, we had scarcely

Figure 5.6 Word grouping Result of Document 2

5.2 Performance of the System

To evaluate the performance of the system, we look at two measures: the accuracy of word grouping, and the computation time.

The word grouping performance on the test documents is tabulated in Table 5.1, including the rates of correction, splitting(fragmentation) and over-merging. An encouraging accuracy of over 99% has been achieved.

	UW image	NUS scanned books	NUS scanned theses	Average
ACCURACY (%)	98.83	99.06	99.26	99.05
SPLITTING (%)	0.56	0.28	0.43	0.42
OVER- MERGING (%)	0.61	0.66	0.31	0.53

Table 5.1Performance of word grouping

Let us turn to the consideration about the processing speed of our system. As we mentioned, the task of word grouping is split into several subtasks, such as generating connected components, constructing area Voronois diagram, etc. The average computation time required at each subtask (using input images of size 1280×2056) is listed in Table 5.2. In case the input image is larger (contains more pixels), our system needs more time to process it.

Generating connected components	**3.00 sec. (54.5%)**
Construction of the area Voronoi tessellation	**2.10 sec. (38.2%)**
Word grouping process	**0.30 sec. (5.5%)**
Others (file I/O etc.)	**0.10 sec. (1.8%)**
Total	**5.50 sec.**

Table 5.2 Computation time

We observe that the first two tasks (generating connected components and the process of constructing area Voronoi tessellation) dominate the overall computation time. Especially for the first task, it consumes more than half of the total time. Thus, a good algorithm for connected component generation will highly raise the efficiency of our system. Another important point is that the time consumed by the task of edge selection is quite small as compared with the first two tasks. This is because the method selects edges based on the local examination.

5.3 Error Analysis

The above experimental results have proven that our system has the ability to correctly extract word objects from document images with a very high accuracy.

There are however still some constrains that cannot be handled by our system. The following examples illustrate the commonly occurred errors:

Firstly, our system is not able to handle the punctuations other than comma and full stop. Refer to Figure 5.7, after generating connected components, we can observe from Figure 5.7 (a) that the colon ":" and comma "," are both recognized as connected components (blocks). The word grouping result of this text line is shown in Figure 5.7 (b). Notice that punctuation detection (described in 4.2.2) helps us to effectively exclude commas from any word objects. However, we are unable to prevent colon to be considered as part of the word "Nigel".

The same kind error also happens to other punctuations such as question mark "?", exclamation mark "!", semicolon ";", and quotation mark """. Fortunately, the frequency of the above punctuation appearing in the document images is really low, in another word, most of the punctuations we met are either comma or full stop which can be detected by our system, therefore the word grouping accuracy of our system will not be dropped down much by this kind of error.

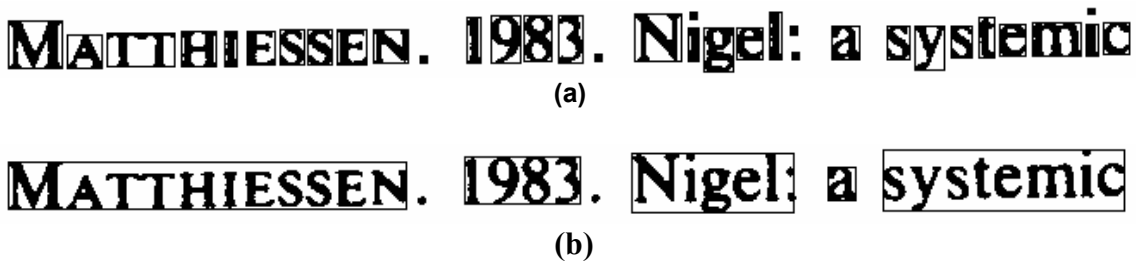


Figure 5.7 Error Case A

The second kind of error is due to wrongly connected components. Our system is based on the area Voronoi tessellation, and the diagram is generated using connected components as generators. Imagine that if there is a mistake of connected components, it will directly

affect the correctness of the area Voronoi tessellation, which may result errors of word grouping. An erroneous case is shown in Figure 5.8: in Figure 5.8 (a), the comma is connected with the letter “a” in front of it, and they two are recognized as one connected component. In this case, we cannot expect the punctuation detection process of our system to handle the comma, because the precondition for detecting punctuation is the punctuation must lie in an individual block. Hence, the word grouping result shown in Figure 5.8 (b) presents a wrongly extracted word “KPa,”.

(a)

(b)

Figure 5.8 Error Case B

Chapter 6:

Conclusion

6.1 Conclusion of the Project

A Voronoi neighborhood based algorithm for grouping word objects in document images is presented in this thesis. The Voronoi neighborhoods are generated from the Voronoi tessellation for word grouping, with the information about the relations and distances of neighboring connected components. The experimental results on various document images have shown that the proposed approach has achieved promising results with a high accuracy, and is robust to various font types, styles, sizes, skew angles, as well as text orientations.

6.2 Advantage of Our Method and Comparison with Others

Voronoi tessellation is an effective tool for representing the neighboring relations among elements in a digital image. With the help of Voronoi tessellation, our method outperforms the existing word grouping methods in the following aspects:

6.2.1 Grouping Accuracy

Please refer to Table 5.1, our method has achieved a very encouraging accuracy of over 99 percent, which is one of the highest among all the current word grouping method. Table 6.1 shows the comparison of perfomances between our method's and Wang etal's method [16], which uses a statistical-based word segmentation algorithm.

	Splitting Error	Over-Merging Error	Accuracy
METHOD OF [16]	0.92%	1.47%	97.43%
OUR METHOD	0.42%	0.53%	99.05%

Table 6.1 Comparison of Performances

The data implies that our method is advantageous over the method of [16] in all the three comparisons with less splitting error, less over-merging error and better accuracy.

6.2.2 Character Size and Spacing

Some existing methods are not capable of dealing with the documents with various sizes of characters. For example, in [4] Fletcher and Kasturi described a method in which Hough transform was used to group characters into words. Hough transform is applied to the centroids of the rectangles enclosing each connected component; the collinear connected components are thus located. The positional relationships between the collinear connected components are then examined to locate the words. This method cannot handle different character sizes, because the process depends on the average height of all connected components. Another example of this case happens in [7]. Jain and

Bhattacharjee considered text image as textured objects and used Gabor filtering for text analysis. This method is also sensitive to font sizes and styles, and it is generally time-consuming

In contrast, our method is robust to various character sizes and character spacing. Figure 6.1 gives an example of word grouping results carried on an image of scanned books. It has been shown that our proposed method is insensitive to the character sizes and fonts.

NUS The ALUMNI

Mad about Sports

by CHUA CHONG JIN

Heartwarming tale

Covering a sports story for an international organisation in Colombo, Sri Lanka, recently, I had such a thrill watching the people there get all excited over their number one track star — Olympic Games bronze medallist Susanthika Jayasinghe.

I can still hear the bounce on the people's feet and feel the air of anticipation as they rushed to the Sugathadasa Stadium from their homes, schools and offices to catch her in action for the 200m final of the 14th Asian Athletic Championships on 12 August 2002. Then there was the 4 x 400m final. The roar of approval as the Sri Lankan guys led by the immensely talented Sugath Tillekeratne, powered their way to gold was so overwhelmingly united.

Hear the hammer thrower

Funny how some of life's best lessons come from the least expected sources, and in the most unusual settings too. Interviewing Japan's Koji Murofushi, silver medalist for the hammer throw at the 2001 World Championships in Edmonton, would — I thought, narrowly — require little thought. I mean, what is there in the hammer throw except for brute strength and perhaps good co-ordination?

I ended up listening to some rather deep insights into, among other things, the beauty of movements which aren't experienced in man's day-to-day routines and the purity, however fleeting, of even the worst man when he takes on the public role of avid sports fan. In a van taking us from Sugathadasa Stadium to a nearby gym, Koji, whose dad is Japanese and mum Romanian, was clearly at his philosophical best.

Thorpe & Du Toit

At this very moment, I am mad about sports. But it's more the human interest stories of athletes that excite me. I know this from my very different responses now compared to four years ago regarding what is arguably one of sports' biggest stories — that of Australian swimmer Ian Thorpe. Luck brought me standing just beside him — without a crowd — at Bukit Jalil in KL where he was taking part in the 1998 Commonwealth Games. I was thrilled.

Covering the 2002 Commonwealth Games in Manchester, I was bored with the media madness surrounding him. Other athletes, I thought, had their talent constantly insulted. For the media was always asking them only about what they thought of Thorpe and his feats — and yes, feet too.

Which was why, in 2002, my choice of top athlete was South African swimmer Natalie Du Toit though I still love watching Thorpe. The 18-year-old from Cape Town had her lower left leg amputated following a motorcycle accident in 2001. Yet she won gold in the 50m and 100m Elite Athlete with a Disability (EAD) freestyle swimming events in Manchester — and even made the final of the 800m open race.

Figure 6.1 Example of word grouping

This is because we do not utilize any local information in our method, such as average height of all connected components, average area of all connected components or average

inter-character spacing. Voronoi Tessellation provides us a precise neighborhood relation among all elements, despite the difference of character sizes.

6.2.3 Skewed Documents

Due to manual mistakes during image scanning, quite a number of imaged documents stored in the digital libraries are skewed. Therefore the ability of dealing with them is essential for an effective word grouping method. However, not all the existing methods are good at handling skewed documents. In [13], Sobottka proposed an approach to automatically extract text from colored books and journal covers, but when dealing with skewed document, the performance of this method deteriorates a lot.

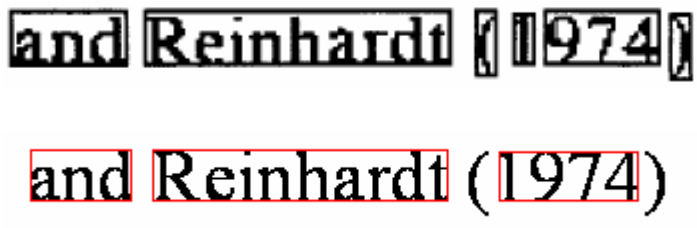
Our method is excellent at handling skewed document, because the skewed text lines have very little impact on the Voronoi Tessellation, and consequently will not affect much the accuracy of word grouping.

6.2.4 Resilience to Noise

Our method is equipped with the capability of detecting and handling many kinds of noise in the imaged document. This ability is explicitly described in Chapter 2. In general, those blocks that are very high are deleted; for any two blocks, if one block is in the other, the smaller block is deleted; if two blocks partially overlap with each other, we merge them two into one block; two types of special symbols.--- elongated horizontally such as dash

('-') or tilde ('~') or vertically elongated symbols such as various kinds of parentheses '{', '}', '[', ']', '(', or ')' are detected using a ruled-based algorithm without involving the recognition of any symbols, and then eliminated from the set of elements which are for future grouping into word objects.

Figure 6.2 shows comparison of resilience to noise between our method and the method of [16]. Figure 6.2 (a) is a case that the algorithm of [16] fails, due to the incapability of handling parenthesis characters, which are '(' and ')'in this example. In contrast, our method correctly detected the noises '(' and ')' as shown in Figure 6.2 (b)".



(a)

(b)

Figure 6.2 Comparison of resilience to noise

6.2.5 Computing Time

Computing time is also an important issue for word grouping. An excellent word grouping method should have not only high accuracy but also a fast processing speed. Our method does well for this aspect; with an average computing time of 5.5 seconds given that the resolution of input imaged document is 1280×2056 pixels.

Taking [13] as an example, it utilizes top-down and bottom-up analysis, and results in 9.72 seconds mean time for processing each input image of 1300×1800 pixels.

Another two examples are [4] and [7]. Fletcher and Kasturi in [4] utilized Hough transform; in [7] Jain and Bhattacharjee made use of Gabor filter for text analysis. Both Hough transform and Gabor filter are known to be computationally costly and therefore will be slower than our system.

References

[1] Bapst, F. and Ingold, R. (1998). ***Using Typography in Document Image Analysis.*** In Proceedings of the 7th International Conference on Electronic Publishing, Held Jointly with the 4th International Conference on Raster Imaging and Digital Typography, St. Malo, France, March 30 - April 3, 1998, pp. 240-251.

[2] Burge, M., and Monagan, G. (1995) ***Using the Voronoi Tessellation for Grouping Words and Multipart Symbols in Documents.*** Proceedings of SPIE International Symposium on Optics. Imaging and Instrumentation, Vol.2573, San Diego, California, pp.116-124.

[3] Chen, S., Haralick, R.M. and Phillips, I. (1995). ***Simultaneous Word Segmentation From Document Images Using Recursive Morphological Closing Transform.*** In Proceedings of the 3rd ICDAR, Montreal, Canada, August 14-16, 1995, pp. 761–764.

[4] Fletcher, L.A. and Kasturi, R. (1988). ***A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images.*** IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 10, No. 6, November 1988, pp. 910-918.

[5] Goto, H. and Aso, H. (1999). ***Extracting Curved Text Lines Using Local Linearity of Text Line.*** International Journal on Document Analysis and Recognition (IJDAR), Vol.2, No.2/3, 1999, pp.111-119.

[6] Ittner, D.J. and Baird, H.S. (1993). ***Language-free Layout Analysis.*** In Proceedings of the 2nd ICDAR, Tsukuba, Japan, October 20-22, 1993, pp. 336–340.

[7] Jain, A.and Bhattacharjee, S. (1992). ***Text Segmentation Using Gabor Filters for Automatic Document Processing.*** Machine Vison Applications, Vol.5, pp.169-184.

[8] Kim, S.H., Jeong, C.B., Kwag, H.K. and Suen, C.Y. (2002). ***Word Segmentation of Printed Text Lines Based on Gap Clustering and Special Symbol Detection.*** In Proceedings of the International Conference on Pattern Recognition (ICPR02), Quebec City, Canada, Aug. 11-15, 2002, pp. 320-323.

[9] Kise, K., Iwata, M., Dengel, A. and Matsumoto, K. (1998) ***Text-Line Extraction as Selection of Paths in the Neighbor Graph.*** Document Analysis Systems, pp.225-239.

[10] Kise, K., Sato, A. and Iwata, M. (1998). ***Segmentation of Page Images Using the Area Voronoi tessellation.*** Computer Vision and Image Understanding, Vol. 70, No. 3, June 1998, pp. 370-382.

[11] Okabe, A., Boots, B. and Sugihara, K. (1992). ***Spatial Tessellations: Concepts and Applications of Voronoi tessellations.*** First Edition, John Wiley & Sons, 1992.

[12] Park, H.C., Ok, S. Y., Yu, Y.J. and Cho, H.G. (2001). ***A Word grouping Algorithm for Machine-printed Documents Using a 3D Neighborhood Graph Model.*** International Journal on Document Analysis and Recognition (IJDAR), Vol.4, No.2, 2001, pp. 115-130.

[13] Sobottka, K., Kronenberg, H., Perroud, T. and Bunke, H. (2000). ***Text Extraction from Colored Book and Journal Covers.*** International Journal on Document Analysis and Recognition (IJDAR), Vol.2, No.4, 2000, pp.163 - 176.

[14] Tan, C.L. and Ng, P.O. (1998). ***Text Extraction Using Pyramid.*** Pattern Recognition, Vol.31, No.1, 1998, pp. 63-72.

[15] Wang, Y., Phillips, I.T. and Haralick, R. (2001). ***Using Area Voronoi Tessellation to Segment Characters Connected to Graphics.*** In Proceedings of the Fourth IAPR International Workshop on Graphics Recognition (GREC2001), Kingston, Ontario, Canada, September 7-8, 2001.

[16] Wang, Y., Phillips, I.T. and Haralick, R. (2000). ***Statistical-Based Approach to Word Segmentation.*** In Proceedings of the International Conference on Pattern Recognition (ICPR00), Barcelona, Spain, September 3 - 8, 2000, pp. 4555-4558.

[17] Xiao, Y. and Yan, H. (2003). ***Text Region Extraction in a Document Image Based on the Delaunay Tessellation.*** Pattern Recognition, Vol. 36, No. 3, 2003, pp. 799-809.

Publication

1. Yue Lu, Zhe Wang and Chew Lim Tan, Word Grouping in Document Images Based on Voronoi Tessellation, International Workshop on Document Analysis Systems (DAS 2004).

Appendix – Program List

This appendix contains a summary of the files that make up our DIAR (Document Image Analysis and Recognition) application system.

- ***DIAR.dsp***

This file (the project file) contains information at the project level and is used to build a single project or subproject.

- ***DIAR.h***

This is the main header file for the application. It includes other project specific headers (including Resource.h) and declares the CDIARApp application class.

- ***DIAR.cpp***

This is the main application source file that contains the application class CDIARApp.

- ***DIAR.rc***

This is a listing of all of the Microsoft Windows resources that the program uses. It includes the icons, bitmaps, and cursors that are stored in the RES subdirectory.

- ***DIAR.clw***

This file contains information used by ClassWizard to edit existing classes or add new classes. ClassWizard also uses this file to store information needed to create and edit message maps and dialog data maps and to create prototype member functions.

- ***res\DIAR.ico***

This is an icon file, which is used as the application's icon. This icon is included by the main resource file DIAR.rc.

- ***res\DIAR.rc2***

This file contains resources that are not edited by Microsoft Visual C++.

- ***MainFrm.h, MainFrm.cpp***

These files contain the frame class CMainFrame, which is derived from CMDIFrameWnd and controls all MDI frame features.

- ***ChildFrm.h, ChildFrm.cpp***

These files define and implement the CChildFrame class, which supports the child windows in an MDI application.

- ***res\Toolbar.bmp***

This bitmap file is used to create tiled images for the toolbar.

- ***DIARDoc.h, DIARDoc.cpp***

These files contain our CDIARDoc class to implement file saving and loading (via CDIARDoc::Serialize).

- ***DIARView.h, DIARView.cpp***

These files contain our CDIARView class. CDIARView objects are used to view CDIARDoc objects.

- ***StdAfx.h, StdAfx.cpp***

These files are used to build a precompiled header (PCH) file named DIAR.pch and a precompiled file named StdAfx.obj.

- ***Resource.h***

This is the standard header file, which defines new resource IDs. Microsoft Visual C++ reads and updates this file.

- ***WangProcess1.cpp***

This file is to generate connected components of the input document image.

- ***WangProcess2.cpp***

This file is used to perform various tasks for word grouping, including image preprocessing, area Voronoi edge generation, edge selection and merging characters into word objects.