A WORD IMAGE CODING TECHNIQUE AND ITS APPLICATIONS IN INFORMATION RETRIEVAL FROM IMAGED DOCUMENTS

ZHANG LI

(B.Sc. (Hons), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF SCIENCE

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2004

Acknowledgements

It is a great pleasure to render my sincere appreciation to all those people that have generously offered their invaluable help and assistance in completing this research work.

First of all, I would like to thank Associate Professor Tan Chew Lim, for his ingenious supervision and guidance during the whole year of my master study; and also for his consistent encouragement and generous support in my research work.

I am also grateful to Dr. Lu Yue, who continuously provided his invaluable suggestions and guidance to this project work. It is my great pleasure to work with him and share his insights in document image retrieval area.

Last but not least, I would like to express my gratitude to Dr. Xiao Tao for sharing with me his knowledge in Wavelet Transformation as well as his ingenious idea in Pattern Recognition field.

Table of Contents

Acknowledgements **i**

Table of Contents **ii**

Summary **iv**

List of Tables **vi**

List of Figures **vii**

Chapter 1 Introduction **1**
1.1 Background 1
1.2 Scope and Contributions 5
1.3 Organization of the Thesis 9

Chapter 2 Feature Code File Generation **11**
2.1 Connected Component Analysis 11
2.2 Word Bounding 13
2.3 Skew Estimation 14
2.4 Skew Rectification 18
2.5 Word Bounding Box Regeneration 20
2.6 Italic Font Detection 21
2.7 Italic Font Rectification 22
2.8 Feature Code File Generation 22

Chapter 3 Word Image Coding **24**
3.1 LRPS Feature Representation 24
3.2 Ascender-and-descender Attribute 24
3.3 Line-or-traversal Attribute 25
3.3.1 Straight Stroke Line Feature 26
3.3.2 Traversal Feature 28
3.4 Post-processing 30
3.4.1 Merging Consecutive Identical Primitives 30
3.4.2 Refinement for Font Independence 31
3.5 Primitive String Token for Standard Characters 33
3.6 Verification 34

Chapter 4 Italic Font Recognition **36**
4.1 Background of Font Recognition 36
4.2 Wavelet Transformation Based Approach 38
4.2.1 Wavelet Decomposition of Word Images 39
4.2.1.1 Pyramid Transform 39
4.2.1.2 Coupled and uncoupled Wavelet Decomposition 40

4.2.2 Statistical Analysis of Stroke Patterns 43
4.2.2.1 Vertical Stroke Analysis 44
4.2.2.2 Diagonal Stroke Analysis 45
4.2.3 Experimental Results 46

Chapter 5 Feature Code Matching 48
5.1 Coarse Matching 48
5.2 Inexact String Matching 49

Chapter 6 Web-based Document Image Retrieval System 56
6.1 System Overview 56
6.2 System Implementation 58
6.3 AND/OR/NOT Operations 60
6.3.1 AND Operation 61
6.3.2 OR Operation 62
6.3.3 NOT Operation 64
6.4 System Evaluation 64

Chapter 7 Search Engine for Imaged Documents 69
7.1 Implementation 69
7.2 Performance Evaluation 71
7.3 Comparison with the Page Capture 73
7.4 Comparison with Hausdorff Distance Based Search Engine 74
7.4.1 Space Elimination and Scale Normalization 75
7.4.2 Word Matching Based on Hausdorff Distance 76

Chapter 8 Conclusions 79
8.1 Contributions 80
8.2 Future Works 81

Bibliography 83

Appendix A – How to Use the Web-based Retrieval System 87

Appendix B – How to Use the Search Engine 88

Summary

With an increasing amount of documents being scanned and archived in the form of digital images, Document Image Retrieval, as part of information retrieval paradigm, has been attracting a continuous attention among the Information Retrieval (IR) communities. Various retrieval techniques based on Optical Character Recognition (OCR) have been proposed and proved to achieve a good performance on high quality printing documents. However, many document image databases contain poor quality documents such as those ancient books and old newspaper in digital libraries. This draws the interest of many researchers in looking for an alternative approach to perform retrieval among distorted document images more effectively.

This thesis presents a word image coding technique that extracts features from each word object and represents them using a feature code string. On top of this, two applications are implemented. One is an experimental web-based retrieval system that efficiently retrieves document images from digital libraries given a set of query words. Some image preprocessing is first carried out off-line to extract word objects from those document images. Then, each word object is represented by a string of feature codes. Consequently, feature code file for each document image is generated containing a set of feature codes representing its word objects. Upon receiving a user's request, our system converts the query word into its feature code using the same conversion mechanism as is used in producing the feature codes for the underlying document images. Search is then performed among those feature code files generated off-line. An inexact string matching algorithm, with the ability of matching a word

portion, is applied to match the feature code of the query word with the feature codes in the feature code files. The occurrence frequency of the query word in each retrieved document image is calculated for relevant ranking. Second application is a search engine for imaged documents in PDF files. In particular, a plug-in is implemented in Acrobat Reader and performs all the preprocessing and matching procedures online when the user inputs a query word. The matching word objects will be identified and marked in the PDF files opened by the user either on a local machine or through a web link.

Both applications are implemented with the ability of handling skew images using a nearest neighbor based skew detection algorithm. Italic fonts are also identified and recognized with a wavelet transformation based approach. This approach takes advantage of 2-D wavelet decomposition and performs statistical stroke pattern analysis on wavelet decomposed sub-images to discriminate between normal and italic styles. A testing version of the search engine is implemented based on Hausdorff distance matching of word images. Experiments are conducted on scanned images of published papers and students' thesis provided by our digital libraries with different fonts and conditions. The results show that better recall and precision are achieved with the word image coding based search engine with less sensitivity towards noise affections and font variations. In addition, by storing the feature codes of the document image in an intermediate file when processing the first search, we need to perform the preprocessing steps only once and thus achieve a significant speed-up in the subsequent search process.

List of Tables

Table 3-1 Primitive properties vs. Character code representation 32

Table 3-2 Primitive string tokens of characters..... 34

Table 5-1 Scoring table and missing space recovery 55

Table 6-1 A snapshot of the index table storing information of queried words 60

List of Figures

Figure 1-1 System components 7

Figure 1-2 Search engine for imaged documents in PDF files 8

Figure 2-1 Connected components 12

Figure 2-2 Word bounding box 13

Figure 2-3 Nearest Neighbor Chains (NNCs) 14

Figure 2-4 Skew angle (a) $\Delta x > \Delta y$ (b) $\Delta x < \Delta y$ 15

Figure 2-5 NNCs for (1): (a) (d) K=2 (b) (e) K=3 (c) (f) K≥4 17

Figure 2-6 Nearest Neighbor Chain (NNC) 18

Figure 2-7 Skew rectification 20

Figure 2-8 A portion of a rectified page image 20

Figure 2-9 Italic word and its rectified image 22

Figure 2-10 Feature code file 23

Figure 3-1 Primitive string extraction 25

Figure 3-2 Refinement for LRPS representation to avoid the effect of serif 31

Figure 4-1 The pyramid decomposition scheme 40

Figure 4-2 One stage of the uncoupled wavelet decomposition scheme 41

Figure 4-3 Two dimensional Discrete Wavelet Decomposition 42

Figure 4-4 An example of one-level wavelet decomposed sub-images 43

Figure 4-5 (a)(b) VSLS running through the mid zone for normal and italic styles respectively (c)(d) CDS for normal and italic styles respectively (length ≥ 3).....45

Figure 4-6 Examples of wavelet decomposed vertical sub-images.....46

Figure 4-7 Recognition accuracy comparisons between traditional method and our method.....47

Figure 6-1 Overview of the web-based document image retrieval system 57

Figure 6-2 AND operation.....62

Figure 6-3 OR operation 63

Figure 6-4 NOT operation.....64

Figure 6-5 Recall and precision chart of the word image coding based system.....67

Figure 6-6 Search result for pre-queried word 67

Figure 6-7 Search result for first-time queried word.....68

Figure 7-1 Snapshot of the search engine embedded in Acrobat Reader 6.0 71

Figure 7-2 Search result for a query word located in an opened PDF document image 71

Figure 7-3 Performance vs. different thresholds 73

Figure 7-4 Recall and Precision wrt word length distribution and noise level.....73

Figure 7-5 Ascender, descender and mid zone of a word image 77

Figure 7-6 Recall and precision chart of Hausdorff distance matching based system 78

Chapter 1

Introduction

1.1 Background

The popularity and importance of image as an information source is evident in modern society [J97]. The amount of visual information is increasing in an accelerating rate in many diverse application areas. In an attempt to move towards a more paperless office, large quantities of printed documents are digitized and stored as images in databases [D98]. As a matter of fact, many organizations are currently using and dependent on image databases, especially if they use document images extensively. Modern technology has made it possible to produce, process, store and transmit document images efficiently. The mainstream now concentrates on how to provide highly reliable and efficient retrieval functionality over these digital images produced and utilized in different services.

With pictorial information being a popular and important resource for many human interactive applications, it becomes a growing problem to find the desired entity from a set of available data. When dealing with images with diverse content, no exact attributes can directly be defined for applications and humans to use. It is thus very difficult to evaluate and control the relevancy of the information to be retrieved from the image database. Nevertheless, advanced retrieval techniques have been studied to narrow down the gaps between human perception and the available pictorial information. For instance, many effective image descriptions and indexing techniques have been used to seek information containing physical,

semantic and connotational image properties. Not only is the information provided by structural metadata or exact contents, such as annotations, captions and text associated with the image needed, but also a multitude of information gained from other domains, such as linguistics, pictorial information, and document category [M97].

In the past years, various ways have been studied to query on imaged documents using physical (layout) structure and logical (semantic) structure information as well as extracted contents such as image features. For example, Worring and Smeulders proposed a document image retrieval method employing the information of implicit hypertext structure extracted from original documents [WS99]. Jaisimha et al described a system with the ability of retrieving both text and graphics information [JBN96]. Appiani et al presented a document classification and indexing system using the information of document layouts [ACC01]. All these are utilizing content-based image retrieval (CBIR) techniques which extract features using different levels of abstraction.

However, for those imaged documents where text content is the dominant information, the traditional information retrieval approach using keywords is still commonly used. It is obvious that conventional document image processing techniques can be utilized for this purpose. For example, many document image retrieval systems first convert the document images into their machine readable text format, and then apply text information retrieval strategies over the converted text documents. Based on this idea, several commercial systems have been developed using page segmentation and layout analysis techniques, following Optical Character Recognition (OCR). These include Heinz Electronic Library Interactive

Online System (HELIOS) developed by Carnegie Mellon University [GG98], Excalibur EFS and PageKeeper from Caere. All these systems require a full conversion of the document images into their electronic representations, followed by text retrieval.

It is generally acknowledged that the recognition accuracy requirements for document image retrieval are considerably lower than those for many document image processing applications [TBCE94]. Document image retrieval (DIR) is relevant to document image processing (DIP), though with some essential differences. A DIP system needs to analyze different text areas in a document image page, understand the relationships among these text areas, and then convert them to a machine-readable format using OCR, in which each character object is assigned to a certain class. The main question that a DIR system seeks to answer is whether a document image contains particular words that are of interest to the user, while paying no attention to other unrelated words. In other word, a DIR system provides an answer of “yes” or “no” with respect to the user’s query, rather than the exact recognition of a character/word like that in DIP. Motivated by this observation, some methods with the ability of tolerating recognition errors of OCR by using the OCR candidates have been proposed recently [KHOY99]. Some are reported to improve the retrieval performance with the combination of OCR and Morphological Analysis [KTK02].

Unfortunately, several reasons such as high costs and poor quality of document images may prohibit complete conversion using OCR. Additionally, some non-text components cannot be represented in a converted form with sufficient accuracy. Under such circumstances, it can be advantageous to explore techniques for direct characterization, manipulation and retrieval of

document images containing text, synthetic graphics and natural images.

In view of the fact that word, rather than character, is the basic meaningful unit for information retrieval, many efforts have been made in the area of document image retrieval based on word image coding techniques without the use of OCR. In particular, to overcome the problem caused by character segmentation, segmentation-free approaches have been developed. They treat each word as a single entity and identify it using features of the entire word rather than each individual character. Therefore, directly matching word images in a document image with the standard input query word is an alternative way of retrieving document images without complete conversion.

So far, efforts made in this area include applications to word spotting, document similarity measurement, document indexing, summarization, etc. Among all these, one approach is to use particular codes to represent characters in a document image instead of a full conversion using OCR. It is virtually a trade-off between computational complexity and recognition accuracy. For example, Spitz presented the character shape codes for duplicate document detection [S97], information retrieval [SS+97], word recognition [S99] and document reconstruction [S02] without resorting to full character recognition. The character shape codes encode whether the character in question fits between the baseline and the x-line or if not, whether it has an ascender or descender, and the number and spatial distribution of the connected components. Its processing to obtain the character shape codes is simple and efficient but has the problem of ambiguity. Additionally, to get the character shape codes, character cells must be segmented at the first step. It is therefore not applicable to the case

where characters are connected to each other within a word object. Chen et al [CB98] proposed a segmentation and recognition free approach using word shape information. In this approach, it first identifies upper and lower contours of each word using morphology and then extracts shape information based on the pixel locations among these contours. Next, Viterbi decoding of the encoded word shape is used to map the word image with the given keyword. Besides this, Trenkle and Vogt [TV93] also provided preliminary experiment on word-level image matching, where various fonts of the image word are generated, based on which features are extracted and compared with the input keyword. In the domain of Chinese document image retrieval, He et al proposed an index and retrieval method based on character codes generated from stroke density [HJLZ99].

As so many efforts have been devoted to the area of document image processing realm by various researchers especially to OCR, it is a fact that information retrieval methods based on document image processing techniques are still the best so far among all the available retrieval methods. However, DIR and DIP address different needs and have different merits of their own. DIR is tailored for directly retrieving information from document images and thus achieves a relatively high performance in terms of recall, precision and processing speed. Therefore, DIR that bypasses OCR still has its practical value today.

1.2 Scope and Contributions

This thesis presents a word image coding technique that can be used to perform online search of word objects in document image files as well as to design web-based document image retrieval systems for retrieving scanned document images from digital libraries. The

differences between our technique and Spitz's can be summarized as follows:

- Features are extracted at the word level, rather than at the character level as it appears in Spitz's character shape codes.

- The procedure of computing word image codes is more complicated, but shows an advantage of eliminating ambiguity among words.

Based on the aforementioned word image coding technique, two applications are presented in view of online and off-line execution of the word image coding mechanism. First application is a web-based document image retrieval system with the image coding mechanism performed off-line during the preprocessing stage. An experimental system is implemented, which takes in user's query words from a web interface and performs matching among the feature codes generated from the query words and the underlying document images. Preprocessing is carried out off-line to denoise the document images such as skew detection and rectification, and produce the corresponding feature codes using the word image coding technique. Feature codes of the input query words are generated using the same mechanism as is used in the word image coding technique. An inexact matching algorithm is employed in matching the feature codes with the property of matching word portion.

The system consists of four components as shown in Figure 1-1. The web interface is the place where the user inputs a set of query words with AND/OR/NOT operations and gets the retrieved documents ranked by the occurrence frequency of the query words in each document. The users can then link to the actual document and identify the locations of the

matching words. The oracle database is used to store an index table that functions as a cache containing information of previously queried words. This speeds up the search process as more users come to use this system and makes it incrementally intelligent. Lastly, a server is used to store the original imaged documents and their corresponding feature code files generated through the off-line operations.

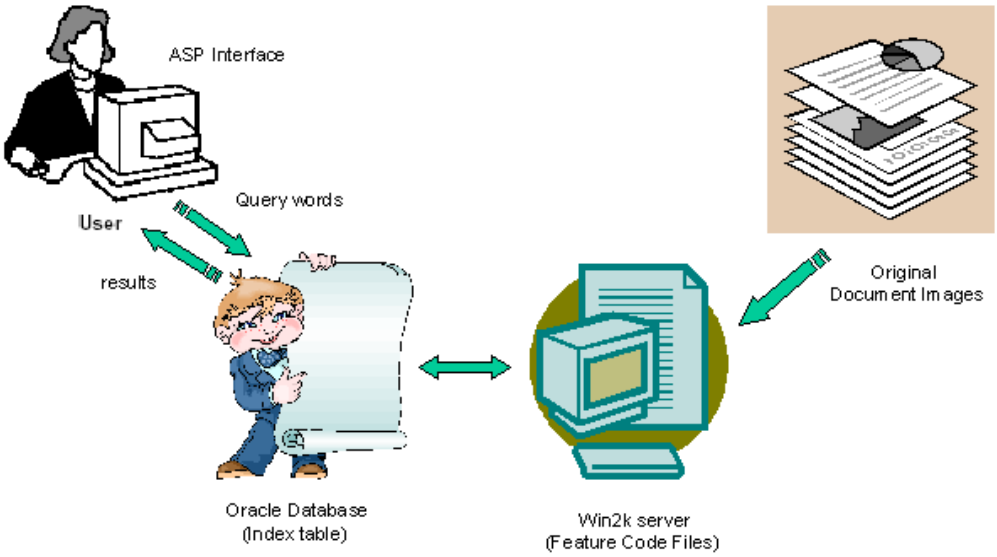


Figure 1-1 System components

The second application is a search engine for imaged documents packed in PDF files. Specifically, a plug-in is implemented and embedded in Acrobat Reader to perform the online search of word objects in the imaged documents. In this application, the word image coding technique employed in the preprocessing phase is done online with no additional database needed for feature code file storage. The feature code file is generated on the user's local machine when he/she performs search for the first time. All the subsequent searches will be simple text matching in the feature code files. A snapshot of the search engine is shown in Figure 1-2.

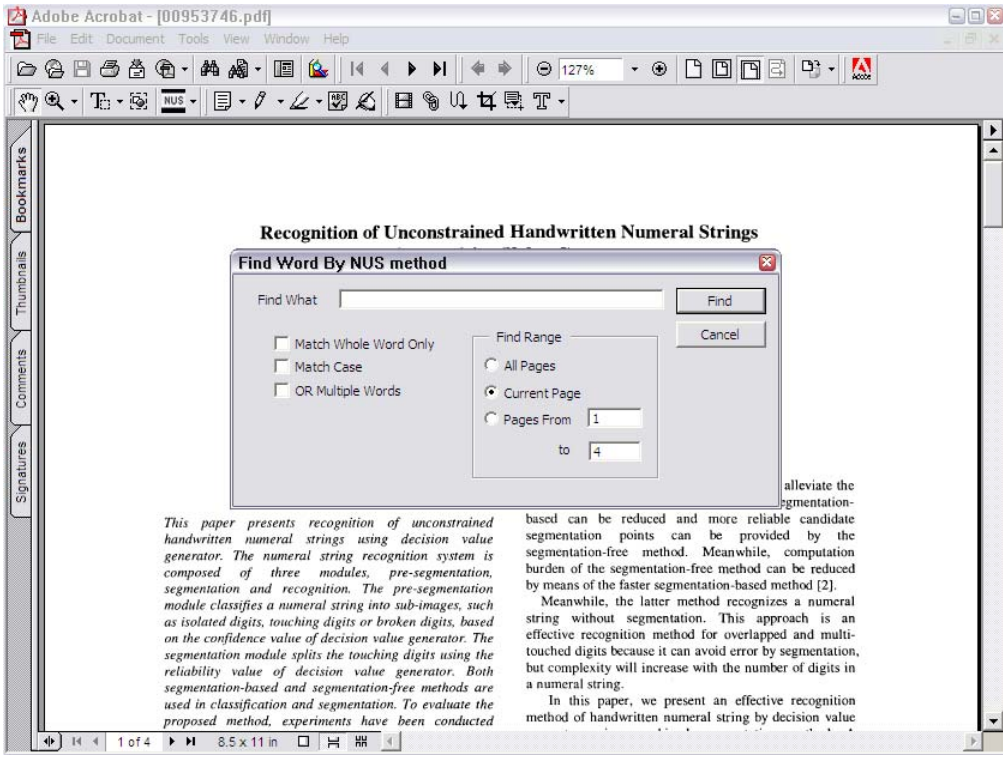


Figure 1-2 Search engine for imaged documents in PDF files

For both applications, a wavelet transformation based technique is proposed for italic font recognition. It is employed during the preprocessing phase to effectively detect italic fonts and rectify them to normal style before generating the feature codes. This is especially helpful in identifying those emphasized words in italic style and also helps to achieve better retrieval performance for italic and normal fonts mixed documents. To evaluate this italic font recognition technique, experiments are conducted on 22,384 frequently used word images in both normal and italic fonts. Our wavelet transformation based technique shows recognition accuracies of 95.76 percent for normal style and 96.49 percent for italic style respectively. Comparisons are done with traditional stroke analysis based approach under the same experimental setup. The results show a significant improvement in the recognition accuracy for four representative fonts in normal and italic styles, namely Times New Roman, Arial,

Courier and Comic Sans MS. Experiments are also conducted on 5,320 normal word images and 489 italic ones extracted from scanned document images. The accuracies achieved are 92.20 percent for normal style and 97.96 percent for italic style respectively.

Last but not least, to compare with the word image coding based search engine, another version of the search engine is implemented based on Hausdorff distance matching of word images. In this case, each word image object is extracted from the imaged document to match with the template word image constructed for the input query word. The Hausdorff distance is calculated to evaluate the distance between two word images as their similarity value. Experiments are performed with scanned images of published papers and students' thesis in our digital libraries with different fonts and quality levels. The results show that better recall and precision are achieved with the word image coding based search engine with less sensitivity to noise affections and font style variations. In addition, by storing the feature codes of the document image in an intermediate file when the first search is performed, we need to perform the preprocessing steps only once and thus achieve a significant speed-up in the subsequent search process.

1.3 Organization of the Thesis

The rest of the thesis is organized as follows:

In chapter 2, we detail the preprocessing procedures that are performed to extract word image objects from the original imaged document and generate their corresponding feature code strings using the word image coding technique.

In chapter 3, we discuss the word image coding technique that is used for feature code generation and evaluate its validity as a unique coding representation at the word level.

In chapter 4, we describe the wavelet transformation based technique for italic font recognition and how it is compared with traditional stroke pattern analysis method.

In chapter 5, we elaborate the inexact string matching algorithm exploited in matching the feature code strings of the word images.

In chapter 6, we illustrate the implementation of the first application of the word image coding technique, namely the web-based document image retrieval system given a set of query words.

In chapter 7, we describe the implementation of the second application of the word image matching technique, namely the search engine for imaged documents in PDF files. Experiments show that our search engine is 2.6 times faster than the Page Capture provided by Adobe Acrobat. Comparisons made with a testing search engine implemented based on Hausdorff distance matching show much better efficiency and less sensitivity to noise and font variations for the word image coding based system.

In chapter 8, we draw some conclusions and discuss about the future works.

Chapter 2

Feature Code File Generation

With respect to each document image, a corresponding feature code file is generated off-line by undergoing some preprocessing procedures prior to the online search process. This feature code file contains all the feature code strings and is stored on a server as a database for future matching. The document images used in our system are scanned from published papers and students' theses packed in PDF files. Each PDF file has over 100 images in page format for those students' theses. Each page image needs to be preprocessed before being converted to its corresponding feature code representation. The detailed procedures are elaborated in the following sections.

2.1 Connected Component Analysis

Consider a particular page of a given document image, we first apply a connected component analysis algorithm to detect all the connected components within this page. Here, we assume all the images are binary images with black and white pixels (otherwise convert to binary images). The connected component is defined as an area inside which all the image pixels are connected to each other. For example, Figure 2-1 shows a portion of a page image after applying the connected component analysis.

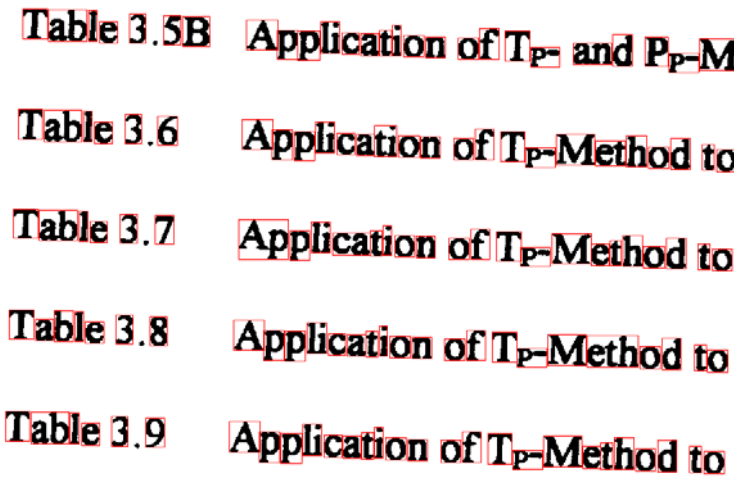


Figure 2-1 Connected components

In particular, the connected component analysis algorithm we are using here is a component-oriented method. Each time we start with a black pixel in a new connected component and go round to mark all the black pixels in its eight neighbors (consider the current pixel as the center of a 3 by 3 matrix). After that we set the current pixel to be white and continue with the previously marked neighbors. The process follows the fashion of breadth-first search and stops until all the neighbors of the marked black pixels are white. The final rectangle area bounded by the boundary pixels is known as a connected component.

Furthermore, additional operations are carried out to remove some useless information obtained from the detected components. In particular, those connected components with too small area are usually punctuations or noise pixels and are therefore removed. One thing to note in this case is the small dot detected as part of ‘i’ and ‘j’, we will group them with the body part of ‘i’ and ‘j’ as one connected component instead of discarding them. This is done by the observation that the gap distance between the dot and the body of ‘i’ and ‘j’ is normally smaller than the gap distance between the dot and the line above it. This property helps us to

obtain a complete shape for ‘i’ and ‘j’. Similarly, those components with too large area (e.g. width/height is greater than 5 times the median width/height of the components) are probably tables or figures and are therefore eliminated as well. What we concern is mainly the text information rather than graphics and tables.

2.2 Word Bounding

Having detected the connected components, we try to find all the word-bounding boxes based on the locations of these connected components. To find the boundaries of each word object, the same idea can be applied as in finding the connected components in the section 2.1. For each connected component, we search all its eight neighboring connected components to find the leftmost component and rightmost component until the gap between two connected components are too large to be within one word. Based on the boundary connected components, we determine the bounding rectangle for the word object. Furthermore, some additional conditions are applied to remove those too large or too small word-bounding boxes and merge those word-bounding boxes with large overlapping area. Figure 2-2 gives an example of the word-bounding boxes detected for a portion of a page image.

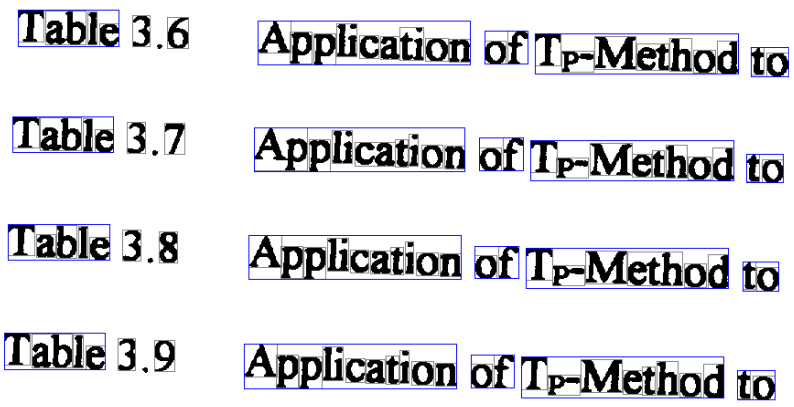


Figure 2-2 Word bounding box

2.3 Skew Estimation

As we can see from Figure 2-1 and 2-2, this particular page image is not in its normal shape in terms of the physical layout. Specifically speaking, each line has a skew angle against the horizontal axis. In order to generate an accurate set of feature code strings for this page image, we need to first rectify this page image back to its normal shape before applying the word image coding scheme. To rectify the page image, we need to first find its skew angle. This is done by using a nearest neighbor chain (NNC) algorithm [LT03] [ZLT03]. The idea lies in the observation that the slope of an inclined line can generally be reflected by the slope of a nearest neighbor chain that consists of several consecutive connected components of similar height/width. For example, in the second line of Figure 2-3, 'i' 'o' 'n' is detected as a NNC of length 3, because 'i' 'o' and 'n' are three consecutive connected components of similar size. As we can see, the slope of this NNC is close to the slope of the whole line.

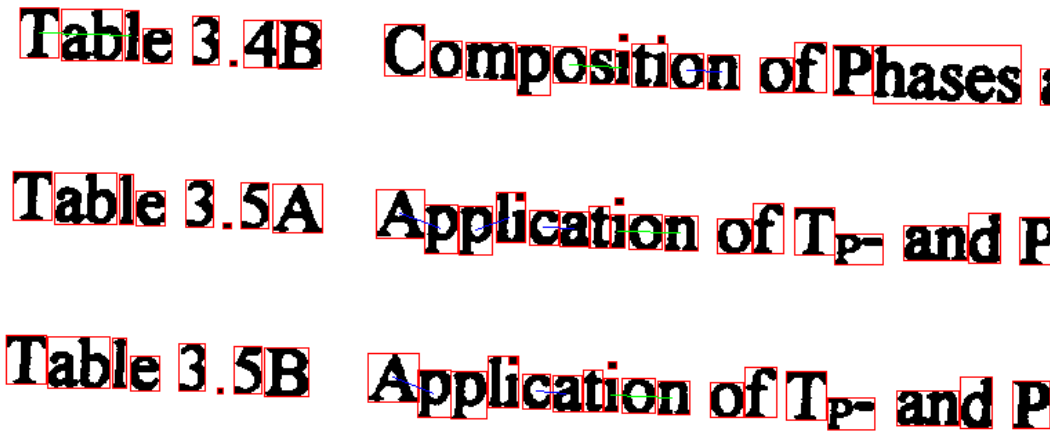


Figure 2-3 Nearest Neighbor Chains (NNCs)

In particular, for a component C_i, we use (x_{c_i}, y_{c_i}) to represent its centroid; (x_{l_i}, y_{t_i}) and (x_{r_i}, y_{b_i}) to represent the upper-left and bottom-right coordinates of the rectangle enclosing

C_i; and h_{c_i} and w_{c_i} to represent the height and width of C_i respectively. Then the centroid distance and gap distance between two components are defined as follows:

Definition 1 The centroid distance between two components C_1 and C_2 is defined as:

$$d_c(C_1, C_2) = \Delta x^2 + \Delta y^2$$

where $\Delta x = | x_{c_1} - x_{c_2} |$ and $\Delta y = | y_{c_1} - y_{c_2} |$ as shown in Figure 2-4.

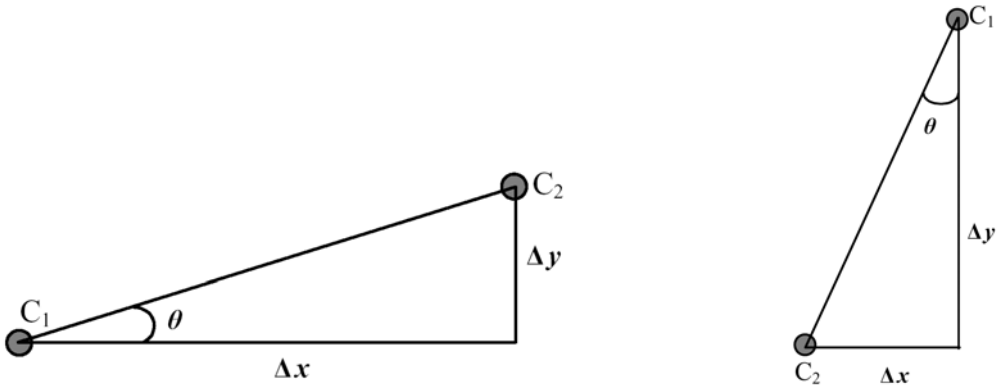


Figure 2-4 Skew angle (a) $\Delta x > \Delta y$ (b) $\Delta x < \Delta y$

Definition 2 The gap distance between two components C_1 and C_2 is defined as:

$$d_g(C_1, C_2) = \begin{cases} max(x_{12} - x_{r1}, x_{11} - x_{r2}) \\ max(y_{t2} - y_{b1}, y_{t1} - y_{b2}) \end{cases}$$

Let m be the total number of connected components generated from a page image, then the nearest neighbor pair is defined as follows:

Definition 3 [C_1, C_2] is a nearest neighbor pair if $\Delta x > \Delta y$, and

(1) $h_{c1} \cong h_{c2}$

(2) $x_{c2} > x_{c1}$

(3) $d_c(C_1, C_2) = \min d_c(C_1, C_m)$

(4) $d_g(C_1, C_2) < \beta * \max(h_{c1}, h_{c2})$

or if $\Delta y > \Delta x$, and

(1) $w_{c1} \cong w_{c2}$

(2) $y_{c2} > y_{c1}$

(3) $d_c(C_1, C_2) = \min d_c(C_1, C_m)$

(4) $d_g(C_1, C_2) < \beta * \max(w_{c1}, w_{c2})$

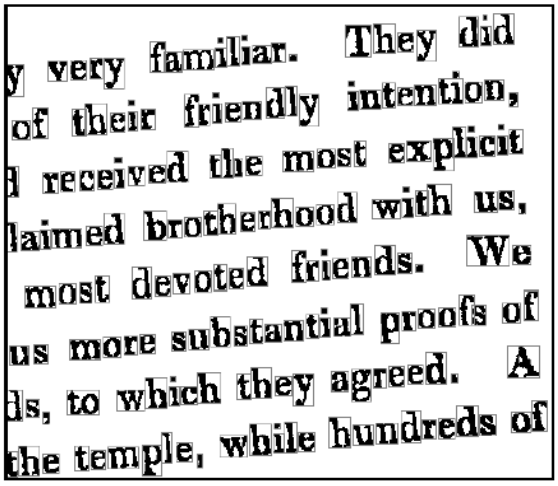
where β is a constant, and is set to be 1.2 experimentally.

According to the definitions above, the adjacent nearest neighbor pairs with similar heights or width will produce a nearest neighbor chain.

Definition 4 K-nearest-neighbor chain (K-NNC) is defined as a string containing K connected components $[C_1, C_2, \cdots, C_K]$, in which C_{i+1} is the nearest-neighbor of C_i for i = 1, 2, …, K-1.

Based on some observations on K-NNCs for several English document images with K=2, K=3 and K≥4 respectively (as shown in Figure 2-5), we conclude that the larger K is, the more accurately the slope of the K-NNC can reflect the skew angle of the page image. As an example of why shorter NNCs are not used in the estimation, Figure 2-6 shows the 2-NNC and 3-NNC respectively for the word “complete”. Clearly, the slope of 3-NNC reflects the skew angle more accurately than that of those 2-NNCs. This is because there may be some noise in shorter NNCs. Therefore, what we do is to extract the longest NNC from the adjacent nearest neighbor pairs and determine the skew angle based on the median of the slopes of all

these NNCs.



(1)

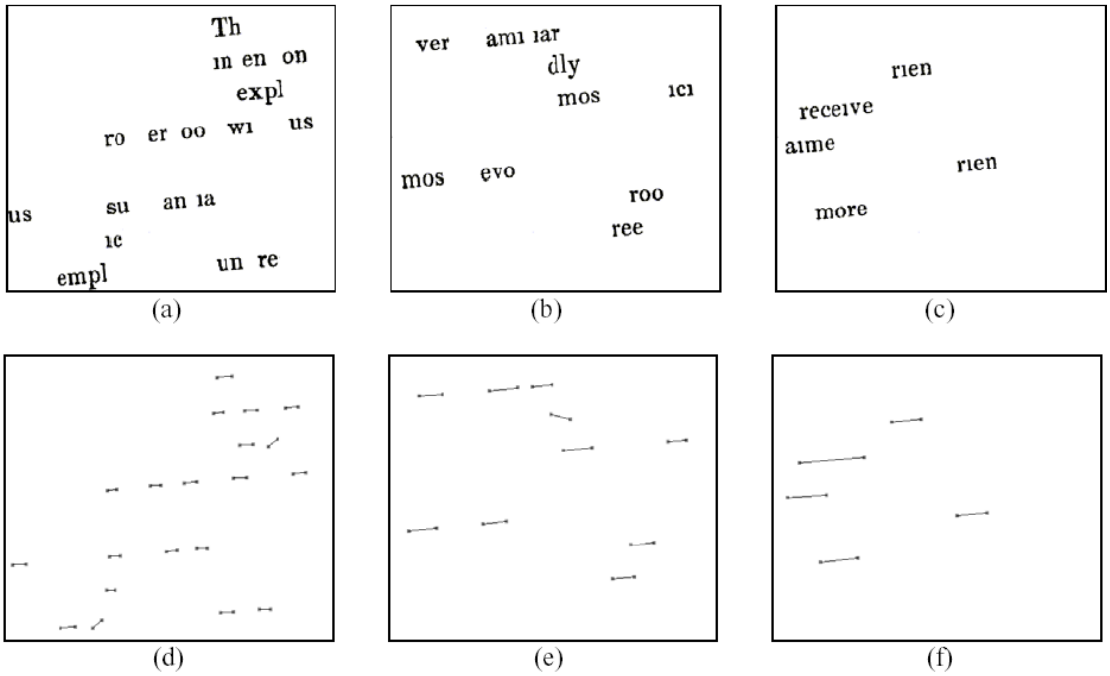


Figure 2-5 NNCs for (1): (a) (d) K=2 (b) (e) K=3 (c) (f) K≥4

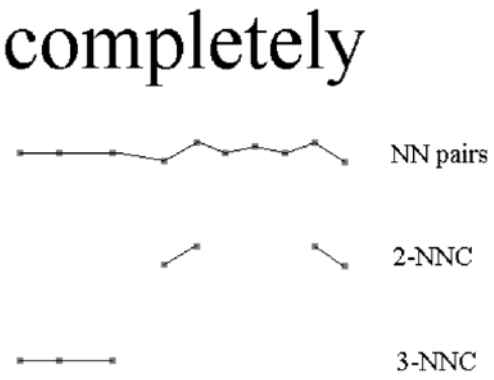


Figure 2-6 Nearest Neighbor Chain (NNC)

Definition 5 Suppose $S^{(n)} = [C^{(n)}_1, C^{(n)}_2, \ldots, C^{(n)}_K]$ is the nth K-NNC ($n = 1, 2, \ldots, N$), then its slope is defined as:

$$slope_K^{(n)} = \begin{cases} (x_{ck}^{(n)} - x_{c1}^{(n)}) / (y_{ck}^{(n)} - y_{c1}^{(n)}) \text{ if } (x_{ck}^{(n)} - x_{c1}^{(n)}) < (y_{ck}^{(n)} - y_{c1}^{(n)}) \\ (y_{ck}^{(n)} - y_{c1}^{(n)}) / (x_{ck}^{(n)} - x_{c1}^{(n)}) \text{ if } (y_{ck}^{(n)} - y_{c1}^{(n)}) < (x_{ck}^{(n)} - x_{c1}^{(n)}) \end{cases}$$

For a constant K, we can obtain the median of the slopes of all its NNCs. This will be the value we use to represent the skew angle of this page image. In addition, we make use of a predefined threshold to guarantee that there are sufficient NNCs of a particular length K in order to avoid the noise factors and give an accurate estimation.

2.4 Skew Rectification

Having obtained the skew angle of the page image, we try to rectify each word back to its normal shape based on this angle. The idea is to obtain an image of word-bounding box inside which the word is in its right position. This can be visualized from Figure 2-7(a). Here, "Application" has a skew angle of β degree with respect to the dashed word-bounding box ***S*** that is horizontal. Now we turn this dashed box clockwise by β degree to obtain a new word-bounding box ***R***. Obviously, "Application" is in a right position with respect to ***R***.

Therefore, ***R*** is the word-bounding box image that we need.

One thing worth mentioning is that the word-bounding boxes we generated at the previous step (Section 2.2) are all horizontal. Next, we need to rotate these word-bounding boxes by the skew angle to obtain a new word-bounding box so that inside which the word is in its normal shape. In order to make sure all the word image pixels can be enclosed in the rotated word-bounding box ***R***, we give a tolerance boundary of 2 pixels for the original word-bounding box ***S*** so that there will not be information loss due to the rotation. This guarantees the accuracy of the feature code generation.

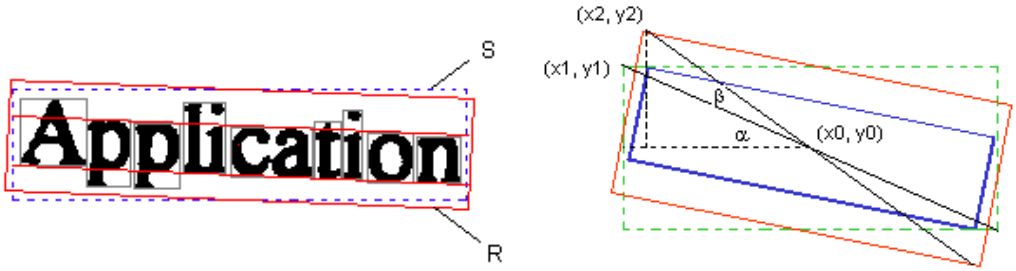
The following formula will map the corresponding image pixels in the original word-bounding box ***S*** to the newly generated word-bounding box image ***R*** as shown in Figure 2-7(b):

$$x_2 = x_0 - [(x_0 - x_1) * cos\beta + (y_0 - y_1) * cos\beta]$$

$$y_2 = y_0 - [(y_0 - y_1) * sin\beta + (x_0 - x_1) * cos\beta]$$

Here, (x_0, y_0) is the center of the horizontal word-bounding box ***S***. What we want to do is to construct a new word-bounding box image inside which all the pixel values are allocated to form a normal shaped word “Application”. This is done by assigning each pixel value inside this new image to the corresponding pixel values in the word-bounding box ***R*** obtained by rotating the original word-bounding box ***S*** by a degree of the skew angle. Now we have obtained a new word image that is in normal shape. Next, we can operate on this small word image to find its corresponding feature code. Figure 2-8 shows a portion of the rectified page

image.



(a) (b)

Figure 2-7 Skew rectification

Table 3.6 Application of T_P-Method to

Table 3.7 Application of T_P-Method to

Table 3.8 Application of T_P-Method to

Figure 2-8 A portion of a rectified page image

2.5 Word Bounding Box Regeneration

After rectifying the word to its normal shape, the previous connected components generated for calculating NNC are no longer accurate. Since the shape of the character strictly affects its bounding area, we cannot simply rotate the previous connected component by the skew angle to obtain the new one. Therefore, we need to regenerate connected components for the normalized word shape. Concerning the efficiency issue, this time we apply the connected components analysis algorithm only for each individual word-bounding box generated in the above step. With a smaller image area, this process will be much faster than scanning through the whole page image.

Next, the word objects are bounded by analyzing the information of relative positions among the new connected components. The idea is the same as the word bounding step in Section 2.2, but the connected components to be searched are only restricted to those contained within the current word image. Therefore, it will be much faster than the previous word bounding step.

2.6 Italic Font Detection

As we noticed, in many document images certain terms are emphasized and distinguished with italic style. These are usually important words with higher information content. As we will see in Chapter 4, Chaudhuri and Garain conducted statistical study [CG98] on the relative abundance and importance of italic, bold and all-capital words in technical journals, proceedings of technical conferences, technical books, etc. It shows that italic style indeed occupies a significant portion in many document images. Thus, it is necessary to identify the italic styles before performing corresponding rectification to produce their normal forms and generate the normal feature code strings for matching.

In view of our word image coding scheme, feature extraction is performed on a word level without character segmentation. This requires the ability of identifying each italic word as an individual entity instead of within a block of italic text. Some existing techniques are targeted at identifying fonts and styles of large text blocks as those listed in Chapter 4. This does not apply to individual italic word recognition as is required here. Since at this stage each word image object is already extracted, it is easy to think of performing stroke pattern analysis on each word image object to distinguish italic and non-italic styles. However, the traditional stroke pattern analysis performed directly on the word image object is highly sensitive to

noise level and typeface variations. To remedy this problem, we proposed a wavelet transformation based technique that performs a 2-D wavelet decomposition step to extract predominant features from the word images, followed by the stroke pattern analysis on the sub-images generated. The predominant features extracted from the word images contain distinguishable information of italic and non-italic styles and meanwhile are less sensitive to noise and typeface variations. Details about this technique will be illustrated in Chapter 4.

2.7 Italic Font Rectification

If a word object is detected as in italic style, a rectification step will be carried out to de-italicize the word before generating its feature code string. This is done by first estimating the oblique angle of the italicized word. Experiments show that in most computer generated fonts, the oblique angle is between 10 to 15 degrees. Next, the word object is rectified by shifting each pixel horizontally left by a corresponding distance calculated according to the oblique angle with respect to the left bottom boundary of the word bounding box. An example of the rectified word “Principle” is shown in Figure 2-9. The word bounding box is relocated with its new left and right boundaries.

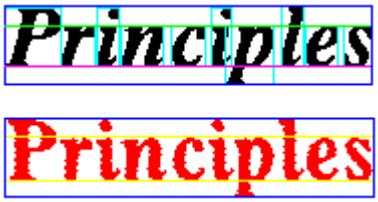


Figure 2-9 Italic word and its rectified image

2.8 Feature Code File Generation

At this stage, each word object is extracted from the document image and rectified to its normal shape if italic rectification is applicable. Next, by applying the word image coding technique, each word image is represented using a primitive string as to be illustrated in Chapter 3. The feature code file is then generated, which contains the information of all the feature code strings corresponding to the word objects and their locations in the document as well as the URL of the document image. Figure 2-10 gives a portion of a feature code file recording the information of a PDF file with 33 pages of image.

```
[FileName] c:\temp\00953744_1.wrd
[NumberOfPage] 33
[ImageSize] 2559,3299
[ImageSkew] 0.0000000
[NumberOfWord] 365

[No] 0
[pos] 1630 416 1739 455
[Ap] 0
[Fn] 14
[FC] -coe-coe-om-c-

[No] 1
[pos] 807 417 1046 456
[Ap] 1
[Fn] 30
[FC] -WV-cod-vw-oem-mnm-co-ceo-cod-

[No] 2
[pos] 1064 417 1309 456
[Ap] 1
[Fn] 34
[FC] -dOS-dnm-oem-mn-oem-co-ndo-ceo-mn-
```

Figure 2-10 Feature code file

Chapter 3

Word Image Coding

Concisely speaking, our word image coding technique is to represent each word object extracted from the document images using specially designed codes according to its features [LZT04]. The features used in our approach are Left-to-right Primitives. Each word object is therefore denoted by a string of these primitives sequenced from the leftmost of a word to its rightmost referred to as Left-to-right Primitive String (LRPS). Primitives are extracted from the word image based on line features and traversal features to be illustrated in section 3.2.

3.1 LRPS Feature Representation

To extract primitives, each word object is explicitly segmented from the leftmost to the rightmost to discrete entities. Each entity, called a primitive here, is represented using two definite attributes (σ, ω), where σ is the Line-or-traversal Attribute (LTA) of the primitive and ω is the Ascender-and-descender Attribute (ADA). Consequently, each word object is expressed as a sequence *P* of $p_i's$.

$$P = < p_1 p_2 \cdots p_n > = < (\sigma_1, \omega_1)(\sigma_2, \omega_2) \cdots (\sigma_n, \omega_n) >$$

3.2 Ascender-and-descender Attribute

We assign five characters to the values of ADA, i.e. $\omega \in \Omega = \{'x','a','A','D','Q'\}$. Each of these five characters reflects a typical feature of the primitive and is defined as follows:

- ‘x’: the primitive is between the x-line and the baseline;

- 'a': the primitive is between the top-boundary and the x-line;
- 'A': the primitive is between the top-boundary and the baseline;
- 'D': the primitive is between the x-line and the bottom-boundary;
- 'Q': the primitive is between the top-boundary and the bottom-boundary.

The definition of x-line, baseline, top and bottom-boundary can be found in Figure 3-1. Each word object extracted from the document image already contains the information of x-line and baseline, which is a by-product of the text line extraction in the preprocessing stage.

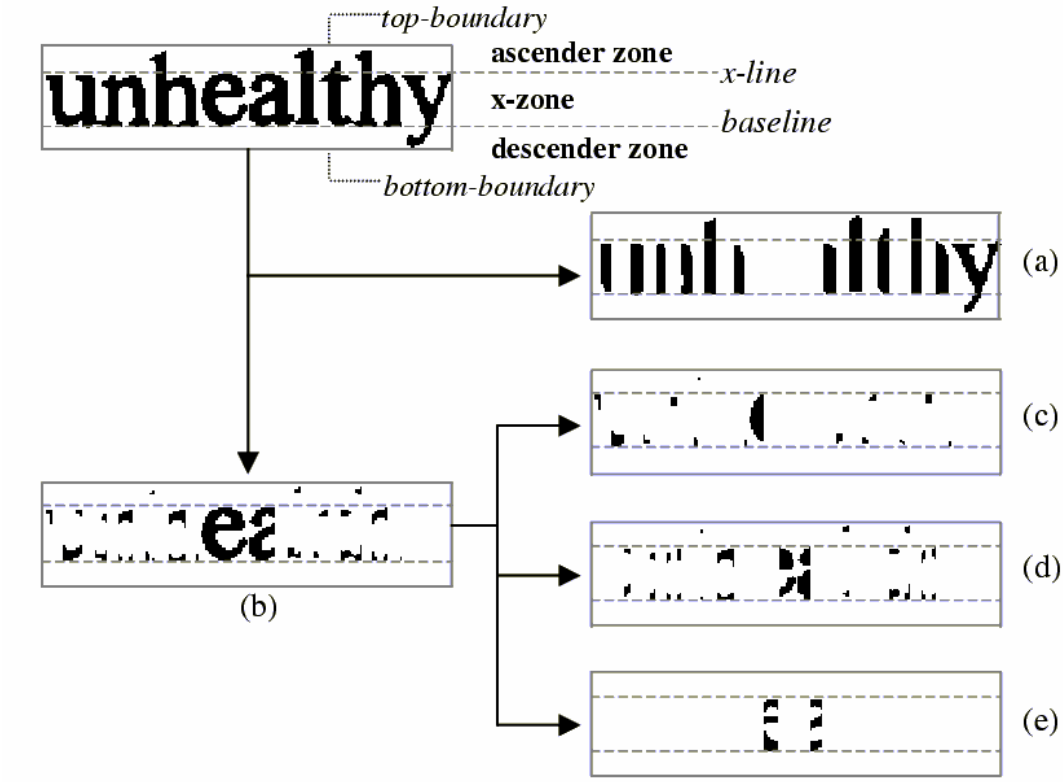


Figure 3-1 Primitive string extraction
(a) straight stroke line features (b) remaining part of word image
(c) traversal T_N = 2 (d) traversal T_N = 4 (e) traversal T_N = 6

3.3 Line-or-traversal Attribute

The generation of LTA is performed in two steps. First, the straight stroke line features are

extracted from the word image, as shown in Figure 3-1(a). Note that only the vertical stroke lines and diagonal stroke lines are extracted at this stage. Then, the traversal features of the remaining word image are analyzed. Finally, the features obtained from the previous two steps are combined to generate the LTAs of the corresponding primitives. In other word, the LTA of a primitive is represented by either a straight stroke line feature or a traversal feature otherwise.

3.3.1 Straight Stroke Line Feature

A run-length based method is utilized to extract straight stroke lines from word images. We use $R(a,\theta)$ to represent a directional run, which is defined by a set of concatenating pixels that contain pixel *a*, along the specified direction θ. $|R(a,\theta)|$ is the run length of $R(a,\theta)$, which is the total number of black pixels in the run.

The straight stroke line detection algorithm is summarized as follows:

- Along the middle line of the x-line and the baseline, detect the boundary pair $[A_l, A_r]$ of each stroke line segment, where A_l and A_r are the left and right boundary points of the line segment respectively;
- Locate the midpoint A_m of each line segment $\overline{A_l A_r}$;
- Calculate $R(A_m,\theta)$ for a range of θ value, from which we select θ_{max} as the $A_m's$ run direction;
- If $|R(A_m,\theta_{max})|$ is near to or larger than the x-height (distance between the x-line and the baseline), the set of pixels between the boundary points A_l and A_r along

the direction θ_{max} are extracted as a straight stroke line.

As is shown in Figure 3-1, the straight stroke lines in the word “unhealthy” are extracted and displayed in Figure 3-1(a), while the remaining image pixels are shown in Figure 3-1(b). According to the direction of a straight stroke line, it is assigned to one of three categories: vertical stroke line, left-down diagonal stroke line and right-down diagonal stroke line. Associated with these three types of straight stoke lines, three basic primitives are generated. The ADAs of these primitives can be evaluated based on their top-end and bottom-end positions of the stoke lines. For example, the left-down diagonal stroke line in the character ‘z’ is located between the x-line and the baseline. Therefore, the primitive associated with this left-down diagonal stroke has a value of ‘x’ for its ADA. Similarly, the right-down diagonal stroke line in the character ‘V’ is located between the top-boundary and the baseline. Hence, the corresponding primitive’s ADA will have the value ‘A’ accordingly.

On the other hand, the LTAs of these three types of primitives are evaluated as follows:

- ‘l’: vertical stroke line, such as those in characters ‘l’, ‘d’, ‘p’, ‘q’, ‘D’, ‘P’, etc. For the primitive whose ADA is ‘x’ or ‘D’, we will further check whether there is a dot on the top of the vertical stroke line. If there is, the LTA of the primitive is re-assigned with the value ‘i’ or ‘j’ respectively.
- ‘v’: right-down diagonal stroke line, such as those in the characters ‘v’, ‘w’, ‘V’, ‘W’, etc.
- ‘w’: left-down diagonal stroke line, such as those in the characters ‘v’, ‘w’, ‘z’, etc. For the primitive whose ADA is ‘x’ or ‘A’. We will further check whether there are two horizontal stroke lines connected with the stroke line at the top and bottom respectively. If there are, the LTA of this primitive is re-assigned with the value ‘z’.

Additionally, it is easy to detect primitives containing two or more straight stroke lines as follows:

- ‘x’: one left-down diagonal stroke line crosses with one right-down diagonal stroke line at the middle line between the x-line and the baseline.
- ‘y’: one left-down diagonal stroke line meets one right-down diagonal stroke line with additional pixels between the baseline and the bottom-boundary under the right-down diagonal stoke.
- ‘Y’: one left-down diagonal stroke line, one right-down diagonal stroke line both with top-end above the x-line and one vertical stroke line meet at one point between the x-line and the baseline.
- ‘k’: one left-down diagonal stroke line, one right-down diagonal stroke line and one vertical stroke line with top-end above the x-line meet at one point between the x-line and the baseline.

3.3.2 Traversal Feature

After the primitives based on the straight stroke line features are extracted as described above, the primitives of the remaining part of the word image is generated based on the traversal features.

To extract the traversal features, we scan the remaining word image column by column. The traversal number T_N is recorded by counting the number of transitions from black pixel to white pixel, or vice versa, along each column. According to the value of T_N, different feature codes are then assigned based on the following definition:

- $T_N = 0$: there is no image pixel in the column. We assign it with the feature code ‘&’. It corresponds to the inter-character space. We treat each inter-character space as a

special primitive. In addition, the overlap of adjacent characters caused by kerning is easily detected by analyzing the relative positions of the adjacent connected components. Based on this, we can insert a space primitive wherever is applicable.

- $T_N = 2$: two parameters are used to assign its feature code. One is the ratio of its black pixel number to the x-height, referred to as κ. The other is the relative position of the strokes with respect to the x-line and the baseline, $\xi = D_m / D_b$, where D_m is the distance from the x-line to the topmost stroke pixel in the column and D_b is the distance from the bottommost stroke pixel to the baseline. The feature codes are assigned as follows:
  - 'n': $\kappa < 0.2$ and $0 \le \xi < 0.3$
  - 'u': $\kappa < 0.2$ and $\xi > 3$
  - 'c': $\kappa > 0.5$ and $0.5 < \xi < 1.5$
  - 'T': $\kappa < 0.2$ and $\xi < -0.2$
- $T_N = 4$: assign it with the feature code 'o' or 'O' based on the location of the topmost stroke pixel. If the topmost stoke pixel is near the x-line, 'o' is assigned. Otherwise, if the topmost stroke pixel is near the top-boundary, 'O' is assigned.
- $T_N = 6$: assign it with the feature code 'e' or 'E' based on the location of the topmost stroke pixel.
- $T_N = 8$: assign it with the feature code 'g' as there are four short stoke lines along the column.

As a result, a series of primitives are generated and expressed as a sequence of (σ, ω) tuples representing either straight stroke line features or traversal features as shown in Figure 3-1(a) and Figure 3-1(c)(d)(e) respectively.

One thing to note is that a few columns may result in no corresponding feature code assigned because they cannot meet any of the requirements for the aforementioned eligible feature codes. Some of these are insignificant features or most likely caused by noise. Therefore, these columns are eliminated automatically at this stage.

3.4 Post-processing

3.4.1 Merging Consecutive Identical Primitives

As we mentioned in section 3.1, each primitive is described by two attributes σ and ω, where σ is assigned with different feature code values according to the type of features detected and ω is also associated with five values to describe the ascender or descender property of the primitive. Based on our observation, the significative combinations of σ and ω are limited. For example, $\sigma = \text{'n'}$ can only correspond to $\omega = \text{'x'}$. Therefore, for conciseness, we can replace each (σ, ω) pair in the primitive sequence generated above by one single character as listed in Table 3-1. Consequently, the sequence of primitives can be expressed as a string of character code representation.

Meanwhile, consecutive identical primitives may appear in the sequence such as the continuous vertical stroke lines in the word “unhealthy”. These are redundant features that can be combined and represented by one single character code. This reduces the length of the feature code representation without loss of feature information. At this stage, the resultant primitive string of the word “unhealthy” in Figure 3-1 is obtained as follows:

<nmuomuomonomu&Odomn&ceo&oemuOd&ndoOdonomu&y>

3.4.2 Refinement for Font Independence

It is desirable that the retrieval system is able to retrieve document images with different fonts and styles. To achieve this, the primitive string we obtained at the earlier stage should be independent of typefaces. Among various fonts, a significant factor that affects the LRPS extraction is the property of serif. This is particularly true for the extraction of traversal features. Therefore, it is a basic necessity to avoid the effect of serif in the LRPS representation.

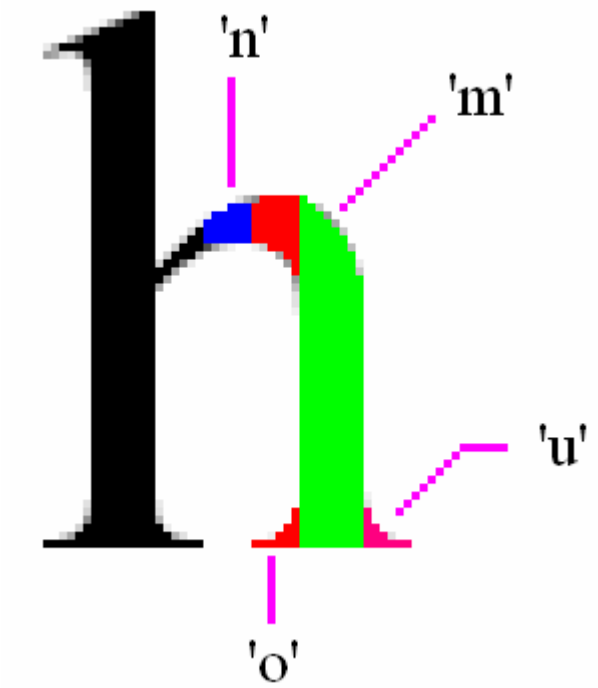


Figure 3-2 Refinement for LRPS representation to avoid the effect of serif

Based on our observation, a primitive produced by serif can be eliminated by analyzing its preceding and succeeding primitives. For instance, a primitive assigned with the character code ‘u’ in a primitive sequence <mu&> is normally generated by a right-side serif in the characters such as ‘a’, ‘h’, ‘m’, ‘n’, ‘u’, etc. Therefore, we can simply remove this primitive

represented by ‘u’ from the primitive sequence <mu&>. Similarly, a primitive assigned with the character code ‘o’ in a primitive sequence <nom> is normally generated by a serif in the characters such as ‘h’, ‘m’, ‘n’, etc. Hence, we can directly eliminate the primitive represented by ‘o’ from the primitive sequence <nom> as well. An illustration is shown in Figure 3-2. More refinement rules are applied to eliminate the primitives caused by serif. With this post-processing step, the primitive string of the word image in Figure 3-1 becomes:

<mumuomnm&dom&ceo&oemd&ndodnm&y>

Besides the ability of dealing with serif, our coding mechanism also features in its independence of bold faces. This is because the earlier step of merging consecutive identical primitives combines redundant features and many of these redundant features are actually caused by bold faces.

Primitive Properties (σ, ω)	Character Code Representation	Primitive Properties (σ, ω)	Character Code Representation
(o, x)	o	(z, x)	Z
(e, x)	e	(l, A)	d
(l, x)	m	(l, D)	q
(c, x)	c	(u, a)	T
(n, x)	n	(c, a)	P
(u, x)	u	(o, A)	O
(v, x)	v	(e, A)	E
(w, x)	w	(c, A)	C
(g, D)	g	(v, A)	V
(i, A)	i	(w, A)	W
(i, Q)	j	(k, A)	K
(k, x)	k	(x, A)	X
(x, x)	x	(Y, A)	Y
(y, D)	y	(z, A)	Z
(e, Q)	Q		

Table 3-1 Primitive properties vs. Character code representation

3.5 Primitive String Token for Standard Characters

Based on the feature extraction mechanism described above, we can associate each of the 26 characters with a standard primitive string token (PST). For example, the primitive string token of character ‘b’ is <doc> and the PST of ‘p’ is <qoc>. Table 3-2 lists the corresponding PSTs for all the characters. Consequently, the standard primitive string of a word can be generated by synthesizing the primitive string token of each character in the word and inserting a special primitive <&> in between to indicate character gap.

Generally speaking, due to many noise factors such as connections between adjacent characters, the resulting primitive string generated from a real word image is usually not as perfect as that synthesized from the standard PST of the corresponding characters. As the example in Figure 3-1 shows, due to noise effect, the primitive substring with respect to the character ‘h’ is extracted as <dom> instead of <dnm> as in the standard representation. Similarly, the connected characters ‘al’ and ‘th’ also result in the variations of the primitive substring generated from the original document image. To solve this problem, an inexact string matching algorithm is employed during the feature code matching step, which compensates for the misgenerated feature code (to be illustrated in Chapter 5).

Character	Primitive Sting Token	Character	Primitive String Token
a	oem	A	WV
b	doc	B	dEd
c	co	C	CO
d	cod	D	dOC
e	ceo	E	dE
f	ndT	F	dOT
g	g	G	COEO
h	dnm	H	dnd

i	i	I	d
j	j	J	ud
k	k	K	K
l	d	L	du
m	mnmnm	M	dVWd
n	mnm	N	dVd
o	coc	O	COC
p	qoc	P	dOP
q	coq	Q	COQC
r	mn	R	dOEO
s	oeo	S	OEO
t	ndo	T	TdT
u	mum	U	dud
v	vw	V	VW
w	vwvw	W	VWVW
x	x	X	X
y	y	Y	Y
z	z	Z	Z

Table 3-2 Primitive string tokens of characters

3.6 Verification

We use a dictionary containing 25,133 commonly used English words to evaluate the validity of the proposed word image coding scheme. Each word is represented by its corresponding word primitive token (WPT) generated by concatenating its characters' primitive string tokens described above. Character gaps are denoted by the special primitive <&> inserted between two adjacent PSTs. For example, the WPT of the word "health" is generated as:

<dnm&ceo&oem&d&ndo&dnm>

The investigation found that each word in the dictionary has a unique coding representation which is distinguishable from all the other words, although there is ambiguity at the character level, e.g. the PSTs of the character 'l' and 'I' are the same. This proves that our coding

scheme produces no ambiguity in its representation for word images, but at the cost of more computational burden comparing to Spitz's character coding scheme.

Chapter 4

Italic Font Recognition

As we noticed in many scanned document images such as those conference papers, it often appears that the “Abstract” section is written in italic font as required. Moreover, there are also some italic words that are scattered in the document for emphasizing purpose or as scientific names. These are usually keywords that carry significant information content in view of retrieving documents based on query words. In this chapter, we compare several traditional italic font recognition methods and propose a wavelet transformation based technique that features in detecting italic font at the word level and with less sensitivity to noise and typeface variations.

4.1 Background of Font Recognition

Font recognition is a fundamental issue in document analysis and recognition, and is also a difficult and time-consuming task. Nowadays many commercial OCR systems have claimed to achieve high recognition accuracy in identifying English and related scripts. Some of the products can accommodate font variations to a reasonable extent. Many are observed to have a deteriorated performance with style variations. Baird and Nagy demonstrated that a significant improvement in recognition accuracy can be achieved by utilizing font information, where a 100-font classifier was automatically adapted to a specific font [BN94].

Generally speaking, there are two complementary approaches that are used to address the font recognition problem: the a priori approach, in which the characters of the analyzed text are

not yet known and the a posteriori approach, where the content of the given text is used to identify the font. Zramdini and Ingold proposed a novel a priori font recognition approach that identifies the typeface, weight, slope and size of a text image block by extracting global typographical features and feeding them into a multivariate Bayesian Classifier [ZI98]. In this approach, a set of known fonts are given as a font model base for classification purpose. Khoubyari and Hull also introduced a method that identifies the predominant font of a document image by matching clusters of word images to a pre-generated database of function words derived from fonts and document images [KH96]. Cooperman used a set of local detectors to estimate font attributes such as serifness and boldness in an OCR system [C97]. Zhu et al. described a global texture-analysis-based font recognition method on normalized text blocks. This content-independent approach avoids connected component analysis for detailed local feature extractions [ZTW01].

Having said that many OCR systems' performance is subject to font variations, it is observed that the performance degradation is more drastic with italic style, which increases with the increase of the slant angle of the italicized characters. One possibility to improve the performance is to detect the italicized words in the document, compute the slant angle and rectify them by a shearing transform corresponding to the slant angle and feed them to the OCR system. Shi and Pavlidis proposed a method of discriminating between italic and non-italic font by analyzing the histogram of stroke slopes for a whole block of text [SP97]. Sun and Si used histogram analysis on gradient oriented grey-level images to detect the slant angle of the characters and further rectify the image using a shear operation [SS97].

Detection of italic words is not only useful in improving OCR performance, but also helpful in automatically retrieving information from the document. This is because many important terms are often printed in italic style and thus the information content is higher in italicized words than in normal ones. Chaudhuri and Garain conducted a statistical study on the relative abundance and importance of italic, bold and all-capital words in 6,000 document pages ranging from technical journals, proceedings of technical conferences to technical books, etc [CG98]. The observations are summarized as follows:

- Almost all paper titles, section or chapter titles are written in Bold style;
- In 49% cases, the paper abstract section is printed in Italic style;
- In 30% cases, the figure or table captions are printed in Bold style and in 19% cases, in Italic style;
- Out of 970 reference sections in those documents, 57% use Italic style for the referred journals, proceedings or publishers; 26% use Italic style for the title of the referred papers or books;
- In 30% cases, the referred author's name is written in Bold cases, and in 5% cases, in Italic style;

4.2 Wavelet Transformation Based Approach

In view of the available techniques for italic font recognition, some are based on image analysis and attribute evaluation of large text blocks. These techniques are not feasible in identifying the font and style of a few words scattered in the document. On the other hand, some methods are based on feature analysis of individual characters, which cause problems when characters are inter-connected in certain distorted document images. Moreover, shape

properties and gradient information of the original slant image are usually subject to font variations such as typefaces, size, serifness, boldness, etc. Therefore, with reference to our word image coding technique that directly extracts individual word features without character segmentation, it is necessary to find an efficient and accurate approach to detect italic words scattered in the document images with less sensitivity to noise and font variations.

4.2.1 Wavelet Decomposition of Word Images

To identify italic words scattered in the normal text, stroke pattern analysis can be applied to each word image to compare with a set of predefined criteria for differentiation. However, stroke patterns obtained from original word images are largely dependent on font styles such as typefaces, size, serifness, boldness, etc. Therefore, to reduce the sensitivity of stroke pattern analysis with respect to font variations as well as noise and distortions, wavelet decomposition is carried out prior to the analysis step to extract dominant word features in horizontal, vertical and diagonal directions for analysis use.

As we mentioned in chapter 2, before applying the italic font recognition technique, each word object has been extracted from the document image. Now for each extracted word image object, a 2-D discrete wavelet transformation (2-D DWT) is performed to extract dominant stroke patterns. According to Heijmans [HG98], wavelet decomposition is based on the pyramid transform.

4.2.1.1 Pyramid Transform

Consider a family $\boldsymbol{V_j}$ of signal spaces. Here, j may range over a finite or an infinite index set.

Assume that we have two families of operators, a family $\varphi_j^{\uparrow}$ of analysis operators mapping $\boldsymbol{V_j}$ into $\boldsymbol{V_{j+1}}$, and a family $\varphi_j^{\downarrow}$ of synthesis operators mapping $\boldsymbol{V_{j+1}}$ back into $\boldsymbol{V_j}$. Here, the upward arrow indicates that the corresponding operator that maps a signal to the higher level, whereas the downward arrow indicates that the operator maps a signal to a lower level. Refer to Figure 4-1 for an illustration. The analysis operator $\varphi_j^{\uparrow}$ is chosen to reduce information from a signal $x_j \in V_j$, yielding a scaled signal $x_{j+1} = \varphi_j^{\uparrow}(x_j)$ in $\boldsymbol{V_{j+1}}$. The synthesis operator $\varphi_j^{\downarrow}$ maps the scaled signal x_{j+1} back to $\hat{x}_j = \varphi_j^{\downarrow}(x_{j+1})$ in $\boldsymbol{V_j}$, in such a way that $\varphi_j^{\downarrow}\varphi_j^{\uparrow}(x_j)$ is "close" to x_j.

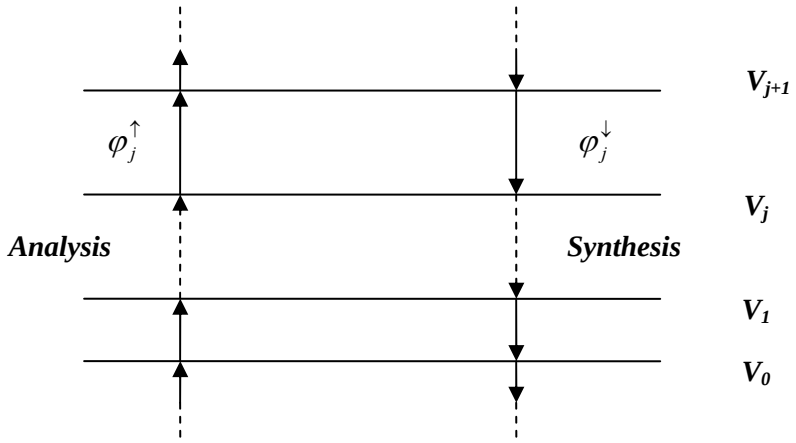


Figure 4-1 The pyramid decomposition scheme

4.2.1.2 Coupled and uncoupled Wavelet Decomposition

The coupled wavelet decomposition extends the pyramid transform scheme. Assume that there exist sets $\boldsymbol{V_j}$ and $\boldsymbol{W_j}$. We refer to $\boldsymbol{V_j}$ as the *signal space at level j* and to $\boldsymbol{W_j}$ as *the detail space at level j*. Signal analysis consists of decomposing a signal in the direction of increasing j by means of *signal analysis operators* $\varphi_j^{\uparrow} : V_j \to V_{j+1}$ and *detail analysis operators* $\omega_j^{\uparrow} : V_j \to W_{j+1}$. On the other hand, *signal synthesis* proceeds in the direction of decreasing j, by means of *synthesis operators* $\psi_j^{\downarrow} : V_{j+1} \times W_{j+1} \to V_j$. When there exists a binary

operation $\oplus$ on $\boldsymbol{V_j}$, which we call *addition* and operators $\varphi_j^{\downarrow} : V_{j+1} \to V_j$ and $\omega_j^{\downarrow} : W_{j+1} \to V_j$ such that $\psi_j^{\downarrow}(x, y) = \varphi_j^{\downarrow}(x) \oplus \omega_j^{\downarrow}(y), \quad x \in V_{j+1}, y \in W_{j+1}$.

We refer to $\varphi_j^{\downarrow}, \omega_j^{\downarrow}$ as the signal synthesis and the detail synthesis operators respectively. This is illustrated in Figure 4-2. The analysis operators $\varphi_j^{\uparrow}$, $\omega_j^{\uparrow}$ and the synthesis operators $\varphi_j^{\downarrow}$, $\omega_j^{\downarrow}$ satisfy conditions similar to the biorthogonality conditions known from the theory of wavelet. In other word, $\varphi_j^{\uparrow}$, $\varphi_j^{\downarrow}$ are referred to as *lowpass* operators and $\omega_j^{\uparrow}$, $\omega_j^{\downarrow}$ are referred to as *highpass* operators [HG00].

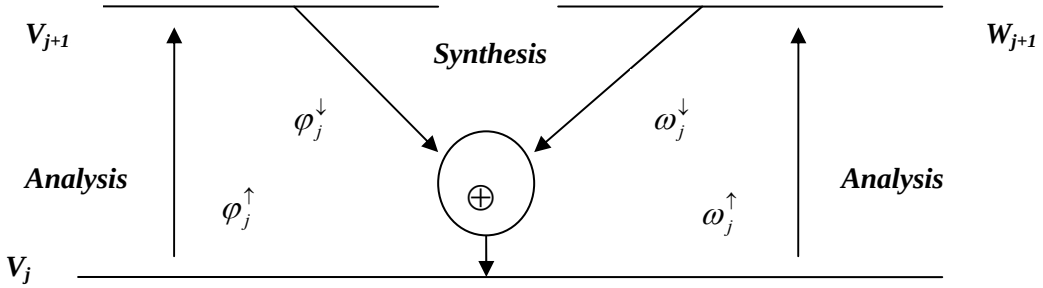


Figure 4-2 One stage of the uncoupled wavelet decomposition scheme

The simplest non-trivial linear uncoupled wavelet decomposition is the *Haar wavelet*. As in the previous definition, choose $\boldsymbol{V_0 = V_1 = W_1 =}$ $\ell^2(\mathrm{Z})$. The analysis operators are defined as:

$$\varphi^{\uparrow}(x)(n) = \frac{1}{\sqrt{2}}\big(x(2n) + x(2n+1)\big)$$
$$\omega^{\uparrow}(x)(n) = \frac{1}{\sqrt{2}}\big(x(2n) - x(2n+1)\big)$$

The 1-D Haar wavelet decomposition scheme can be easily extended to two and higher dimensions by using a separable filter bank, e.g. by sequentially applying the 1-D decomposition on the columns and rows of a 2-D image [M98]. We can also define a

non-separable 2-D version of the Haar wavelet. Let ***n, 2n*** denote the points *(m, n), (2m, 2n)*, and $2n^{+},2n_{+},2n_{+}^{+}$ denote the points *(2m, 2n+1), (2m+1, 2n)* and *(2m+1, 2n+1)* respectively, then we have:

$$\varphi^{\uparrow}(x)(n)=\frac{1}{2}\left(x(2n)+x(2n^{+})+x(2n_{+})+x(2n_{+}^{+})\right) \Rightarrow cA_{j+1}$$

$$\omega_{h}^{\uparrow}(x)(n)=\frac{1}{2}\left(x(2n)-x(2n_{+})+x(2n^{+})-x(2n_{+}^{+})\right) \Rightarrow cD_{j+1}(h)$$

$$\omega_{v}^{\uparrow}(x)(n)=\frac{1}{2}\left(x(2n)-x(2n^{+})+x(2n_{+})-x(2n_{+}^{+})\right) \Rightarrow cD_{j+1}(v)$$

$$\omega_{d}^{\uparrow}(x)(n)=\frac{1}{2}\left(x(2n)-x(2n^{+})-x(2n_{+})+x(2n_{+}^{+})\right) \Rightarrow cD_{j+1}(d)$$

Various wavelet filters such as Haar, Daubechies and Symlets can be employed in a 2-D Discrete Wavelet Transformation (2-D DWT) step [M89]. Essentially, the transformation step decomposes the original word image into an approximation sub-image and three detailed sub-images in vertical, horizontal and diagonal directions. Experiments show that one-level decomposition using symlet of order two (sym2) works particularly well in extracting vertical, horizontal and diagonal stroke patterns of word images. Sym2 employs a low-pass filter with coefficients [-0.1294, 0.2241, 0.8365, 0.4830] and a high-pass filter with coefficients [-0.4830, 0.8365, -0.2241, -0.1294] as illustrated in Figure 4-3.

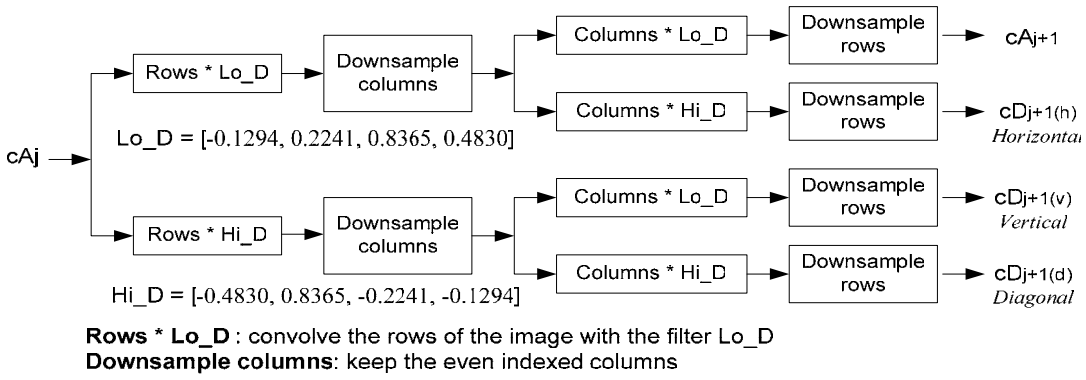


Figure 4-3 Two dimensional Discrete Wavelet Decomposition

Figure 4-4 shows an example of the one-level wavelet decomposed sub-images in horizontal, vertical and diagonal directions. The sample word image “European” is of size 37×154 pixels and is extracted from scanned paper document with noise and distortions. It is obvious that vertical strokes come out especially in the vertical channel, horizontal strokes in the horizontal channel, and diagonal strokes in the diagonal channel. These are strong and distinguishable features of a typical word image, which are less sensitive to distortions and font style variations such as size, serifness, boldness, etc.

Figure 4-4 An example of one-level wavelet decomposed sub-images

4.2.2 Statistical Analysis of Stroke Patterns

The sub-images generated from the wavelet decomposition contain ample information in terms of vertical, horizontal and diagonal stroke patterns of the word image. This can be

effectively utilized to distinguish italic and non-italic fonts. In particular, statistical analysis of both vertical and diagonal stroke patterns is performed on the corresponding sub-images generated above and is combined to produce a discriminative recognition measure.

4.2.2.1 Vertical Stroke Analysis

It is observed through our experiments among 22,384 frequently used English words that in over 99% of normal word images and over 14% of italic word images, at least two vertical straight line segments (VSLS) would run through the mid zone, as indicated by the arrows in Figure 4-5(a) and (b). Some even go up to the ascender zone or down to the descender zone. These are distinctive features between italic and non-italic styles. Therefore, by analyzing the horizontal histogram of the combined horizontal and vertical sub-images, the mid zone is detected and VSLSs are identified. Suppose a word of width $\boldsymbol{W}$ (in pixels) has $\boldsymbol{N}$ VSLS with height $(h_1, h_2, \cdots, h_N)$ respectively, their normalized total height $\boldsymbol{H}$ is obtained as follows:

$$H = \frac{1}{W}\sum_{i=1}^{N} h_i$$

The total height is used because it carries higher weights for longer VSLSs. Experiments show that over 99% of the normal word images satisfy the criterion that $\boldsymbol{N \geq N_0}$ and $\boldsymbol{H \geq H_0}$, where $\boldsymbol{N_0}$ and $\boldsymbol{H_0}$ are predefined thresholds with experimental values 2 and 1.6 respectively. However, some italic words such as those containing characters ‘w’, ‘v’ or ‘y’ might also satisfy this criterion. To distinguish these words, diagonal stroke analysis is then taken into consideration.

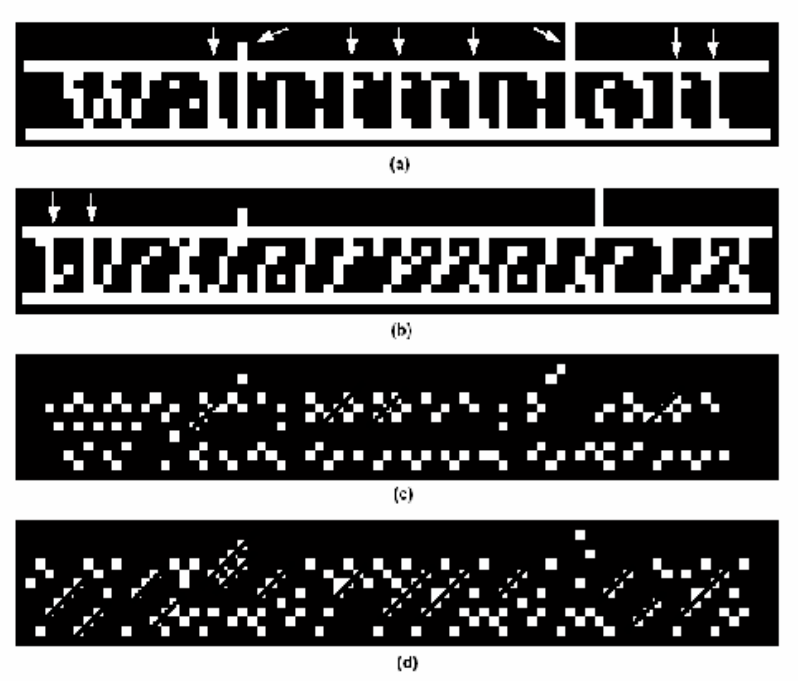


Figure 4-5 (a)(b) VSLS running through the mid zone for normal and italic styles respectively (c)(d) CDS for normal and italic styles respectively (length ≥ 3)

4.2.2.2 Diagonal Stroke Analysis

It is observed that italic font produces a great number of long continuous diagonal strokes (CDS) in the diagonal sub-image comparing to the normal font, as illustrated in Figure 4-5(c) and (d). The normal font of "watermelon" produces 4 CDSs with length greater than two while the italic font produces 14 such CDSs. Suppose $\boldsymbol{M}$ is the number of CDSs with length $(l_1, l_2, \cdots, l_M)$ respectively, their normalized total length $\boldsymbol{L}$ is obtained as follows:

$$L = \frac{1}{W}\sum_{i=1}^{M} l_i$$

Experiments show that over 98% of italic word images satisfy the criterion that $\boldsymbol{M} \geq \boldsymbol{\mu}$ and $\boldsymbol{L} \geq \boldsymbol{L_0}$, while only about 2% of normal word images satisfy this criterion. Here, $\boldsymbol{\mu}$ and $\boldsymbol{L_0}$ are predefined threshold with value 3 and 0.33 respectively. Therefore, by combining the vertical

stroke analysis and the diagonal stroke analysis, a set of statistics is obtained to effectively differentiate between italic and non-italic fonts.

4.2.3 Experimental Results

Experiments have been carried out to test the proposed method with 22,384 frequently used word images in both italic and non-italic styles for four different fonts (Times New Roman, Arial, Courier and Comic Sans MS). For simplicity, the word images are computer-generated 256-color bitmap images in various sizes. An example of wavelet decomposed sub-images generated for the word image “Client” in the four different fonts of size 12pt are shown in Figure 4-6.

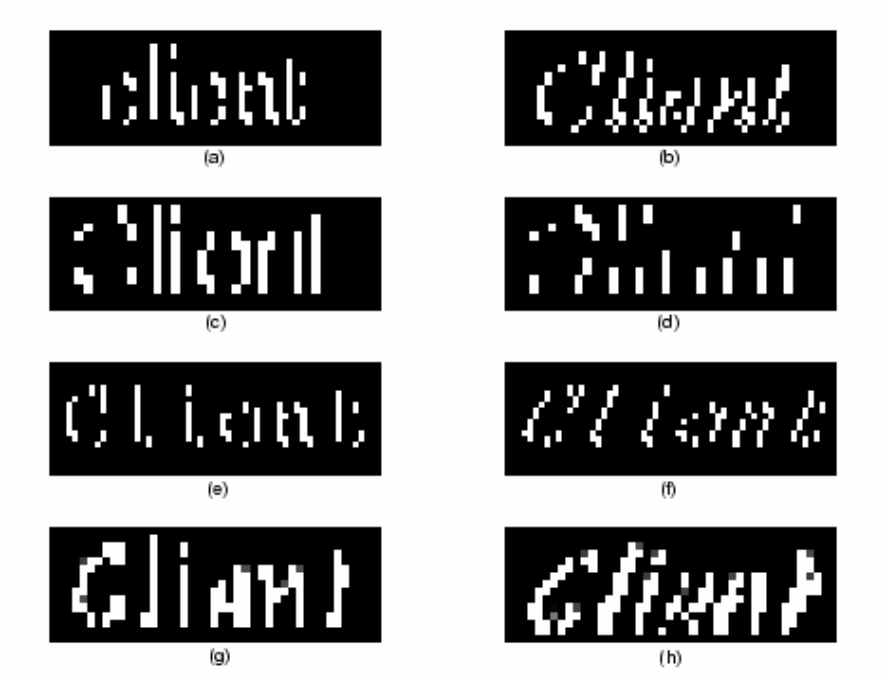


Figure 4-6 Examples of wavelet decomposed vertical sub-image in normal and italic styles (a)(b) Times New Roman (c)(d) Arial (e)(f) Courier (g)(h) Comic Sans MS

Comparisons between the traditional stroke analysis method and our wavelet transformation based approach are carried out with experiments conducted on documents with mixed normal

and italic words in four commonly used fonts. The average recognition accuracies are shown in Figure 4-7. Experiments are also conducted on 5,320 normal word images and 489 italic images extracted from scanned paper documents. The accuracies achieved are 92.20% for normal style and 97.96% for italic style, as shown in Figure 4-7.

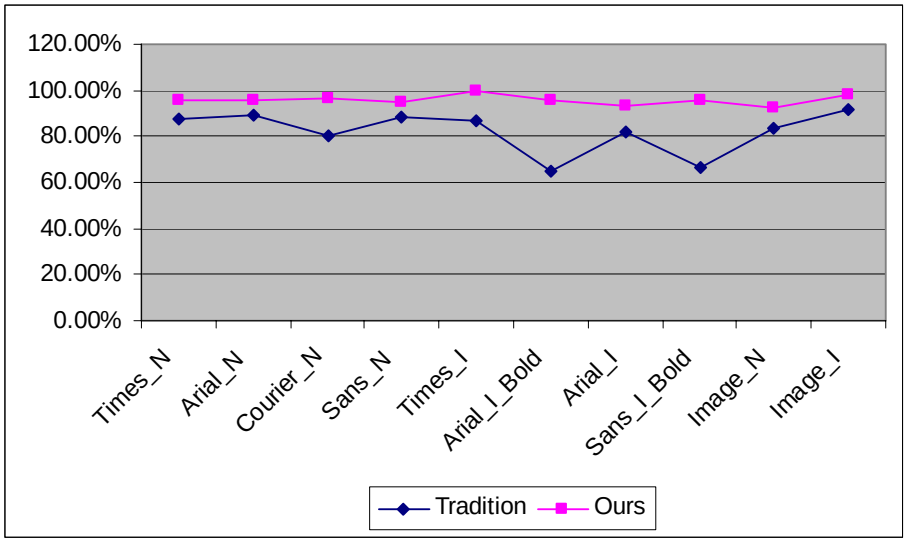


Figure 4-7 Recognition accuracy comparisons between traditional stroke analysis method and our method

Chapter 5

Feature Code Matching

For a word image in a printed text, two characters could be separated apart by a few white columns caused by inter-character spaces in general. On the other hand, it is also common that one character overlaps with another by a few columns caused by kerning. Things may become even worse, when two or more adjacent characters touch each other due to condensed spacing. This poses a challenge to separate such touching characters. Nevertheless, we utilize an inexact feature string matching algorithm to resolve this problem.

5.1 Coarse Matching

As the name indicates, coarse matching is essentially a refinement step, which serves the purpose of restricting the number of word objects to be compared in the feature code files to a smaller range. This effectively speeds up the subsequent inexact string matching algorithm. The main criterion used in coarse matching is the word's x-feature and the number of primitive codes in the feature code string. The x-feature is defined as follows:

- 0: there is neither ascender nor descender in the word;
- 1: there is ascender but no descender in the word;
- 2: there is no ascender but descender in the word;
- 3: there are both ascender and descender in the word.

In consideration of the case where a word object could contain portion of the query word and should also be selected as a match, e.g. the word "unhealthy" should be matched if the query

word is “health”, the following criteria are used for coarse matching:

- If the x-feature of the query word is 0, all the word objects in the document image are possibly matched;
- If the x-feature of the query word is 1, only the words whose x-features are either 1 or 3 are possibly matched;
- If the x-feature of the query word is 2, only the words whose x-features are either 2 or 3 are possibly matched;
- If the x-feature of the query word is 3, only the words whose x-features are 3 are possibly matched.

For example, if the query word is “health”, any word objects without ascenders (i.e. x-feature is 0 or 2) will be ruled out for the next step.

Next, the number of the primitive codes in the feature code string is used to eliminate more words. Suppose the number of the query word’s primitive codes is $\boldsymbol{N_Q}$, and the number of a word object’s primitive codes is $\boldsymbol{N_W}$. The ratio of $\boldsymbol{N_W}$ to $\boldsymbol{N_Q}$ is considered. If $\boldsymbol{N_W/N_Q} < \delta$, where δ is a predefined threshold (e.g. 0.8 in our experiments), the word object will be eliminated for further matching since the word object is too short to match with the query word.

5.2 Inexact String Matching

Following the coarse matching step, an inexact string matching algorithm is applied to measure the similarity between two primitive code strings. The string matching problem can be stated as finding a particular sequence/sub-sequence in the primitive code string of a word object. The procedure of matching word images hence becomes a measurement of the similarity between the string $A =< a_1, a_2, \cdots, a_n >$ representing the features of the query

word and the string $B =< b_1, b_2, \cdots, b_m >$ representing the features of a word object extracted from the document image. Matching partial words becomes evaluating the similarity between the feature string ***A*** and a sub-sequence of the feature string ***B***. For example, the problem of matching the word “health” with the word “unhealthy” is to find whether there exists a sub-sequence closest to ***A***=<dnm&ceo&oem&d&ndo&dnm> in the primitive code sequence of the word “unhealthy”.

In a word image, it is common that two or more adjacent characters are connected to each other. It is possibly caused by low scanning resolution or poor printing quality. This results in the deletion of the feature <&> in the corresponding feature code string comparing to the primitive string generated from the standard PSTs of the query word. Moreover, noise effect also produces substitution or insertion of features in the primitive string of the word image. The deletion, insertion and substitution are very similar to the course of evolutionary mutations of DNA sequences in molecular biology [AG99].

Lopresti and Zhou applied the inexact string matching strategy to information retrieval [LZ96] and duplicate document detection for dealing with imprecise text data generated from OCR [L01]. Drawing inspiration from the alignment problem of two DNA sequences and the research done by Lopresti and Zhou, we apply the technique of inexact string matching to evaluate the similarity between two primitive code strings, one from the input query word and the other from the word image extracted from the document image.

Informally, an alignment of two strings *A* and *B* is obtained by first inserting chosen spaces, either into or at the ends of *A* and *B* so the length of the strings will match, and then placing

the two resulting strings one above the other so that every character or space in one of the strings is matched to a unique character or a unique space in the other string [G97]. Specifically speaking, two differing features that mismatch correspond to a substitution; a space in the first string corresponds to an insertion of the extra feature into the second string; and a space in the second string corresponds to a deletion of the extra feature from the first string. A dash '-' is used to represent a space primitive inserted into the corresponding positions of the strings in the case of deletion.

Now the problem we are going to solve is: Given two feature code strings A and B of length n and m respectively, establish the optimal alignment according to the weight $\delta(a_i, b_j)$ assigned to the alignment of character a_i and b_j (including spaces). Here, we define $V(i,j)$ to be the optimal score of aligning the prefixes $[a_1, a_2, \ldots, a_i]$ and $[b_1, b_2, \ldots, b_j]$. Then the optical score of aligning A and B is precisely the value $V(n,m)$.

The optimal alignment score of two strings A and B can be computed by a dynamic programming with recurrences. The base conditions are:

$$\forall i, j : \begin{cases} V(i,0) = \delta(a_i, -) \\ V(0,j) = \delta(-, b_j) \end{cases}$$

$\delta(a_i, -)$ is defined to be the matching value between the ith element of A and the space character, since string B is empty. Similarly, $\delta(-, b_k)$ is defined to be the matching value between the jth element of B and the space character.

The general recurrence relation is:

$$V(i,j) = max\begin{cases} 0 \\ V(i-1, j-1) + \delta(a_i, b_j) \\ V(i-1, j) + \delta(a_i -) \\ V(i, j-1) + \delta(-, b_j) \end{cases} \qquad (1)$$

The zero in the above recurrence implements the operation of *restarting* the recurrence, which ensures that the unmatched prefixes are discarded from the computation. The following three operations can be interpreted as follows:

- Aligning A_i with B_j: the score in this case is the score of the operation $\delta(a_i, b_j)$ plus the score of aligning *i-1* elements of *A* with *j-1* elements of *B*.
- Aligning A_i with a space character in string *B*: the score in this case is the score of the operation $\delta(a_i, -)$ plus the score of aligning the previous *i-1* elements of *A* with *j* elements of *B* (i.e. a space character is inserted into string *B*).
- Aligning B_j with a space character in string *A*: similar to the previous case, a space character is inserted to string *A*.

Following the above recurrence relation, a table can be constructed to evaluate the optimal matching score of string *A* and *B*. Each table entry records the optimal matching score for the corresponding prefixes. The table is constructed starting from the upper-left corner and increasing in a row-wise manner. The following pseudo code describes the algorithm:

```
for j=1 to m do
begin
    for i=1 to n do
    begin
```

Calculate V(i, j) using V(i-1, j-1), V(i-1, j), V(i, j-1)

end

end

Finally, the maximum scoring is normalized as:

$$score = \max_{\forall i,j} V(i,j) \Big/ V_A^*(n,n) \qquad (2)$$

where $V_A^*(n,n)$ is the matching score between the string A and itself. This can be obtained by generating a table with two identical feature code strings as row and column attributes. The maximum operation in Equation (2) and the restarting recurrence operation in Equation (1) ensure the partial matching.

If the score is greater than a predefined threshold λ, then we recognize that the word image (or its portion) matches the user-specified query word.

On the other hand, the similarity of two entire words, i.e. without portion matching, is calculated as:

$$S_2 = V(n,m) \big/ \min\left(V_A^*(n,n), V_B^*(m,m)\right)$$

The problem can be evaluated systematically using a tabular computation. In particular, a bottom-up approach is used to compute ***V(i, j)*** with ***i*** and ***j*** starting from the smallest values up to ***n*** and ***m*** respectively. This computation ends up with a table of size (n+1) × (m+1). The table holds the values of ***V(i, j)*** at different values of ***i*** and ***j*** (as shown in Table 5-1). The values in row zero and column zero are filled in directly from the base conditions of ***V(i, j)***.

Then, the remaining n × m cells are filled in one row at a time in the order of increasing ***i***. For each row ***i***, the cells are filled in the order of increasing ***j***. Table 5-1 shows the table computed for the primitive string of the word image “unhealthy” extracted from the document image and the primitive string of the query word “health”. It is observed that the maximum score obtained from the table corresponds to the matching of character sequence “health” in the word “unhealthy”. This shows that the partial matching property actually simulates the function of word stemming which is normally performed in the text retrieval approaches. It is also feasible to reinforce the functionality of word stemming by translating the stemming rules into their corresponding LPRS representations for the refinement of the LPRS representation during the feature code generation step. Computing the entire table using dynamic programming for two strings of length n and m can be done in ***O(nm)*** time, since only three arithmetic operations and comparisons are needed for each cell.

The match may be imprecise in the sense that certain primitives are missing or miscoded. As we mentioned earlier, some adjacent characters in a word image may be connected to each other due to various reasons. This results in less number of inter-character spaces being detected and thus less number of spacing primitives <&> being translated in the primitive string. Looking again at table 5-1, we find the best matching sequence by backtracking from the maximum score obtained. Our experiments show that the maximum score obtained decreases if there are missing spacing primitives in the primitive string of the word image. To remedy this problem, we modify the scoring function to take the missing spaces into consideration as follows:

$$S_m = \left(\max_{\forall i,j} V(i,j) + \tau(N_g) \right) \Big/ V_A^*(n,n)$$

where N_g is the number of missing spacing primitives included in the backtracking sequence.

In our experiments, we have $\tau(N_g) = 2 \times N_g$.

				un							h		e		al				th				y							
			m	u	m	u	o	m	n	m	&	d	o	m	&	c	e	o	&	o	e	m	d	&	n	d	o	d	n	m	&	y
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	d	0	0	0	0	0	0	0	0	0	**2**	1	0	0	0	0	0	0	0	0	0	2	1	0	2	1	2	1	0	0	0
h	n	0	0	0	0	0	0	0	2	1	0	1	**2**	1	0	0	0	0	0	0	0	0	1	1	3	2	2	1	4	3	2	1
	d	0	2	1	2	1	0	2	1	4	3	2	1	**4**	3	2	1	0	0	0	0	2	1	0	2	3	2	2	3	6	5	4
	&	0	1	1	1	1	0	1	1	3	6	5	4	3	**6**	5	4	3	2	1	0	1	1	3	1	2	2	1	2	5	8	7
	c	0	0	1	1	1	1	0	1	1	5	4	5	4	5	**8**	7	6	5	4	3	2	1	2	3	2	2	1	1	4	7	6
e	e	0	0	0	1	1	1	1	0	1	4	3	4	5	4	7	**10**	9	8	7	6	5	4	3	2	1	2	1	1	3	6	5
	o	0	0	0	0	1	3	2	1	0	3	2	5	4	4	6	9	**12**	11	10	9	8	7	6	5	4	3	2	1	2	5	4
	&	0	0	0	0	0	2	2	1	0	2	2	4	4	6	5	8	11	**14**	13	12	11	10	9	8	7	6	5	4	3	4	4
	o	0	0	0	0	0	2	2	2	1	1	1	4	4	5	6	7	10	13	**16**	15	14	13	12	11	10	9	8	7	6	5	4
a	e	0	0	0	0	0	1	2	2	2	1	0	3	4	4	5	8	9	12	15	**18**	17	16	15	14	13	12	11	10	9	8	7
	m	0	2	1	2	1	0	3	2	4	3	2	2	5	4	4	7	8	11	14	17	**20**	19	18	17	16	15	14	13	12	11	10
	&	0	1	1	1	1	0	2	2	3	6	5	4	4	7	6	6	7	10	15	16	19	19	21	20	19	18	17	16	15	14	13
l	d	0	0	0	1	0	0	1	1	2	5	8	7	6	6	5	5	6	9	14	15	18	**21**	20	19	22	21	20	19	18	17	16
	&	0	0	0	0	0	0	0	0	1	4	7	7	6	8	7	6	5	8	13	14	17	20	**23**	22	21	20	20	19	18	20	19
	n	0	0	0	0	0	0	0	2	1	3	6	7	7	7	8	7	6	7	12	13	16	19	22	**25**	24	23	22	22	21	20	19
	d	0	0	0	0	0	0	0	1	2	2	5	6	7	6	7	6	5	6	11	12	15	18	21	24	**27**	26	25	24	23	22	21
t	o	0	0	0	0	0	2	1	0	1	1	4	7	6	6	6	7	8	7	10	11	14	17	20	23	26	**29**	28	27	26	25	24
	&	0	0	0	0	0	1	1	0	0	3	3	6	6	8	7	6	7	7	9	10	13	16	19	22	25	28	28	27	26	28	27
	d	0	0	0	0	0	0	1	0	0	2	5	5	6	7	6	5	6	6	8	9	12	15	18	21	24	27	**30**	29	28	27	26
h	n	0	0	0	0	0	0	0	3	2	1	4	4	5	6	7	6	5	5	7	8	11	14	17	20	23	26	29	**32**	31	30	29
	m	0	2	1	2	1	0	2	1	2	1	3	3	6	5	6	7	6	5	6	7	10	13	16	19	22	25	28	31	**34**	33	32

Table 5-1 Scoring table and missing space recovery

Chapter 6

Web-based Document Image Retrieval System

In this chapter, we detail the implementation of the first application of our word image coding technique, namely the web-based document image retrieval system given a set of query words [ZLT03]. First, we give a structural overview of the retrieval system (shown in Figure 6-1) and briefly describe how the various components are combined to make the system an efficient web application for digital libraries. Then, we detail the implementation and the AND/OR/NOT operations supported by the query statement.

6.1 System Overview

First of all, some image processing procedures are carried out on the original document images embedded in the PDF files. These include connected component detection, word object bounding, skew estimation and rectification (if applicable), etc. After each word image object is extracted from the document image, italic font detection algorithm is applied to check for italic property and rectification is done before generating the corresponding feature codes using the word image coding technique illustrated in Chapter 3. For each PDF file, a feature code file is generated, containing the URL of the corresponding PDF file and the information of each word object such as its location, feature codes and so on. This feature code file is stored on a server and is used during the feature code matching step discussed in Chapter 5. All the above image processing operations are carried out off-line prior to the online search process, which reduces the query response time during retrieval.

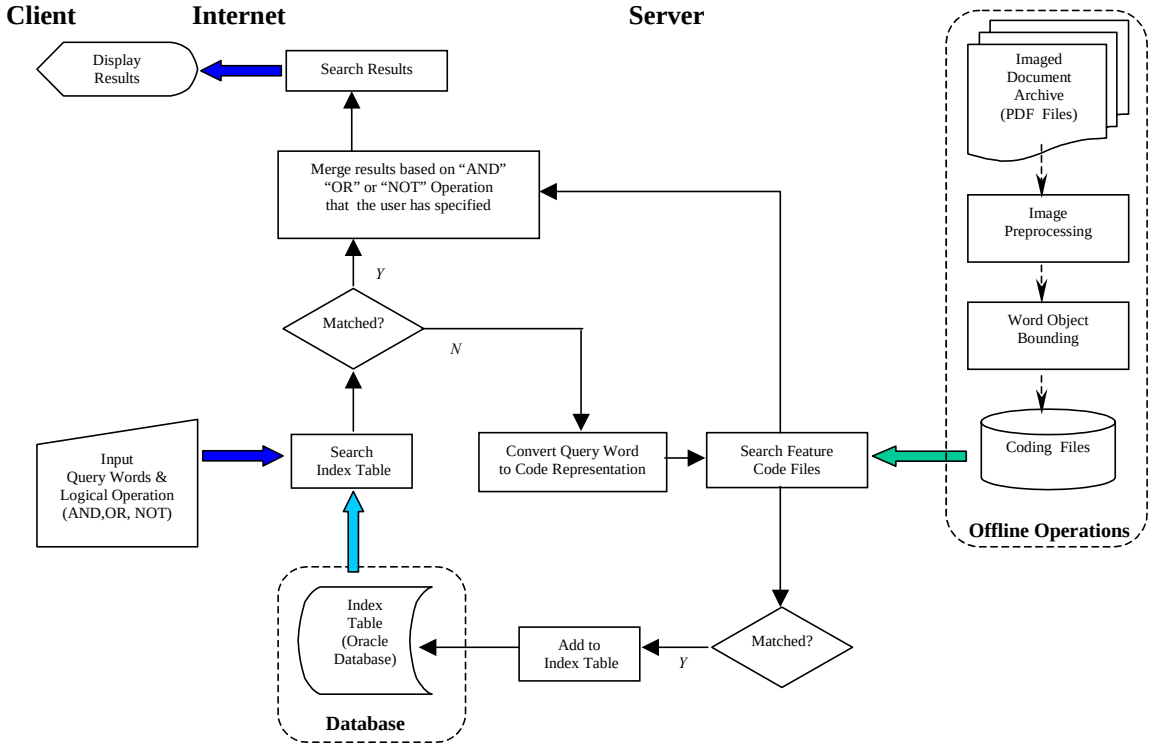


Figure 6-1 Overview of the web-based document image retrieval system

On the client side, users can input a set of query words and choose to perform AND/OR/NOT operations among these query words. Once the request is submitted to the server, the server will start processing each query word and merge the search result at the end of each iteration based on the logical operations chosen. Finally, a temporary result table that stores all the matching documents with their URLs and the normalized occurrence frequencies of the query words will be returned to the user for display. Users can then link to the actual document and check out the exact locations of the query words with the help of the plug-in search tool we embedded in Acrobat as to be described in Chapter 7.

As for the processing of each query word, it is done as follows: First, the server tries to search for the query word in an index table stored in the oracle database. This index table is used to store information of the words that have previously been searched and succeeded. Hence, if

there are matches, information of the corresponding documents that contain this query word will be retrieved directly from the index table and stored in a temporary table for subsequent merging. This information includes the documents' URLs as well as the normalized occurrence frequency of the query word in each of these documents. Otherwise, if no matches are found in the index table, we generate the feature codes of the query word by matching each character with a standard feature code string as is defined based on the word image coding mechanism. Then, an inexact string matching algorithm is employed to perform feature codes matching in the underlying feature code files stored on the server.

With the purpose of constructing an incremental intelligent system to speed up the retrieval process for subsequent searches, the results of earlier queries are stored in the index table for efficient retrieval for future queries. If there are newly found matches, the index table will be updated accordingly by adding the corresponding information related to the current query word such as the URL of the matching document and the normalized occurrence frequency.

6.2 System Implementation

An experimental platform of the proposed web-based document image retrieval system has been implemented.

- Its web interface is developed in Active Server Pages (ASP) hosted on Microsoft Internet Information Server (IIS) 5.1. It allows the user to input a set of query words and perform AND/OR/NOT operations among them.
- Operations such as feature code matching described in Chapter 5 are implemented as COM Component using C++ so that the matching functions can be used inside the ASP program.

- The index table containing information about queried words is stored on an Oracle database server. A snapshot of a portion of the index table is given in Table 6-1. The corresponding three fields are the keyword, the URL of the document containing this keyword and the normalized occurrence frequency of this word in the current document. The occurrence frequency is normalized so that for those rare words (usually more meaningful and significant), the frequency value will be significant enough to be represented using a limited number of digits. In particular, when "AND" operation is performed, the frequency values will be multiplied together to obtain the new frequency value. This may result in a very small percentage value. Therefore, it is necessary to perform some normalization on the frequency value, for example, scaling by 10 times.
- The temporary result table used for merging among the intermediate search result for each query word is also stored in the Oracle server. It contains only two fields: the URL of the document that contains the query words up to the current round and the corresponding occurrence frequency.
- The off-line preprocessing operations described in Chapter 2 are implemented using C++, which include connected component analysis, skew detection, skew rectification, italic detection, italic rectification and feature code generation.
- Lastly, the original document images and their corresponding underlying feature code files are stored on a server for the use of feature code matching.
- Totally 478 document image files provided by the digital library of our university are included in the test. These document images were scanned from the published conference papers, earlier students' theses and packed in PDF files. Each of them contains about 100 to 200 pages.

A brief system workflow is as follows: On the client side, the user inputs a set of query words through a web interface and meanwhile indicates AND/OR/NOT operations to be performed among the query words. Next, on the server side, each query word will be processed by the

server as follows: First, it will be looked up in an index table stored in the Oracle database. If there are exact matches, the corresponding entries will be retrieved directly and stored in a temporary table for subsequent merging; otherwise, the system will convert the query word into feature code string and match it with the feature code strings in the feature code files pre-generated. If there are newly found matches, the index table will be updated accordingly. Moreover, a result-merging step will be carried out at the end of each query word processing step based on the AND/OR/NOT operation the user has chosen. In the end, the user will obtain a list of matching documents with their URLs, and the normalized occurrence frequency of the query words appearing in it. He/She can then link to the actual document images for further reading and verification.

Keyword	Document URL	Frequency (%)
…	…	…
approach	Soccf-chim3-003.comp.nus.edu.sg/ASP/test/1.pdf	0.012
assembly	Soccf-chim3-003.comp.nus.edu.sg/ASP/test/1.pdf	0.234
assembly	Soccf-chim3-003.comp.nus.edu.sg/ASP/test/2.pdf	0.143
assembly	Soccf-chim3-003.comp.nus.edu.sg/ASP/test/3pdf	0.003
…	…	…

Table 6-1 A snapshot of the index table storing information of queried words

6.3 AND/OR/NOT Operations

Basically our system supports all AND/OR/NOT operations over a set of query words. Users are prompted through a web interface to input a set of query words separated by an empty space and then choose to perform AND/OR operation on them, then followed by a set of query words that should not be included in the resulting documents. The “NOT” operation is performed after the AND/OR operations, which removes those documents that contain those

words specified in the “NOT” query input box.

6.3.1 AND Operation

Generally speaking, if the “AND” operation is chosen, the system will do as follows: It starts from the first word, processes it to obtain a result table and stores the table temporarily in the Oracle database. It then joins this table with the resulting record set of each subsequent round to obtain a new result table. In this manner, at the end of each round, the result table will store information of the documents that contain all the query words up to now and the multiplication of their corresponding normalized frequencies appearing in this document. In the end, only those documents that contain all the specified query words will be left in the result table for merging with the subsequent result of “NOT” operation. However, in the case where no single document contains all the query words, the search will stop right at the round when either the result table or the current record set is empty. Hence, no time-wasting search will be performed for the subsequent words. The user will be notified that no image documents are found that match the query input expression.

To be more specific, let's consider an example “AND” operation on “approach analysis assembly technique”. Suppose there are five underlying documents in total that we will perform our search on, namely 1.pdf, 2.pdf, 3.pdf, 4.pdf and 5.pdf. If “approach” is contained in 1, 2 and 3; “analysis” is contained in 2, 3 and 4; “assembly” is contained in 5; “technique” is contained in 3. Figure 6-2 shows the result table at the end of each round and also the merging process.

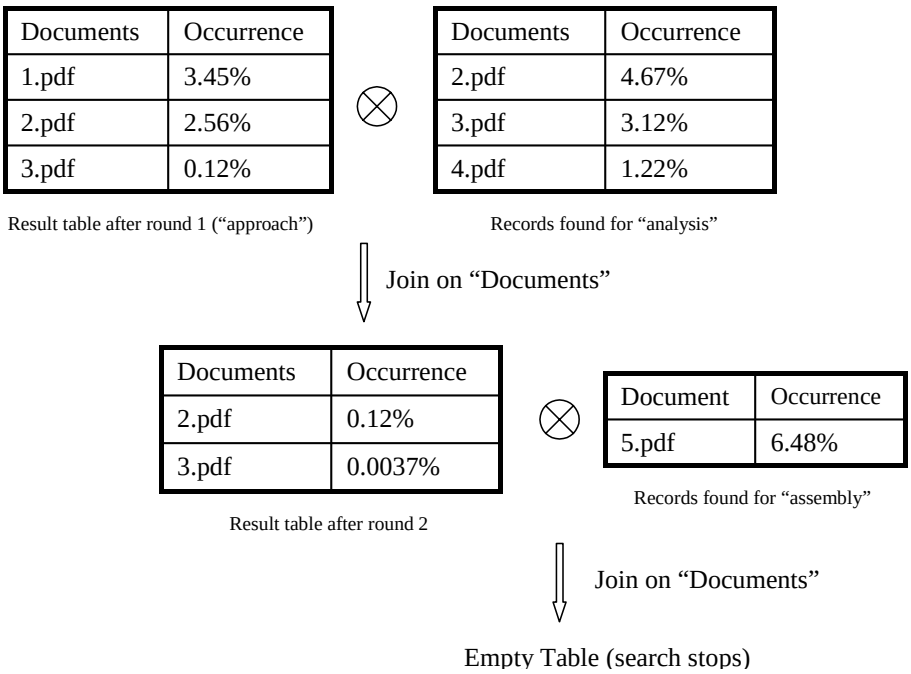


Figure 6-2 AND operation

As we can see from the figure, both “approach” and “analysis” are contained in 2.pdf and 3.pdf. Therefore, after round 2, the result table will only have two entries. Moreover, the corresponding normalized frequency is the multiplication of the normalized frequencies that each of these two words appears in this document. Subsequently, we obtain the set of documents that contain “assembly” and join them with the result table after round 2. Since the join operation is performed on “Documents” field, there is no common file that contains all the first three words. So after round 3, the result table is empty and the search will stop here without further search on the last word “technique”. Finally, the user will be informed that no documents are found for the current input query.

6.3.2 OR Operation

Similarly, for the “OR” operation, the procedures are the same except for the merging step. In this case, we will do a union instead of join. That is, the new result table will contain all the

documents that appear either in the previous result table or in the current record set. Moreover, the normalized frequency will be the summation of the respective normalized frequencies if two words both appear in the same document. Figure 6-3 shows the search process for the same example above under the "OR" operation.

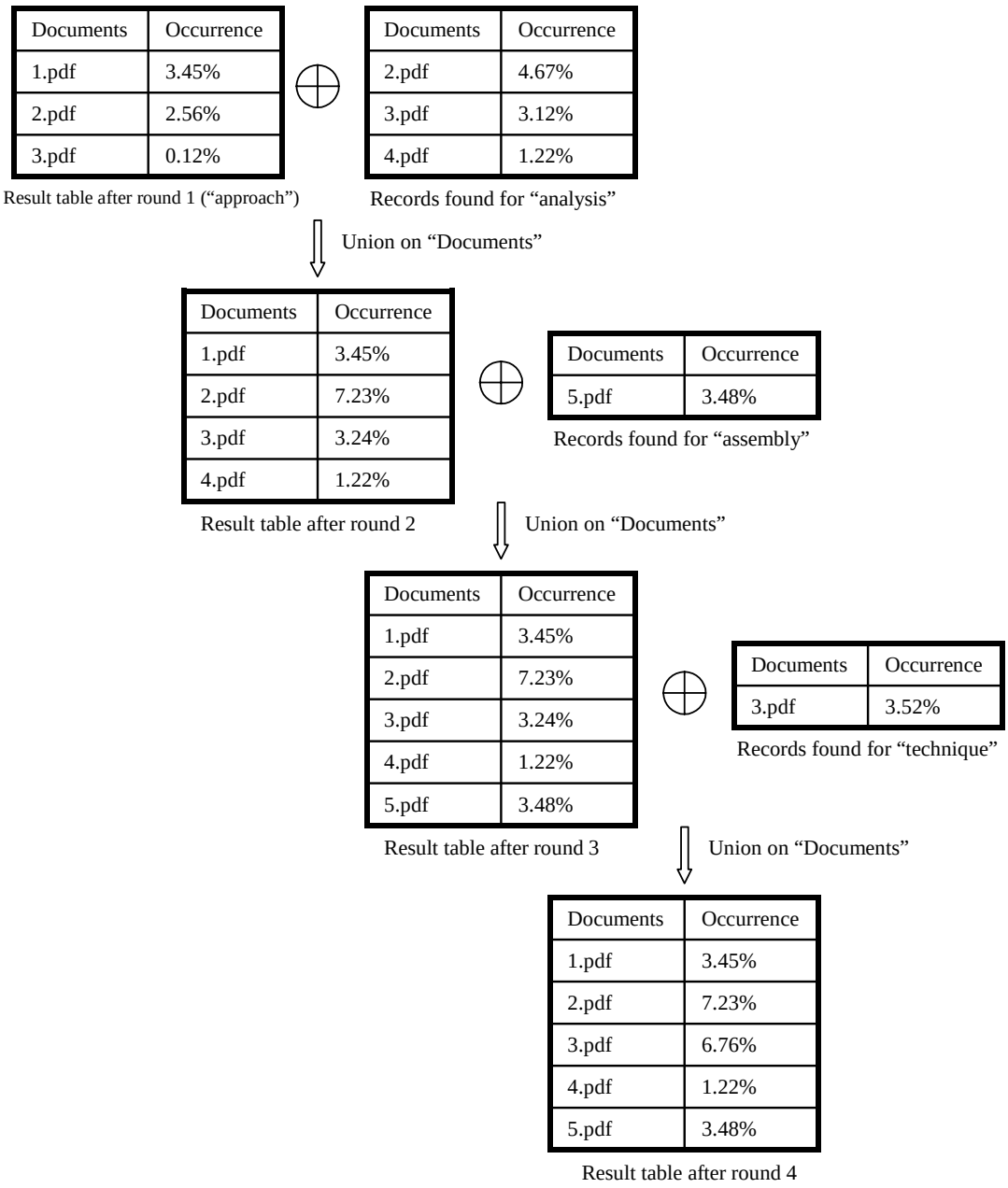


Figure 6-3 OR operation

6.3.3 NOT Operation

The “NOT” operation is carried out after the AND/OR operations. In particular, after AND/OR operations, we will get a list of matching documents stored in a temporary result table. If the result table is not empty and there are some more query words in the “NOT” input box, we will go on processing each of these words and remove those common documents that contain these words not to be included. Finally, the result table will contain all those documents that satisfy the user’s input query. We can then directly retrieve those corresponding information and return them back to the user for display. Figure 6-4 shows an example of the “NOT” operation.

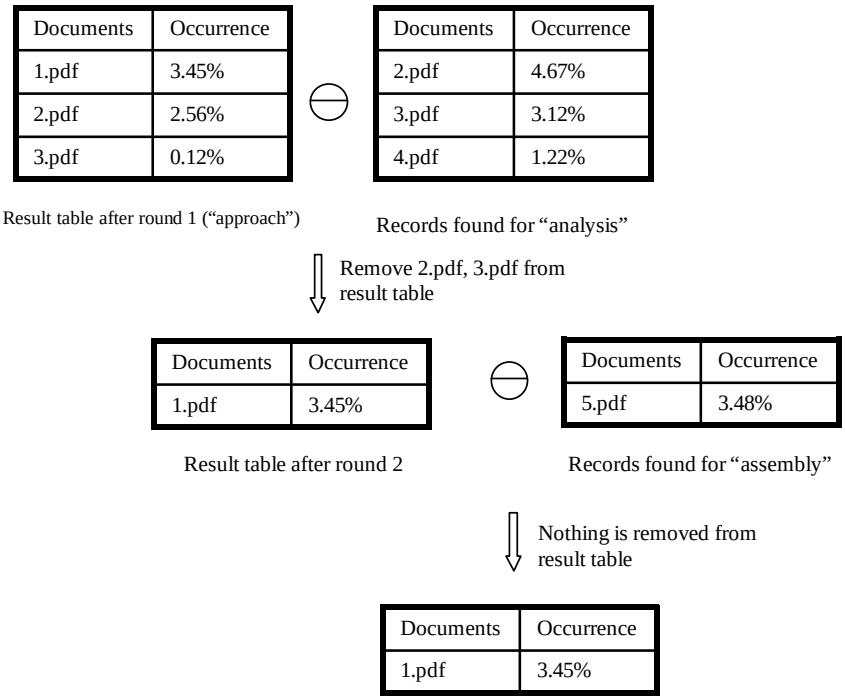


Figure 6-4 NOT operation

6.4 System Evaluation

In order to develop a good retrieval system, it is important to be able to evaluate the overall

system performance and the performance of each system component separately. Basically, there are two large components to consider for evaluating our document image retrieval system based on word image coding technique, i.e. Preprocessing and Online Retrieval.

Preprocessing involves five steps: connected component analysis, word bounding box identification, skew detection and rectification, italic font detection and rectification, feature code generation. Due to its computationally intensive nature, this is done off-line prior to the online search process. The underlying document images provided by our digital library are first fed into an automatic system to undergo all these preprocessing procedures and produce the feature code files. The feature code files are then stored on a server for the use of online matching. Experiments are conducted to evaluate each of the embedded algorithms, such as connected component analysis, word image coding and italic font detection as illustrated in Chapter 2, 3 and 4 respectively. The connected component analysis encompasses the process of merging overlapping components and removing noise and punctuation components. This reduces the noise affection in the feature code generation step. The word image coding technique is tested over a standard dictionary containing 25,133 frequently used English words and shown to be a unique and efficient coding scheme with less sensitivity to font and style variations. The italic font detection approach based on wavelet transformation is also evaluated over datasets containing both computer generated word images and distorted images extracted from scanned paper documents. The results show an advantage in identifying scattered italic fonts in different typefaces and sizes at the word level.

The performance of Online Retrieval is evaluated based on three measured parameters:

precision, recall and retrieval speed. Figure 6-5 shows the recall and precision obtained for different sets of scanned document images with queries on different fonts and styles. There are totally 478 documents ranging from scanned conference papers to students' theses provided by the digital library of NUS. In general, our retrieval system achieves a very good performance in terms of recall and precision. Regarding the retrieval speed, our system records the elapsed time from the point when the user specifies the query words until he/she gets the retrieved results. Generally speaking, there are two scenarios to consider. One is when all the input query words have been queried before. In this case, all the corresponding information about the query words is already stored in the index table. The time to search for these words is merely to retrieve the corresponding entries from the Oracle database and hence trivial (usually less than 0.1 second for each word). An example of this scenario is shown in Figure 6-6. The other scenario is when some of the query words are not stored in the index table. In this case, we need to perform feature code matching in the underlying feature code files for each of these words. If there are newly found matches, the index table will be updated and the users will be informed. This is usually time-consuming because for each word we have to search every underlying feature code file in order to identify the matches. The experiment shows that the time needed for this scenario is relatively longer. An example of this scenario is shown in Figure 6-7.

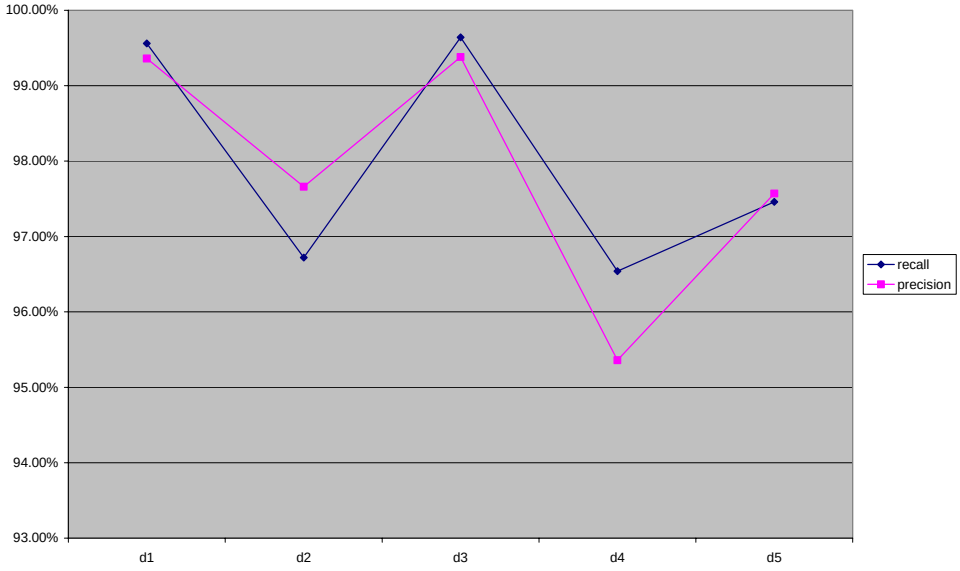


Figure 6-5 Recall and precision chart of the word image coding based system for different categories of documents
d_1=clean documents, d_2=noisy documents, d_3=query on normal fonts, d_4= query on bold fonts, d_5=query on italic fonts

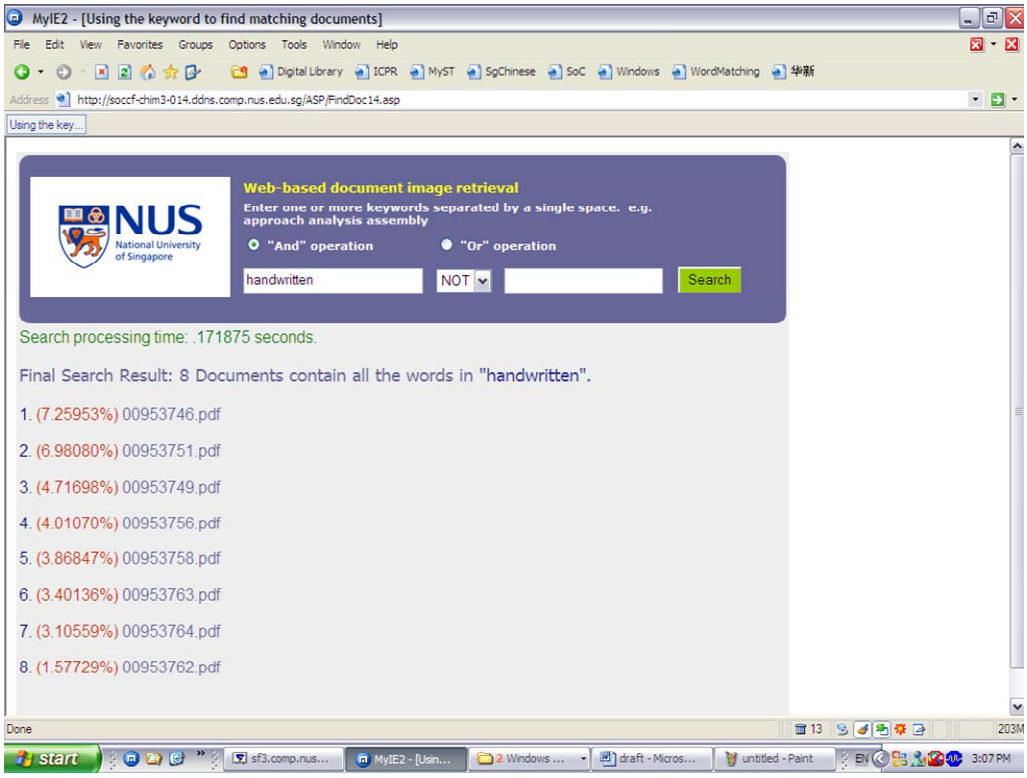


Figure 6-6 Search result for pre-queried word

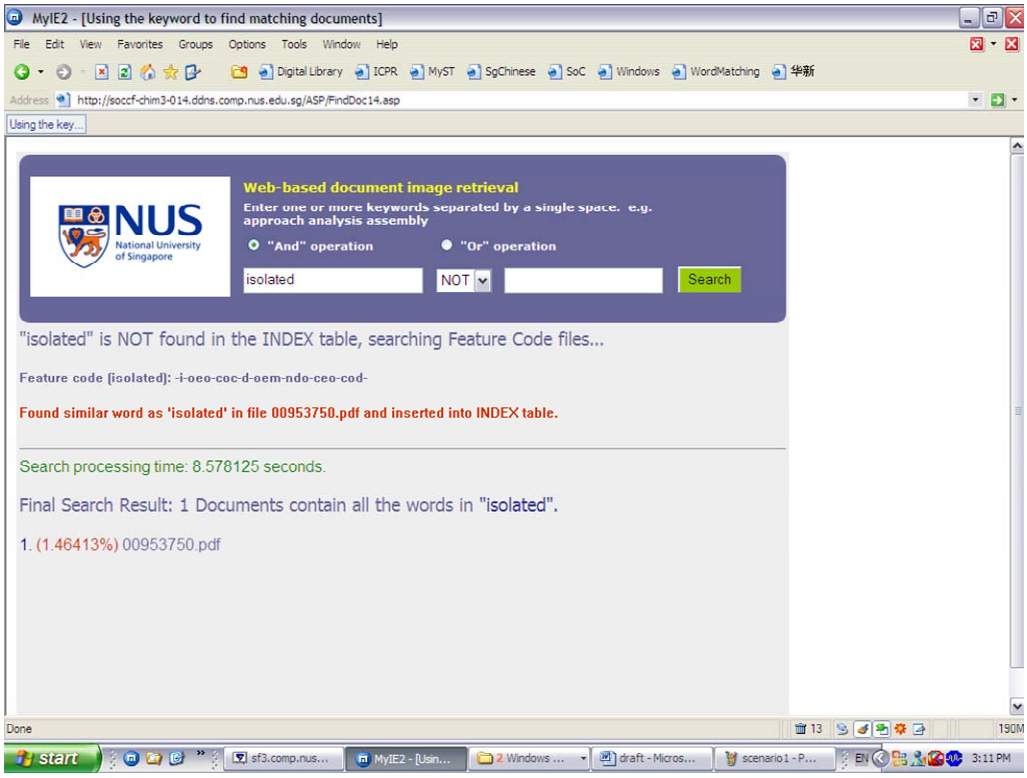


Figure 6-7 Search result for first-time queried word

Chapter 7

Search Engine for Imaged Documents

As we noticed, the normal word search tool provided by Adobe Acrobat Reader does not work for imaged documents packed in PDF format. Currently, there are two popular ways for word searching in imaged documents, namely OCR and Page Capture. Today's OCR, as represented by Caere OmniPage and Xerox TextBridge, does a far better job of reproducing all the elements of a page including both text and graphics, than previous generations of OCR. The accuracy of both packages far surpasses earlier offerings. However, the output of OCR is primarily designed to be edited and modified because of the heritage of the many data entry applications [M96]. On the other hand, Page Capture does a good job in retaining the appearance of the document in terms of both font and layout information. A preferable feature of Page Capture is that it paints a snippet of the image of questionable text into the output PDF document. This makes the unrecognized text readable by the human reader who is using the PDF file; while OCR simply represents the unrecognized text by tildes or noise in its output. However, it is always time consuming to fully convert the imaged documents into its ASCII format before performing the search. In view of this, our search engine is designed to perform direct search on the imaged documents without any loss of information during conversion [LZT+04]. Experiments show that it is approximately 2.6 times faster than the Page Capture provided by Adobe Acrobat.

7.1 Implementation

The search engine works just like the normal word search tool provided by Adobe Acrobat Reader. When a PDF file is opened in Acrobat Reader, a plug-in search tool is shown in the toolbar that allows the user to input a query word and locates the matching words in the document. An example snapshot of the search tool is shown in Figure 7-1. The plug-in is developed using Acrobat SDK with the underlying preprocessing and matching steps implemented in C++. In particular, our search tool works on PDF files opened by Acrobat Reader either from a local PC or from the web through a link. This is therefore a typical supplementary to our web-based document image retrieval system, which allows the user to open the retrieved documents to actually locate the matching words in the documents using our embedded search tool.

When an imaged document is presented to the search engine, it goes through the preprocessing steps as illustrated in Chapter 2. All the connected components in the image are detected and word objects are bounded using a merger operation. Extracted word bitmaps are then represented using a feature code string based on the feature coding mechanism described in Chapter 3. When a user inputs a query word through the search prompt, the search engine generates its corresponding primitive code string by concatenating the characters' PSTs according to the character sequence of the word. This feature code string is then matched with the code strings stored in the feature code files generated for each imaged document. The matching is based on the string matching algorithm presented in Chapter 5. Figure 7-2 shows an example of the search result of the word "Character" located in the PDF document image.

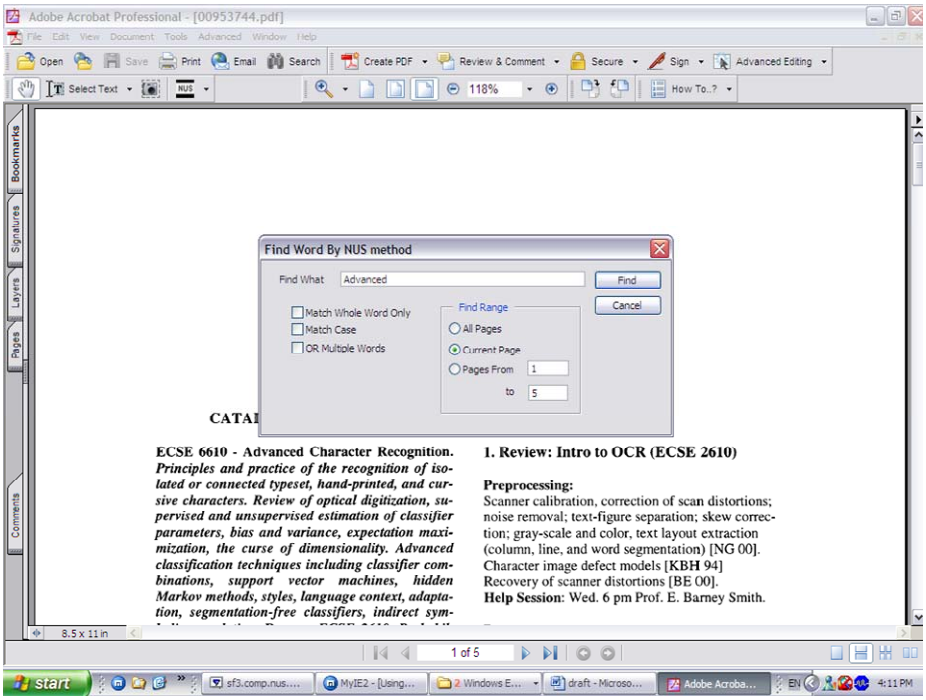


Figure 7-1 Snapshot of the search engine embedded in Acrobat Reader 6.0

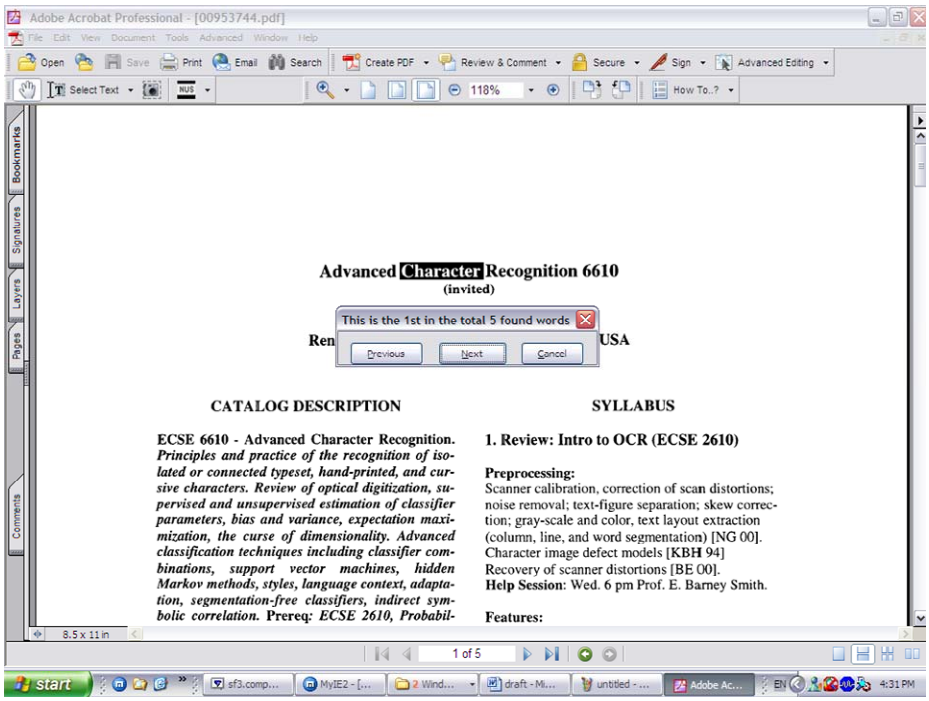


Figure 7-2 Search result for a query word located in an opened PDF document image

7.2 Performance Evaluation

To evaluate the performance of our search engine, two resources of imaged documents packed

in PDF files are used. One is provided by the digital library of NUS. They include 113 PDF files with 3,250 pages of imaged documents scanned from students' theses and 15 PDF files with 328 pages of imaged documents scanned from old books. Other documents are journal and conference papers downloaded from the online databases such as IEEExplore and ScienceDirect. We obtained totally 39 such documents with 294 pages. Therefore, in total, 167 PDF files with 3,872 pages of imaged documents are involved in the test. We selected 150 words as queries and searched for their corresponding words and variations in the documents. The search engine achieves a precision ranging from 88.97% to 99.03% and a recall ranging from 86.21% to 99.15%, depending on the threshold θ used in the feature code matching process. Figure 7-3 shows the average precision and recall as well as the F_1 rating. On average, the best F_1 rating that the system can achieve is 0.9699, where the precision and recall are 98.15% and 95.85% with the threshold set at 0.85. In addition, the precision and recall achieved for words in different length and noise level are analyzed and shown in Figure 7-4. As we can see, the precision and recall for shorter length words are lower than those for longer words due to the edit distance measure used for matching the string features. That the precision for shorter length words is higher than its recall is because we explicitly used full word matching for shorter length words. This avoids some meaningless partial matching like "as" in "gas". On the other hand, for longer query word, the recall is higher than precision. This is because the partial matching sometimes generates matches that are undesired such as matching "format" with "information". The retrieval performance degrades when the noise level of the documents increases, but it still achieves fairly good recall and precision for

longer query words with an appropriate threshold chosen.

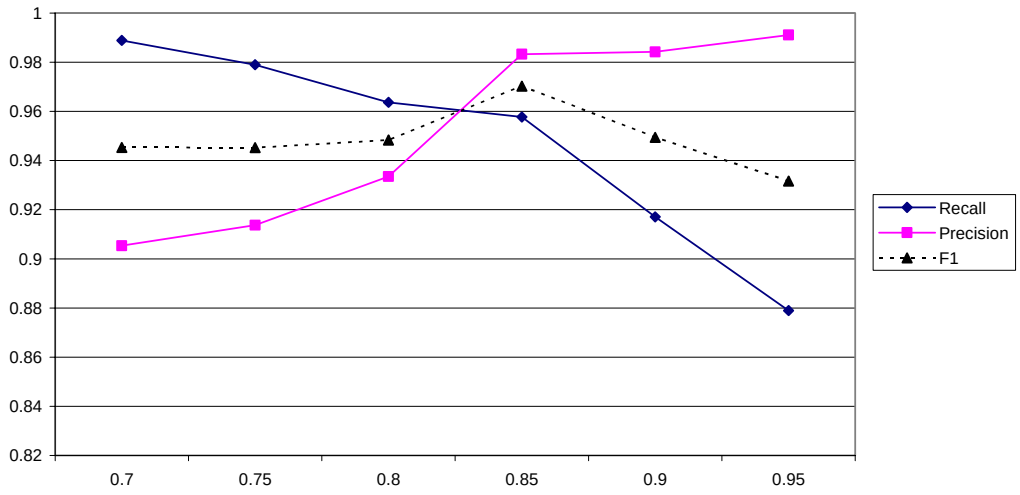


Figure 7-3 Performance vs. different thresholds

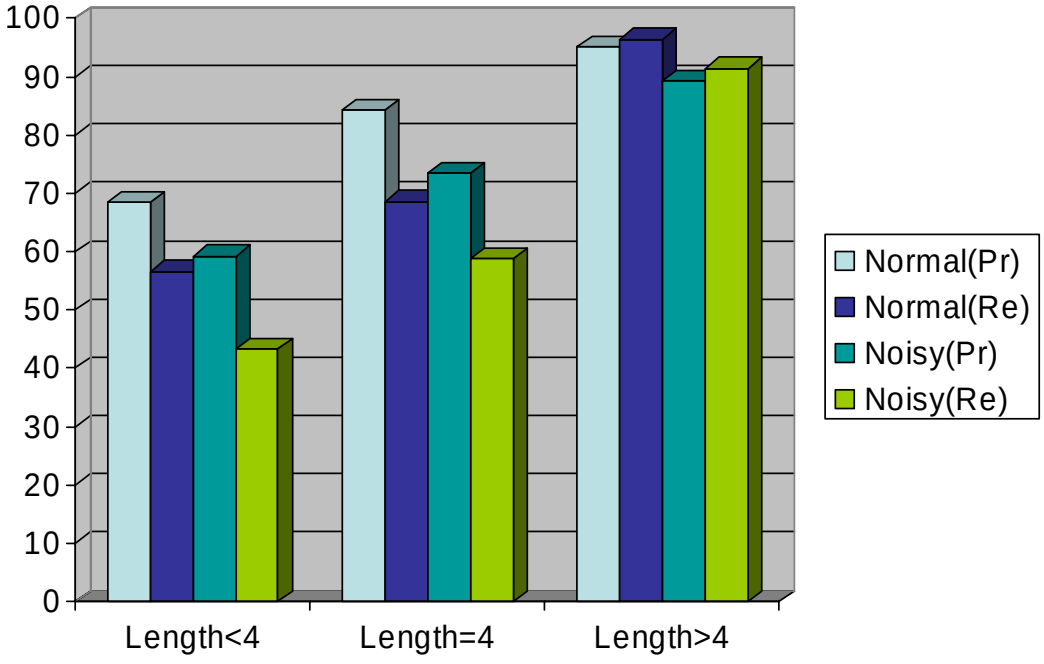


Figure 7-4 Recall and Precision wrt word length distribution and noise level

7.3 Comparison with the Page Capture

The experiments on computer generated page images show that the precision and recall

achieved by the Page Capture of Adobe Acrobat, which basically uses an OCR engine at the back end, are 99.93% and 94.17% respectively. It is noticed that the precision of Page Capture is very high while the recall is a little lower comparing to our search engine. The reason is that the Page Capture tool provided by Adobe Acrobat is lexicon dependent. A lexicon is built into its recognition engine that helps in achieving a high precision. However, it does not perform well in terms of recognition of those uncommon words such as technical terms and people's names. Our search engine here does not rely on any language or lexicon information. This adds in additional flexibility and scalability. Experiments on some noisy documents as illustrated in Figure 7-4 show that our search engine achieves a precision and recall of 89.22% and 91.46%, which is higher than that of Page Capture, 88.12% and 80.34% respectively. This shows that our search engine has a better performance than OCR based approach for degraded documents because of the special treatment for inter-connected characters. In addition, experiments show that our search engine surpasses the Page Capture tool of Adobe Acrobat at about 2.6 times in terms of efficiency because no lexicon or language model is needed. On the other hand, OCR is generally meant for "recognition" problems whereas our word image coding technique is mainly targeted to "retrieval". Hence, a direct comparison between the two may be like comparing oranges and apples.

7.4 Comparison with Hausdorff Distance Based Search Engine

To show the performance advantages of our word image coding based search engine, we also implemented another version based on Hausdorff distance matching of the word images. In

general, word matching may be either at the feature level or at the pixel level. As a low-level matching, the pixel-level matching such as Hausdorff distance is simple but sensitive to changes of image characteristics such as fonts and noise. The main difference between this second system and our first system is that no features are extracted from the word images in the Hausdorff distance based system, instead, direct matching of two word images are used with the Hausdorff distance used as the similarity measure. A typical workflow of this second system can be illustrated as follows:

- The system takes in each query word, maps each character to a standard template image and combines all the character images to obtain a template word image;
- The preprocessing steps including connected components analysis, word bounding box identification, skew detection and rectification, italic font detection and rectification are carried out to extract the word image objects;
- Space elimination and scale normalization are further carried out on the extracted word image objects for best matching;
- Hausdorff distance between the template word image and each word image object is calculated to measure their similarity level as the matching criterion;
- If the distance is greater than a predefined threshold, the word images are identified as a match.

7.4.1 Space Elimination and Scale Normalization

In a word image, it is common that two or more adjacent characters are connected to each other. This is possibly caused by low scanning resolution or poor printing quality. It is so far still a challenging problem to separate them effectively. On the other hand, the templates of the word images used for the input query words are synthesized directly from the standard

bitmap images of each character, in which each character occupies a uniform size of image pixels, e.g. 32×32 per character. This results in a non-uniform spacing between adjacent characters. To remedy this problem, we condense the characters in the word image by eliminating all the spaces between the adjacent characters in both the template image and the word image object extracted from the document image. Finally, the processed word image objects will be normalized to the size of the template image for matching.

7.4.2 Word Matching Based on Hausdorff Distance

Hausdorff distance has been widely used in two-dimensional image matching, especially in the area of object matching. Named after Felix Hausdorff, Hausdorff distance is the maximum distance of a set to the nearest point in the other set. More formally, Hausdorff distance from set ***A*** to set ***B*** is a maximum function, defined as

$$h(A,B) = \max_{a \in A} \left\{ \min_{b \in B} \{ d(a,b) \} \right\}$$

where *a* and *b* are points of sets ***A*** and ***B*** respectively, and ***d(a,b)*** is any metric (e.g. Euclidean distance) between these points. It is noted that Hausdorff distance is asymmetric, which means that most of the time ***h(A, B)*** is not equal to ***h(B, A)***. This asymmetry is a property of maximum functions, while minimum functions are symmetric. Thus, a more general definition of Hausdorff distance would be:

$$H(A,B) = \max\{h(A,B), h(B,A)\}$$

which defines the Hausdorff distance between two sets ***A*** and ***B***. The two distances ***h(A, B)*** and ***h(B, A)*** are sometimes referred to as *forward* and *backward* Hausdorff distance of A to

B. In terms of word matching, ***H(A, B)*** measures the degree of mismatch between two point sets ***A*** and ***B***.

In particular, Y. Lu et al. observed that a word image can be divided into different regions, namely the ascender, the descender, and the mid zone, as shown in Figure 7-4. A weighted Hausdorff distance (WHD) is thus proposed for applications of Hausdorff distance in word image matching. By defining different weight for the contribution of different regions of the word image, the directed distance of WHD is computed as:

$$h_{WHD}(A,B) = \frac{1}{Na}\sum_{a\in A} w(a)\cdot d(a,B)$$

where $N_a = \sum_{a\in A} w(a)$, the weight *w(a)*, *w(m)* and *w(d)* for three regions, namely ascender, mid zone and descender are defined as:

$$w(a) = w(d) = 2*w(m)$$

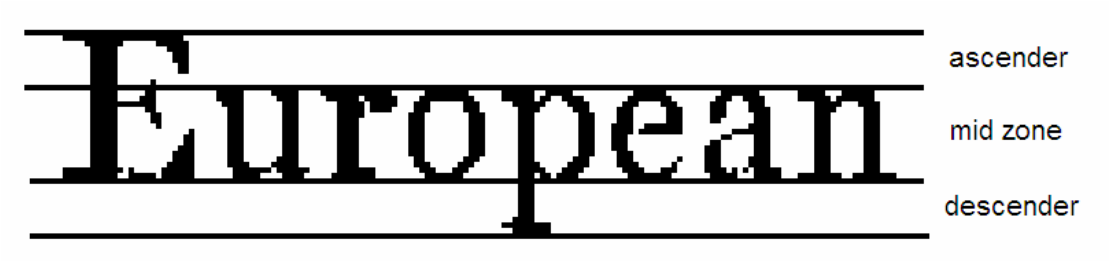


Figure 7-5 Ascender, descender and mid zone of a word image

To compare the two versions of the search engine, same experiment setup is used with a wide range of documents and queries on different set of fonts and styles. Figure 7-5 shows the recall and precision chart for the image coding based version and the Hausdorff distance matching based version. We can see that matching based on Hausdorff distance also produces

a pretty high recall and precision as our word image coding based matching when working on clean Times New Roman documents. However, its performance deteriorates severely when working on Arial documents and bold styles. This is because a standard set of template image in Times New Roman font is used to generate the image for the input query word. Therefore, Hausdorff distance matching is sensitive to font and style variations. In addition, pixel level matching is more time consuming comparing to simple text matching. This also accounts for a much better efficiency in the image coding based version. On the other hand, the Hausdorff distance based approach applies to not only English documents but also documents in other languages such as Chinese documents. This is clearly an advantage of the Hausdorff distance based version.

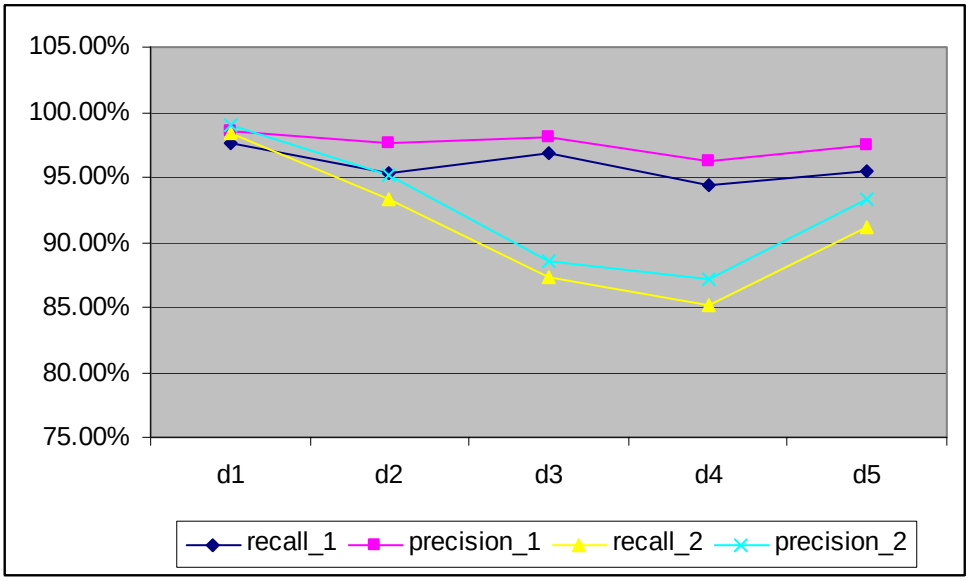


Figure 7-6 Recall and precision chart of Hausdorff distance matching based system for different categories of documents
d_1=clean documents in Times New Roman font, d_2=noisy documents, d_3=documents in Arial font, d_4= query on bold style, d_5=query on italic style

Chapter 8

Conclusions

In this thesis, we presented a novel word image coding technique that represents each word image object extracted from imaged documents using a feature code string. This coding mechanism avoids the character segmentation step commonly used in current OCR technology and achieves a better performance in dealing with degraded document images. On top of this word image coding technique, two experimental applications are implemented to perform information retrieval in imaged documents and have potential employment in digital libraries. In particular, the first application is a web-based document image retrieval system with the word image coding technique employed during the off-line preprocessing step. This system is used to retrieve relevant document images based on a set of input query words specified by the user through a web interface. The second application is a search engine for imaged documents in PDF format. It is a typical plug-in search tool embedded in Adobe Acrobat Reader that explicitly locates the query word in the opened PDF file either from a local machine or through a web link. Both applications are implemented with the ability of recognizing word objects in various fonts and styles, such as bold and italic. In addition, skew and noisy images are taken care of during the preprocessing step with a robust skew detection and rectification algorithm proposed earlier on. In the following two sections, we will first review the major contributions of this thesis and then discuss the additional work that should be done in the future.

8.1 Contributions

This thesis presented a novel word image coding technique that can be used in designing and developing applications to retrieve information from imaged documents. Our word image coding technique can be viewed as an alternative to the current OCR technology with a main difference that our technique extracts features on a word level instead of explicitly recognizing each individual character as in OCR. Two experimental applications are implemented which showed an encouraging performance in terms of recall, precision and retrieval efficiency.

The main contributions of this thesis are summarized as follows:

- Presented a word image coding technique that extracts features from word objects and represents them using a typical coding string. Refinement is done to incorporate italic font identification and retrieval.
- Employed a connected component detection algorithm and a nearest-neighbor based skew detection algorithm during the preprocessing step to rectify the skew images and extract the word objects with a normal upright style.
- Proposed and implemented an italic font recognition algorithm based on wavelet transformation to detect italic words scattered in the document images and rectify them before generating the feature code strings. Comparisons are done with traditional stroke pattern analysis approaches and show a better performance in terms of accuracy and efficiency.
- Designed and developed a web-based document image retrieval system that takes in a set of users' query words through a web interface and returns a list of relevant documents ranked according to the occurrence frequency of the query words in the documents. Preprocessing steps are first carried out off-line to generate the

corresponding feature code files for the document images. String matching is then used to match the feature code string of the users' query word with the feature code strings stored in the feature code files. If matches are found, the corresponding document images will be returned to the user. The user can link to the actual documents opened using Adobe Acrobat Reader and explicitly locate the matching words spotted.

- Designed and developed a search engine for imaged documents packed in PDF files. The search engine is essentially a plug-in search tool embedded in Adobe Acrobat Reader that performs word search in the opened PDF document either from a local machine or through a web link. When a document is presented to Acrobat Reader, it goes through a series of preprocessing steps which extract the word objects and represent them using a string of feature codes for a later matching. When a user inputs a query word, its feature code string representation will be generated and matched with the code strings of the word objects in the document image. As a result, the most relevant words will be marked in the documents based on a similarity threshold.

- Developed another version of the search engine based on Hausdorff distance matching of word images. Comparisons are done with the word image coding based search engine and show that our word image coding based system achieves a better recall and precision with less sensitivity to font style variations. In addition, a better efficiency is achieved in terms of the online search process since the preprocessing steps are performed off-line. On the other hand, pixel matching with gap processing within each word object appears to be time consuming for the Hausdorff distance matching based system.

8.2 Future Works

- As we mentioned in the thesis that the two applications of our word image coding technique are basically two experimental models, therefore, further scaled and comprehensive testing are needed to make robust applications for the use of our digital library.

- The web-based document image retrieval system with the underlying index table stored in an Oracle database needs to be well trained in order to show its retrieval efficiency and intelligence.
- Currently, finding imaged documents of relevant contents still has to rely on painful downloading of individual scanned documents for local viewing. Our search engine opens up the possibility of screening imaged documents for selective downloading.
- The search engine for PDF document images currently can only work with single query word on the current page image. It can be extended with the ability of searching multiple words on a range of pages.
- The word image coding technique can be further improved to be case insensitive by constructing a map between the PSTs of lowercase letters and its corresponding uppercase letters.
- The word image coding technique currently only works on English documents. With a different set of feature associative mapping, the technique can be extended to deal with documents in other languages as well. This will eventually extend our applications to handle multi-lingual documents.
- Our wavelet transformation based italic font recognition algorithm currently is only tested on an extensive English dictionary. It can be extended to deal with documents in other languages as well such as Chinese documents and other Asian or European languages.

Bibliography

[ACC01] E. Appiani, F. Cesarini, A. M. Colla, Automatic Document Classification and Indexing in High-volumn Applications, *Int'l Journal on Document Analysis and Recognition,* vol. 4, pp. 69-83, 2001.

[AG99] A. Apostolico, R. Giancarlo, Sequence Alignment in Molecular Biology, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 47, pp. 85-115, 1999.

[BN94] H. S. Baird, G. Nagy, A Self-Correction 100-Font Classifier, *Proc. of SPIE conf. on Document Recognition*, 106-115, 1994.

[C97] R. Cooperman, Producing Good Font Attribute Determination Using Error-Prone Information, *Int'l Society for Optical Eng. J.*, vol. 3027, pp. 50-57, 1997.

[CB98] F. R. Chen, D. S. Bloomberg, Summarization of Imaged Documents without OCR, *Computer Vision and Image Understanding,* vol. 70, no. 3, pp. 307-319, 1998.

[CG98] B. B. Chaudhuri and U. Garain, Automatic Detection of Italic, Bold and All-Capital Words in Document Images, *Proc. 14th Int'l Conf. on Pattern Recognition (ICPR)*, vol. 1, pp. 610-612, 1998.

[CG01] B. B. Chaudhuri, U. Garain, Extraction of Type Style-based Meta-information from Imaged Documents, *Int'l Journal on Document Analysis and Recognition,* no. 3, pp. 138-149, 2001.

[CWB93] F. R. Chen, L. D. W, D. S. Bloomberg, Detecting and Locating Partially Specified Keywords in Scanned Images Using Hidden Markov Models, *Proc. of the 2nd Int'l Conf. on Document Analysis and Recognition,* pp. 133-138, 1993.

[D98] D. Doermann, The Indexing and Retrieval of Document Images: A Survey, *Computer Vision and Image Understanding,* vol. 70, no. 3, pp. 287-298, 1998.

[G97] D. Gusfield, Algorithms on Strings, Trees, and Sequences, *Cambridge University Press*, 1997.

[GG98] E. A. Galloway, V. M. Gabrielle, The Heinz Electronic Library Interactive On-line System: An Update, *The Public-Access Computer Systems Review,* vol. 9, no. 1, 1998.

[HG98] H. J. A. M. Heijmans, J. Goutsias, Some Thoughts on Morphological Pyramids and Wavelets, *Signal Processing IX: Theories and Applications,* pp. 133-136, 1998.

[HG00] H.J.A.M. Heijmans, J. Goutsias, Multiresolution Signal Decomposition Schemes, Part 2: Morphological Wavelets, *IEEE Transactions on Image Processing,* Vol. 9, No. 11, pp. 1897-1913, 2000.

[HJLZ99] Y. He, Z. Jiang, B. Liu, H. Zhao, Content-based Indexing and Retrieval Method of Chinese Document Images, *Proc. of the 5th Int'l Conf. on Document Analysis and Recognition,* pp. 685-688, 1999.

[J97] R. Jain, Visual Information Management, *Communications of the ACM* 40(12): 31-32.

[JBN96] M. Y. Jaisimha, A. Bruce, T. Nguyen, DocBrowse: A System for Information Retrieval from Document Image Data, *Proceeding of the SPIE,* vol. 2670, pp. 350-361, 1996.

[KH96] S. Khoubyari, J. J. Hull, Font and Function Word Identification in Document Recognition, *Computer Vision and Image Understanding,* vol. 63, no. 1, pp. 66-74, 1996.

[KHOY99] T. Kameshiro, T. Hirano, Y. Okada, F. Yoda, A Document Image Retrieval Method Tolerating Recognition and Segmentation Errors of OCR Using Shape-feature and Multiple Candidates, *Proc. of 5th Int'l Conf. on Document Analysis and Recognition,* pp.681-684, 1999.

[KTK02] K. Katsuyama, H. Takebe, K. Kurokawa, Highly Accurate Retrieval of Japanese Document Images Through a Combination of Morphological Analysis and OCR, *Proc. SPIE, Document Recognition and Retrieval,* vol. 4670, pp. 57-67, 2002.

[L01] D. Lopresti, A Comparison of Text-based Methods for Detecting Duplication in Scanned Document Databases", *Information Retrieval*, vol. 4, no. 2, pp. 153-173, 2001.

[LT03] Y. Lu, C. L. Tan, Improved Nearest Neighbor Based Approach to Accurate Document Skew Estimation, International Conference on Document Analysis and Recognition, ICDAR 2003, 3-6 August, Edinburgh, UK.

[LZ96] D. Lopresti, J. Zhou, Retrieval Strategies for Noisy Text, *Proc. of the Fifth Annual Symposium on Document Analysis and Information Retrieval*, LA, NV, pp. 255-269, 1996.

[LZT04] Y. Lu, L. Zhang, C. L. Tan, Retrieving Imaged Documents in Digital Libraries Based on Word Image Coding, *International Workshop on Document Image Analysis for Libraries*, CA, USA, 2004.

[LZT+04] Y. Lu, L. Zhang, C. L. Tan, A Search Engine for Imaged Documents in PDF Files, *27th Annual International ACM SIGIR Conference*, Sheffield, UK, 2004.

[M89] S. Mallat, A Theory for Multiresolution Signal Decomposition: the Wavelet Representation, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674-693, Jul. 1989.

[M96] T. Mckinley, Acrobat Capture vs. OCR: Apples and Oranges, *Intelligent Imaging,* 1996.

[M98] S. Mallat, A Wavelet Tour of Signal Processing, San Diego, CA: Academic,

1998.

[M97] M. T. Maybury, Intelligent Multimedia Information Retrieval, *AAAI/The MIT Press*.

[S97] A. L. Spitz, Duplicate Document Detection, *Proc. of SPIE, Document Recognition IV (L. M Vincent and J. J. Hull edit),* vol. 3027, San Jose, CA, USA, pp. 88-94, 1997.

[S99] A. L. Spitz, Shape-based Word Recognition, *Int'l Journal on Document Analysis and Recognition,* vol. 1, no. 4, pp. 178-190, 1999.

[S02] A. L. Spitz, Progress in Document Reconstruction, *Proc. of 16th Int'l Conf. on Pattern Recognition,* vol. 1, pp. 464-467, 2002.

[SP97] H. Shi and T. Pavlidis, Font Recognition and Contextual Processing for More Accurate Text Recognition, *Proc. Fourth Int'l Conf. Document Analysis and Recognition, (ICDAR '97)*, pp. 39-44, Aug. 1997.

[SS97] C. Sun and D. Si, Skew and Slant Correction for Document Images Using Gradient Direction, *Proc. Int'l Conf. on Document Analysis and Recognition (ICDAR)*, vol. 1, pp. 142-146, 1997.

[SS+97] A. F. Smeaton, A. L. Spitz, Using Character Shape Coding for Information Retrieval, *Proc. of the Fourth Int'l Conf. on Document Analysis and Recognition,* pp. 974-978, 1997.

[TBCE94] K. Tagvam, J. Borsack, A. Condir, S. Erva, The Effects of Noisy Data on Text Retrieval, *Journal of the American Society for Information Science,* vol. 45, no. 1, pp. 50-58, 1994.

[TV93] J. M. Trenkle, R. C. Vogt, Word Recognition for Information Retrieval in the Image Domain, *Symposium on Document Analysis and Information Retrieval,* pp. 105-122.

[WS99] M. Worring, A. W. Smeulders, Content Based Internet Access to Paper Documents, Int'l Journal on Document Analysis and Recognition, vol. 1, pp. 209-220, 1999.

[ZI98] A. Zramdini and R. Ingold, Optical Font Recognition Using Typographical Features, *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 20, no. 8, pp. 877-882, Aug. 1998.

[ZLT03] L. Zhang, Y. Lu, C. L. Tan, A Web-based System for Retrieving Document Images from Digital Library, Workshop on Document Image Analysis and Retrieval, in conjunction with CVPR2003, 16-22 June 2003, Madison, Wisconsin, USA.

[ZLT04] L. Zhang, Y. Lu, C. L. Tan, Italic Font Recognition Using Stroke Pattern Analysis on Wavelet Decomposed Word Images, *International Conference of Pattern Recognition*, Cambridge, UK, 2004.

[ZTW01] Y. Zhu, T. N. Tan, Y. H. Wang, Font Recognition Based on Global Texture

Analysis, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 23, no. 10, 2001.

Appendix A – How to Use the Web-based Retrieval System

- Access the web interface through the following URL (as shown in Figure A-1):

  http://soccf-chim3-014.ddns.comp.nus.edu.sg/ASP/FindDoc14.asp

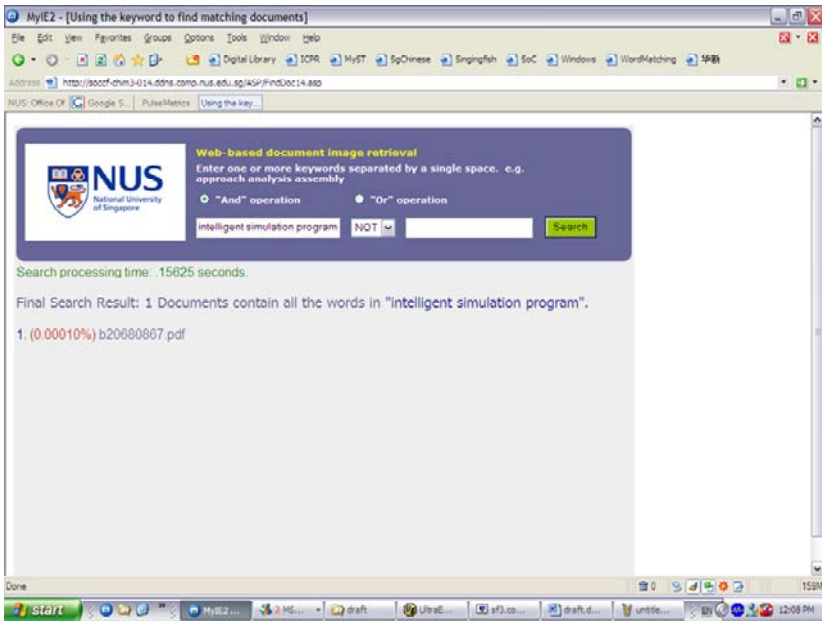


Figure A-1 Web-based document image retrieval system

- Input a set of query words in the first AND/OR input box, separated by spaces, e.g. “intelligent simulation algorithm”.
- Indicate either to perform AND or OR operation by clicking the radio button.
- Input a set of query words in the second NOT input box, also separated by spaces, e.g. “computational approach”.
- Click “Search”, then the retrieved documents will be returned and ranked according to the occurrence frequency of the query word in each document.
- Link to the actual document images through the hyperlink over the retrieved documents’ name for online reading and verification.

Appendix B – How to Use the Search Engine

- Create a folder called AcrobatSDK under your Adobe Acrobat installation directory, e.g. C:\Program Files\Adobe\Acrobat 6.0\Acrobat\Plug_ins\AcrobatSDK
- Put the NUSFind.api under the AcrobatSDK directory.
- Open the document image using Acrobat Reader from the local machine.
- The new plug-in will appear on the toolbar as shown in Figure B-1.

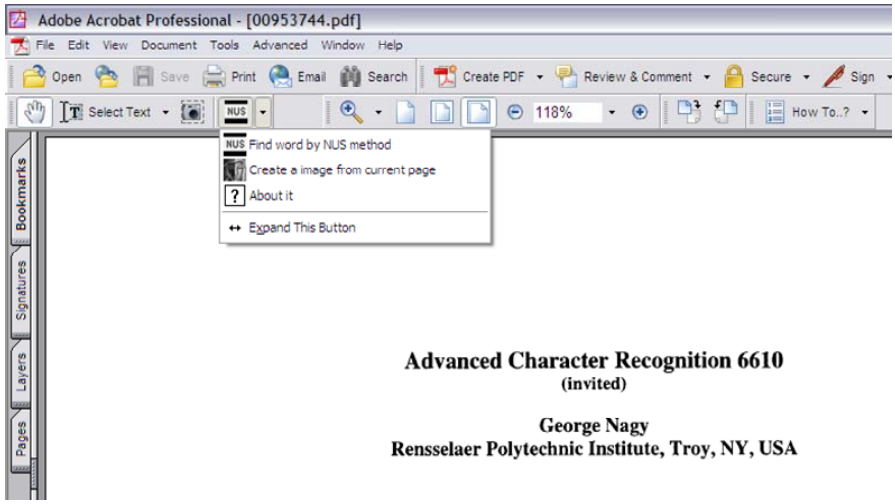


Figure B-1 Plug-in drop down menu

- Select "Find Word By NUS method" from the drop down menu and you will be prompted with the search dialog box as shown in Figure B-2.

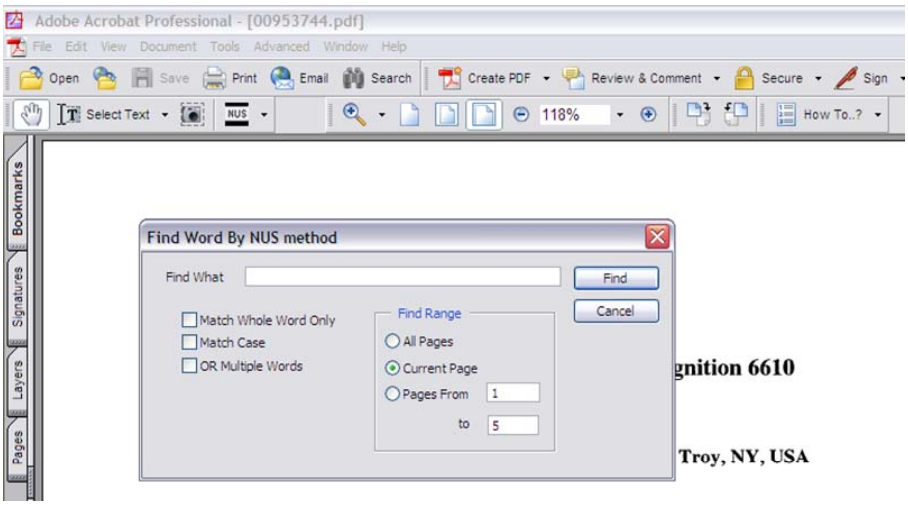


Figure B-2 Search prompt dialog box

- Input the query word that you would like to search in the dialog box and select "current page" as the Find Range.
- If "Match Whole Word Only" is selected, only the exactly matched words will be identified.
- The matching words will be identified and marked in black as shown in Figure B-3.

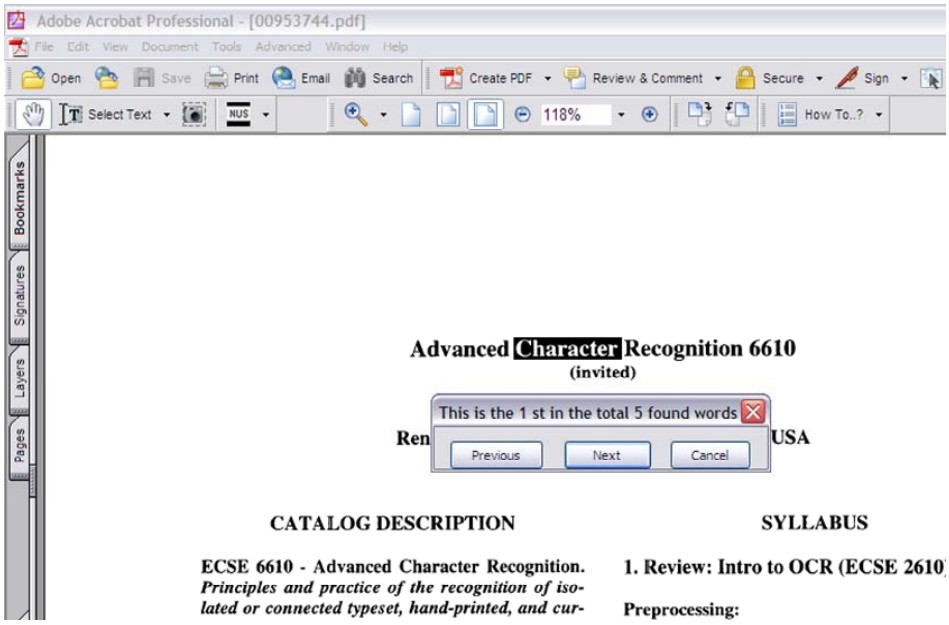


Figure B-3 Spotted words in the documents