

A Word Image Coding Technique and its Applications in Information Retrieval from Imaged Documents

Li Zhang, Chew Lim Tan

*School of Computing, National University of Singapore
3 Science Drive 2, Singapore 117543
Email: {zhangli,tancl}@comp.nus.edu.sg*

Abstract

With the need of current fast evolving digital libraries, an increasing amount of documents are being digitized into an electronic format for easy archival and dissemination purposes. Thus Document Image Retrieval (DIR), as part of information retrieval (IR) paradigm, is receiving attentions among the IR communities in recent years. This paper presents two DIR applications based on a word image coding technique to extract features from each word image object and represent them using feature code strings for comparison. The first application is a web-based retrieval system that retrieves document images online from digital libraries based on a set of input query words. The second one is a plug-in search tool embedded in Acrobat Reader that performs word spotting within the opened document images and marks the matching words explicitly in the document. Both applications achieve good precision and recall according to our experiments on document images such as students' theses provided by our university digital library.

1. INTRODUCTION

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 (Doermann, 1998). Modern technology has made it possible to produce, process 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. In the past years, various ways have been studied to query on imaged documents using physical layout and logical (semantic) structure information as well as extracted contents such as image features using different levels of abstraction. However, for imaged documents with text content as dominant information, the traditional IR 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 text format, and then apply text IR strategies over the converted text documents. Based on this idea, several commercial systems have been developed using page segmentation and layout analysis techniques, following OCR. These include Heinz Electronic Library Interactive Online System (HELIOS) developed by Carnegie Mellon University (Galloway, 1998), 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 (DIP) applications (Tagvam, 1994). A DIP system needs to analyze different text areas in a document image, understand the relationships among these text areas, and then convert them to a machine-readable format using OCR. 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 words, 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 as in DIP. Motivated by this observation, some methods with the ability of tolerating OCR errors by using the OCR candidates have been proposed (Kameshiro, 1999) (Katsuyama, 2002).

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 (Spitz 1997), information retrieval (Smeaton, 1997), word recognition (Spitz, 1999) and document reconstruction (Spitz, 2002) 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. His 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 and thus not applicable to the case where characters are interconnected. Chen et al. (Chen, 1998) proposed a segmentation free approach using word shape information. He first identifies upper and lower contours of each word using morphology and then extracts shape information based on 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 et al. (Trenkle, 1993) 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. (He, 1999) proposed an index and retrieval method based on character codes generated from stroke density.

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.

This paper presents a word image coding technique that can be used to perform online search of word objects in document images 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: (1) Features are extracted at the word level, rather than at the character level as it appears in Spitz's character shape codes; (2) The procedure of computing word image codes is more complicated, but shows an advantage of eliminating ambiguity among words.

Based on the word image coding technique, two applications are presented in view of online and off-line execution of the word image coding mechanism. The 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 respectively. Preprocessing is carried out off-line to de-noise 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 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 without the need of additional database to store feature code files. Nevertheless, 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.

The rest of the paper is organized as follows: Section 2 describes the word image coding technique and evaluates its validity as a unique coding representation at the word level. Section 3 details the preprocessing procedures that are performed to extract word image objects from the original imaged document and generate the feature code strings using the word image coding technique. Section 4 elaborates the inexact string matching algorithm used to match the feature code strings of the word images and the input query words. In section 5, we propose a wavelet decomposition based approach to identify italic font. Next, section 6 presents the web-based retrieval system based on a set of input query words and section 7 describes the search tool in Acrobat Reader for word spotting in opened document images. Finally, we draw some conclusions in section 8 and discuss about some future works.

2. WORD IMAGE CODING TECHNIQUE

Generally speaking, our word image coding technique is to represent each word image object using specially designed codes according to its features (Lu, 2004). The features used here 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 2.2.

2.1 LRPS Feature Representation

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

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

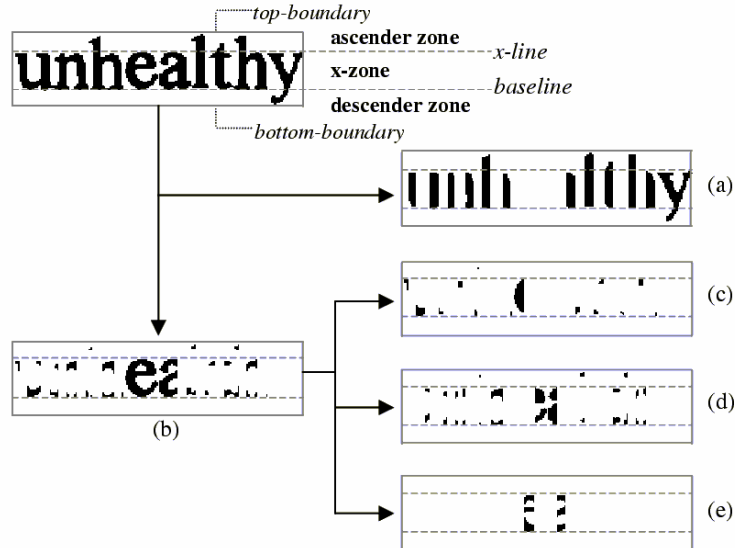


Fig. 1 Primitive string extraction
 (a) straight stroke line features (b) remaining part of word image
 (c) traversal $T_N = 2$ (d) traversal $T_N = 4$ (e) traversal $T_N = 6$

2.2 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. For example, 'x' indicates that the primitive is between the x-line and the baseline and 'A' indicates that the primitive is between the top-boundary and the baseline. The definition of x-line, baseline, top and bottom-boundary can be found in Fig. 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.

2.3 Line-or-Traversal Attribute

The LTA value is generated in two steps. First, the straight stroke line features are extracted from the word image, as shown in Fig. 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 LTA of the corresponding primitive. In other word, the LTA of a primitive is represented by either a straight stroke line feature or a traversal feature otherwise.

2.3.1 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 containing pixel a , along the specified direction θ . $|R(a, \theta)|$ is the run length, which is the total number of black pixels in the run. If $|R(a, \theta)|$ is greater or equal to x-height, the pixels between the topmost and bottommost pixel are extracted as a straight stroke line.

Fig. 1(a) shows the straight stroke lines (SSLs) extracted from the word “unhealthy”. Fig. 1(b) shows the remaining word image. There are three types of SSLs based on their directions: vertical stroke line, left-down diagonal stroke line and right-down diagonal stroke line. Based on this, three basic primitives are generated. The ADAs of these primitives can be obtained from their top-end and bottom-end positions of the SSLs. For example, the left-down diagonal stroke line in the character ‘z’ is located between the x-line and the baseline. Therefore, the corresponding primitive has a value of ‘x’ as its ADA value. On the other hand, the LTAs of these three types of primitives are evaluated as ‘l’, ‘w’ and ‘v’ respectively. Furthermore, for the primitives whose ADA are ‘x’ or ‘D’, we need further check whether there is a dot on top of the vertical stroke line. If there is, the LTA of the primitive is re-assigned with the value ‘i’ or ‘j’ respectively.

2.3.2 Traversal Feature

After the SSL features are extracted, traversal features in the remaining word image are analyzed by scanning the remaining 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. Different feature codes are then assigned based on the value of T_N ,

- $T_N = 0$: there is no image pixel in the column. We assign it with the feature code ‘&’ that corresponds to the inter-character space. The overlap of adjacent characters caused by kerning is easily detected by analyzing the relative positions of the adjacent connected components based on which space primitive can be inserted.
- $T_N = 2$: two parameters are used to assign its feature code. One is the ratio of its black pixel number to the x-height, denoted by 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.

Similar definitions are specified for $T_N = 4$, $T_N = 6$ and $T_N = 8$. As a result, a series of primitives are generated and expressed as a sequence of (σ, ω) pairs representing either straight stroke line features or traversal features as shown in Fig. 1(a) and Fig.1 (c)(d)(e) respectively. One thing to note is that some columns may not be assigned with any feature code because they cannot meet any of the requirements for the aforementioned eligible features. They are usually insignificant features or caused by noise and thus can be safely eliminated.

2.4 Post-processing

2.4.1 Merging Consecutive Identical Primitives

As we mentioned earlier, each primitive is described using two attributes σ and ω , where σ is assigned with values based on the line or traversal features detected and ω is assigned with five values describing the ascender or descender property of the primitive. Based on our observation, the valid combinations of σ and ω are limited. For example, $\sigma = 'n'$ can only correspond to $\omega = 'x'$. Therefore, for conciseness, we can replace each (σ, ω) pair in the primitive sequence using one single character as listed in Table 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 due to bold faces. These redundant features can be combined and represented using one single character. At this stage, the resultant primitive string of the word “unhealthy” in Fig. 1 is obtained as: $\langle n\mu o\mu o\mu o\mu o\mu \& O\mu o\mu n\& \mu c\mu e\mu o\& \mu o\mu e\mu \mu O\mu d\& \mu n\mu d\mu o\mu O\mu d\mu o\mu n\mu \mu \& \mu y \rangle$.

2.4.2 Refinement for Font Independence

To enable our retrieval system to handle document images with different fonts and styles, the primitive string we obtained earlier should be independent of typefaces. Among various fonts, a significant factor that affects the LRPS extraction is the serif font. This is particularly true for the extraction of traversal features. 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 $\langle \mu \& \rangle$ is normally generated by a right-side serif in the characters such as ‘a’, ‘h’, ‘m’, etc. Therefore, we can simply remove ‘u’ from the primitive sequence $\langle \mu \& \rangle$. Similarly, many other refinement rules are applied. With this step, the primitive string of the word image in Fig. 1 becomes: $\langle \mu \mu \mu \mu \mu \mu n\& \mu d\mu o\mu n\& \mu c\mu e\mu o\mu \mu d\& \mu n\mu d\mu o\mu n\mu \& \mu y \rangle$. 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 of which many are actually caused by bold faces.

Table 1 Primitive properties vs. Character code representation

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		

2.5 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>. Consequently, the standard primitive string of a word can be generated by synthesizing the PST 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 PSTs. As the example in Fig. 1 shows, due to noise effect, the primitive substring for 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 misrepresentations comparing to the standard version. 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 section 4).

2.6 Verification

We use a dictionary of 25,133 commonly used English words to evaluate the validity of our word image coding scheme. Each word is represented by its corresponding word primitive

token (WPT) generated by concatenating each character's PST 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.

3. PRE-PROCESSING FOR FEATURE CODE GENERATION

With respect to each document image, a corresponding feature code file is generated 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 for the feature matching function. The detailed procedures are elaborated in the following sections.

3.1 Connected Component Analysis and Word Bounding

Given a page of an imaged document, we first apply connected component analysis to detect all the connected components within this page. Here, we assume all the images are binary images (or otherwise converted to binary). Furthermore, additional operations are carried out to remove some noise 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 on top of 'i' and 'j', we will group them with the body part of 'i' and 'j' as one connected component instead of discarding them. 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 are concerned is the text information rather than graphics and tables.

Based on the connected components, we try to find all the word-bounding boxes based on the locations of these connected components. 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 with large overlapping area.

3.3 Skew Estimation and Rectification

As we notice in some images, the text lines are skew. In order to generate an accurate set of feature code strings, we need to first rectify this page image back to its normal shape before

applying the word image coding scheme. Therefore, we first try to find its skew angle using a nearest neighbor chain (NNC) algorithm (Lu, 2003). The idea is based on the observation that the slope of an inclined line can be estimated by the slope of a NNC that consists of several consecutive connected components of similar height/width. Specifically, we take the median of the slopes of all the NNCs of sufficient length as the skew angle of the page image. Given the skew angle, we can then rotate each word-bounding box by the skew angle to obtain a new word-bounding box inside which the word is in its normal shape.

3.4 Italic Font Detection and Rectification

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. In view of our word image coding technique, we propose a wavelet-decomposition based italic font recognition approach to identify individual words in italic style and rectify them accordingly before generating the feature code representation (to be illustrated in section 5).

3.5 Feature Code File Generation

At this stage, each word object is extracted from the document image and rectified to its normal shape. Next, by applying the word image coding technique, each word image is represented using a primitive string as described in section 2. The feature code file is then generated, which contains information of all the feature code strings of the word objects, their locations in the document and the URL of the document image.

4. FEATURE CODE MATCHING

Following a coarse matching step based on the word's x-feature and the number of primitive codes in the feature code string, 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 $A = \langle a_1, a_2, \dots, a_n \rangle$ representing the query word and the string $B = \langle b_1, b_2, \dots, b_m \rangle$ representing the 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 $A = \langle \text{dnm}\&\text{ceo}\&\text{oem}\&\text{d}\&\text{ndo}\&\text{dnm} \rangle$ 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 $\langle \& \rangle$ 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 (Apostolico, 1999). Lopresti et al. applied the inexact string matching strategy to information retrieval (Lopresti, 1996) and duplicate document detection for dealing with imprecise text data generated from OCR (Lopresti, 2001). Drawing inspiration from the alignment problem of two DNA sequences and the research done by Lopresti et al., 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.

The optimal alignment score of two strings A and B can be computed by a dynamic programming with recurrences. $\delta(a_i, -)$ is defined to be the matching value between the i th element of A and the space character, since string B is empty. Similarly, $\delta(-, b_j)$ is defined to be the matching value between the j th element of B and the space character. The base condition and the general recurrence relation are:

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

The zero in the above recurrence implements the operation of *restarting* the recurrence, which ensures that the unmatched prefixes are discarded from the computation. 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. Finally, the maximum scoring is normalized as:

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

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.

Table 2 shows the table computed for the primitive string of the word image “unhealthy” and that of the query word “health”. It is observed that the maximum score obtained 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 that is normally performed in the text retrieval processes. It is also feasible to reinforce the functionality of word stemming by translating the stemming rules into their corresponding LPRS representations during the refinement 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.

Table 2 Scoring table and missing space recovery

		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		
h		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	0	2	1	0	0	0	0	0	0	0	0	0	2	1	0	2	1	2	1	0	0	0	
	n	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	0	
	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
e	&	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	0
	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	0
	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	0
	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
a	&	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	0
	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	0
	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	0
	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
l	&	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	0
	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	0	
	n	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	0	
t	d	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	0	
	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	0	1	1	0	0	3	3	6	6	8	7	6	7	9	10	13	16	19	22	25	28	28	27	26	28	27	0	
	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	0
y	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	0
	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	0

The match may be imprecise in the sense that certain primitives can be missing or miscoded. As we mentioned earlier, some adjacent characters in a word image may be interconnected due to various reasons. This results in less number of inter-character spaces being detected and thus less spacing primitives $\langle \& \rangle$ being translated in the primitive string. Looking again at table 2, we find the best matching sequence by backtracking from the maximum score obtained. Our experiments show that the maximum score 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:

$$S_m = \left(\max_{\forall i,j} V(i,j) + \tau(N_g) \right) / V_A^*(n,n) \quad (3)$$

where N_g is the number of missing spacing primitives included in the backtracking sequence. In our experiments, we have $\tau(N_g) = 2 \times N_g$.

5. ITALIC FONT RECOGNITION

As we noticed in many occasions italic fonts are used to distinguish or emphasize certain words scattered in the normal (upright) text such as scientific terms. 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. In this section, we propose a wavelet transformation based technique that features in detecting italic font at the word level with less sensitivity to noise and typeface variations.

5.1 Wavelet Decomposition of Word Images

To identify italic fonts, a simple but reliable way is to extract stroke patterns from each word image for analysis. However, stroke patterns obtained from original word images are largely dependent on font styles such as size, serifness, bold, etc. Therefore, to reduce the sensitivity of stroke pattern analysis to font variations as well as noise and distortions, wavelet decomposition is performed prior to the stroke pattern analysis step to extract dominant word features in horizontal, vertical and diagonal directions.

In order to obtain good features during the wavelet decomposition step, each word object needs to be normalized to have a uniform “x-height” during pre-processing. The normalization does not need to be very precise since it will not affect the dominant features extracted from the discrete wavelet decomposition. After the word image is normalized, 2-D discrete wavelet decomposition (2-D DWT) is performed to extract dominant stroke patterns. Various wavelet filters such as Daubechies and Symlets (Mallat 1998) can be employed in this 2-D DWT step (Mallat, 1989), which essentially computes an approximation sub-image as well as three detailed sub-images in vertical, horizontal and diagonal directions. In particular, one-level decomposition using symlet of order two (sym2) is found to excel at 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]. The decomposition procedure is illustrated in Fig. 2.

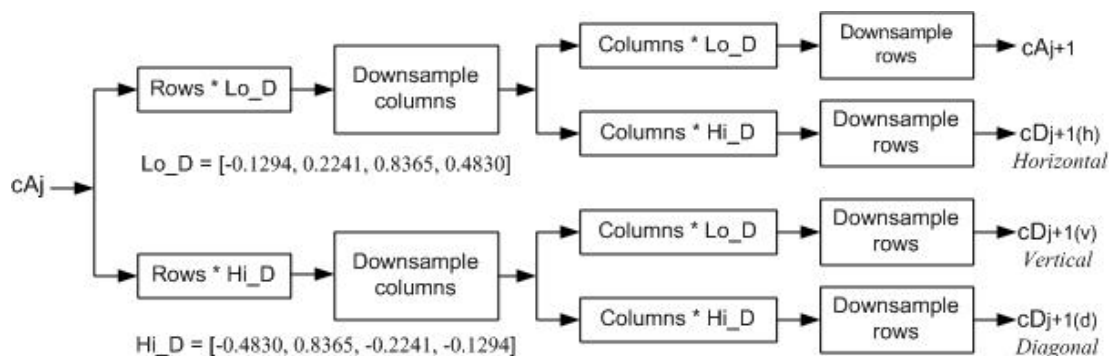


Fig. 2 Two dimensional Discrete Wavelet Decomposition

European

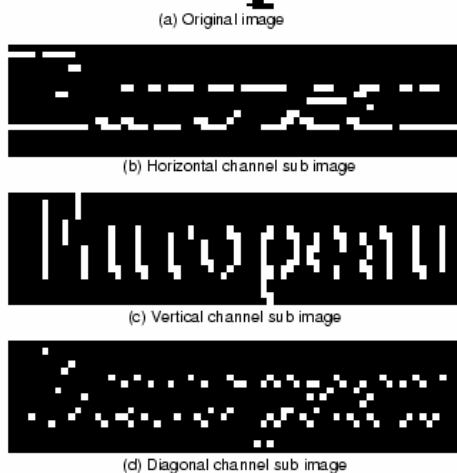


Fig. 3 An example of one-level wavelet decomposed sub images

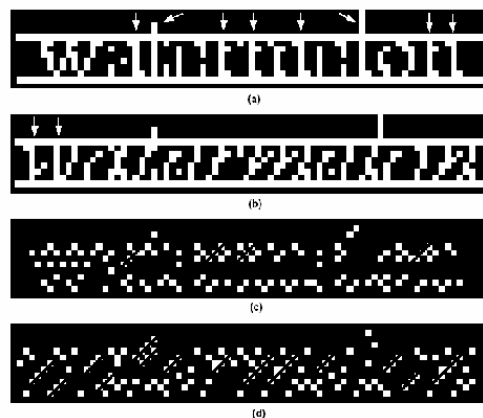


Fig. 4 (a)(b) VLSL running through the mid zone for normal and italic styles respectively (c)(d) CDS for normal and italic styles respectively (length ≥ 3)

Fig. 3 shows an example of the one-level wavelet decomposed sub-images in horizontal, vertical and diagonal directions. The word image “European” is of size 37×154 pixels and is extracted from a scanned paper document with noise distortions. It is normalized to have an “x-height” of 7 pixels in order to obtain good features with the use of Sym2 wavelet

during the decomposition. Obviously, vertical strokes come out especially in the vertical channel, horizontal strokes in the horizontal channel and diagonal strokes in the diagonal channel. These are typical and dominant features that are less sensitive to distortions and font style variations such as size, serifness, bold, etc.

5.2 Stroke Pattern Analysis

The sub-images generated during wavelet decomposition step contain ample information in terms of vertical, horizontal and diagonal stroke patterns, which can be effectively utilized to distinguish italic and non-italic fonts. Specifically, statistical analyses of both vertical and diagonal stroke patterns are performed on the corresponding sub-images and are combined to produce a discriminative recognition measure.

5.2.1 Vertical Stroke Analysis

It is observed that among 22,384 frequently used English words, in over 99% of normal word images and over 14% of italic word images, at least two vertical straight line segments (VSLs) run through the mid zone, as indicated by the arrows in Fig. 4(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 fonts. Therefore, by analyzing the horizontal histogram of the combined horizontal and vertical sub-images, the mid zone is detected and VSLs are identified. Suppose a word of width W (in pixels) has N VSLs with height (h_1, h_2, \dots, h_N) respectively, their normalized total height H is obtained as follows:

$$H = \frac{1}{W} \sum_{i=1}^N h_i \quad (4)$$

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

5.2.2 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 Fig. 4(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 M is the number of CDSs with length (l_1, l_2, \dots, l_M) , their normalized total length L is obtained as follows:

$$L = \frac{1}{W} \sum_{i=1}^M l_i \quad (5)$$

Experiments show that over 98% of italic word images satisfy the criterion that $M \geq \mu$ and $L \geq L_0$, while only about 2% of normal word images satisfy this criterion. Here, μ and 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.

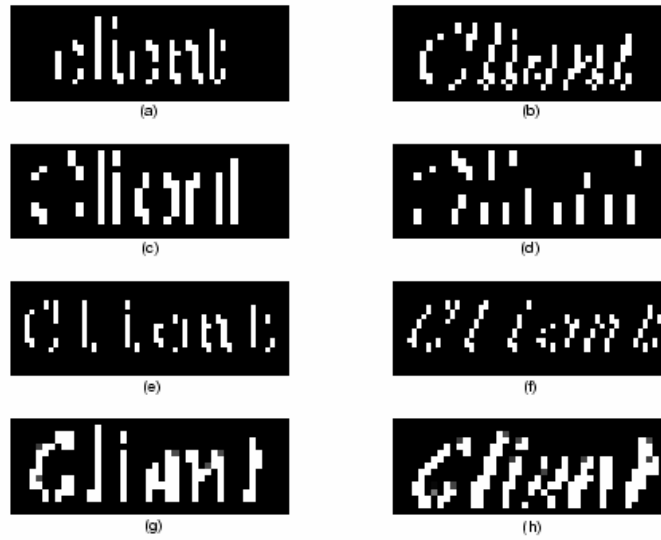


Fig. 5 Examples of wavelet decomposed vertical sub-image in normal and italic styles
 (a)(b) Times New Roman (c)(d) Arial (e)(f) Courier (g)(h) Comic Sans MS

5.3 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 four different fonts of size 12pt are shown in Fig. 5. Experiments conducted on 5,320 normal word images and 489 italic images extracted from scanned paper documents show an accuracy of 92.20% for normal style and 97.96% for italic style as shown in Fig. 6. Comparisons between the traditional stroke analysis method and our

approach are done on documents with mixed normal and italic words in four commonly used fonts. The average recognition accuracies are shown in Fig. 6.

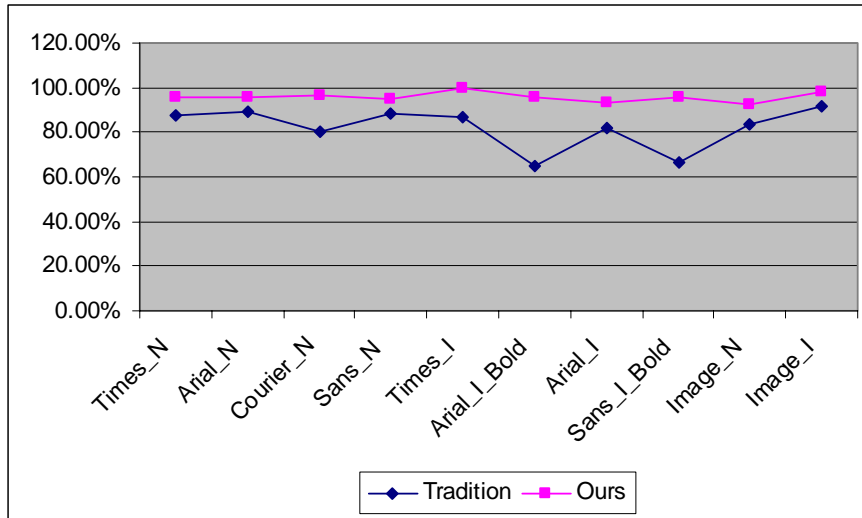


Fig. 6 Recognition accuracy comparisons

6. WEB-BASED RETRIEVAL SYSTEM

6.1 System Overview

This web-based retrieval system is an application of our word image coding technique. First of all, some image pre-processing procedures are carried out on the original document images as described in section 3. These include connected component detection, word object bounding, skew estimation and rectification (if applicable), italic font detection and rectification, etc. 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. All these image processing operations are carried out off-line prior to the online search process, which reduces the query response time during retrieval.

On the client side, users can input a set of query words for AND/OR/NOT operations. 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 result table that stores all the matching documents with their URLs and the normalized term frequencies will be generated and a ranked list of documents 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 section 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 matching documents 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, the feature code representation of the query word is generated by matching each character with a standard PST based on the word image coding mechanism as described in section 2.5. Then, an inexact string matching algorithm is employed to perform feature code matching in the underlying feature code files stored on the server. An overview of the system work flow is shown in Fig. 7.

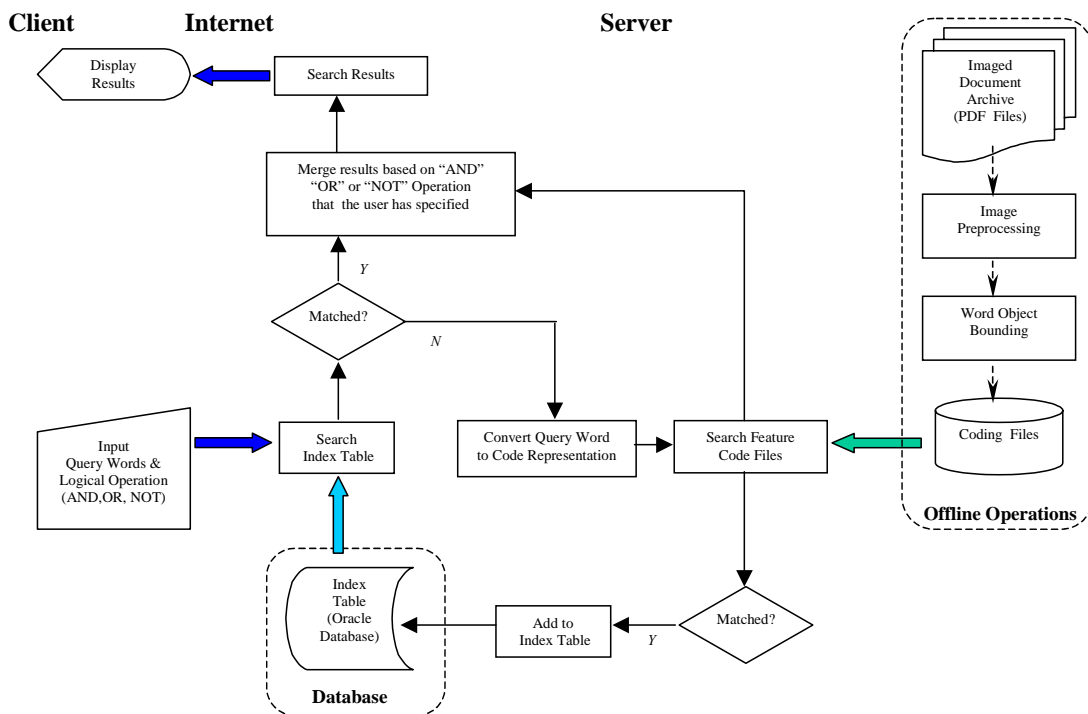


Fig. 7 Overview of the web-based document image retrieval system

6.2 System Evaluation

The performance of the online retrieval is evaluated based on three measured parameters: precision, recall and retrieval speed. Fig. 8 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 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 0.2 second for each search word). An example of this scenario is shown in Fig. 9. The other scenario is when some 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 new word. If there are newly found matches, the index table will be updated and the users will be informed. This is usually time-consuming (about 8 seconds for a database of 80 documents) because for each word we have to search every underlying feature code file in order to identify the matches.

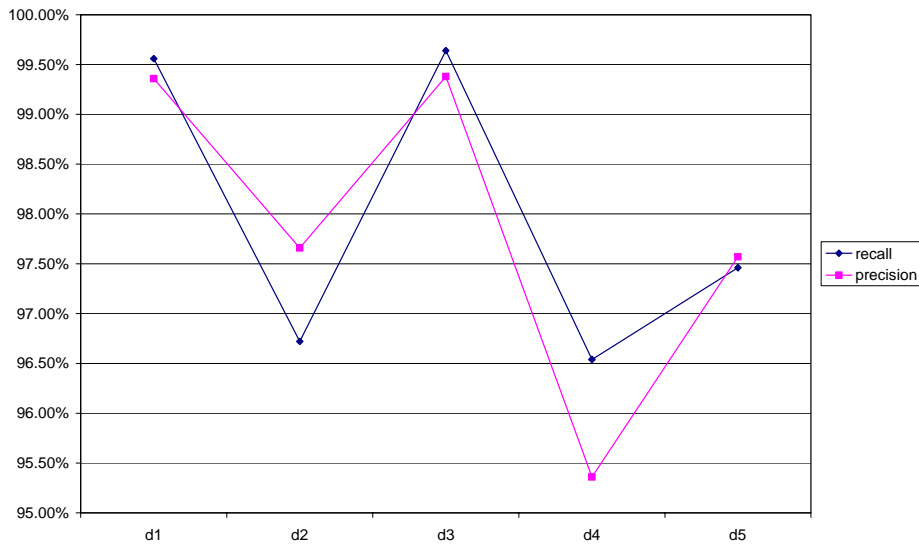


Fig. 8 Recall and precision chart for different categories of documents
d₁=clean documents, d₂=noisy documents, d₃=query on normal fonts,
d₄= query on bold fonts, d₅=query on italic fonts

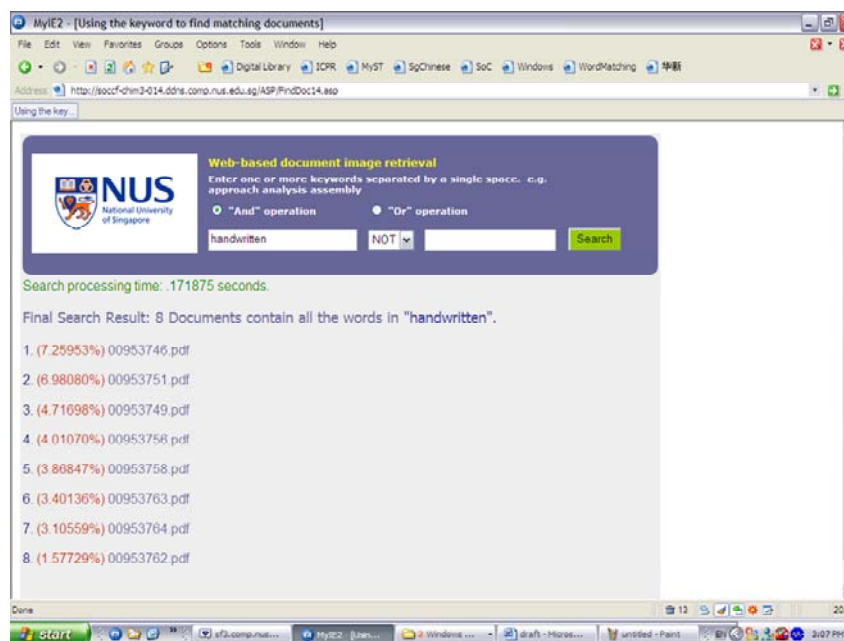


Fig. 9 Search result for pre-queried word

7. SEARCH TOOL IN ADOBE READER

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 (Mckinley, 1996). 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 (Lu, 2004). Experiments show that it is approximately 2.6 times faster than the Page Capture provided by Adobe Acrobat.

7.1 Implementation

The search tool for document images works just like the normal word search tool provided by 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. 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 explicitly locate the matching words in the documents.

When the search is executed on an opened document image, a set of preprocessing steps are first performed as illustrated in section 3. All the connected components in the image are detected and word objects are bounded using a merger operation. Extracted word bitmaps are represented using a feature code string based on the coding mechanism described in section 2. When a user inputs a query word through the search prompt, its corresponding primitive code string is generated by concatenating the characters' PSTs according to the character sequence in the word object. 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 section 4.

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 IEEEExplore 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. Fig. 10 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 Fig. 11. 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.

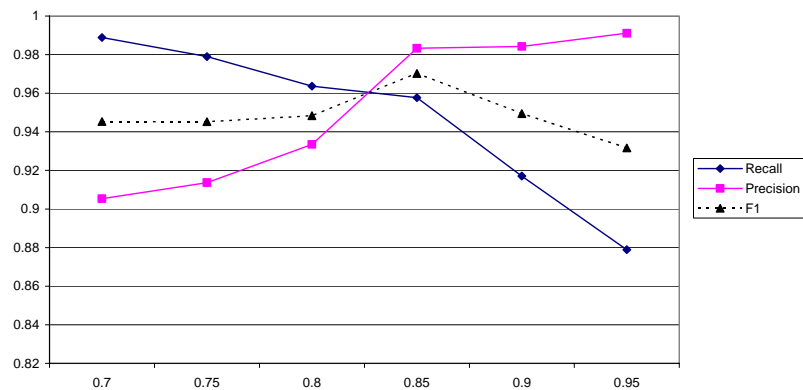


Fig. 10 Performance vs. different thresholds

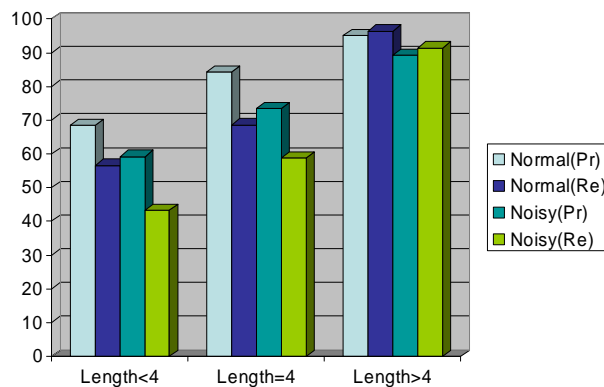


Fig. 11 Recall and Precision wrt word length distribution and noise level

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 tool. 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 tool 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 Fig. 11 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 tool 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 tool 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.

8. CONCLUSION AND FUTURE WORKS

In this paper, we presented a novel word image coding technique that represents each word image object extracted from imaged documents using a unique 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, two experimental applications are implemented to perform information retrieval in imaged documents, which 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 tool for imaged documents in PDF format. It is a typical plug-in search tool embedded in Acrobat Reader that explicitly locates the query word in the PDF file opened either from a local machine or through a web link.

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. Though currently it 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. On the other hand, the web-based document image retrieval system with the underlying index table stored in an Oracle database still needs to be well trained in order to show its retrieval efficiency and intelligence. The search engine for PDF document images currently only works 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.

REFERENCES

- [1]. Doermann D. (1998), "The Indexing and Retrieval of Document Images: A Survey", *Computer Vision and Image Understanding*, vol. 70, no. 3, pp. 287-298.
- [2]. Galloway E.A., Gabrielle V.M. (1998), "The Heinz Electronic Library Interactive On-line System: An Update", *The Public-Access Computer Systems Review*, vol. 9, no. 1.
- [3]. Tagvam K., Borsack J., Condir A., Erva S. (1994), "The Effects of Noisy Data on Text Retrieval", *Journal of the American Society for Information Science*, vol. 45, no. 1, pp. 50-58.
- [4]. Kameshiro T., Hirano T., Okada Y., Yoda F. (1999), "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.
- [5]. Katsuyama K., Takebe H., Kurokawa K. (2002), "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.
- [6]. Spitz A.L. (1997), "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.
- [7]. Smeaton A.F., Spitz A.L. (1997), "Using Character Shape Coding for Information Retrieval", *Proc. of the Fourth Int'l Conf. on Document Analysis and Recognition*, pp. 974-978.
- [8]. Spitz A.L. (1999), "Shape-based Word Recognition", *Int'l Journal on Document Analysis and Recognition*, vol. 1, no. 4, pp. 178-190.
- [9]. Spitz A.L. (2002), "Progress in Document Reconstruction", *Proc. of 16th Int'l Conf. on Pattern Recognition*, vol. 1, pp. 464-467.
- [10]. Chen F.R., Bloomberg D.S. (1998), "Summarization of Imaged Documents without OCR", *Computer Vision and Image Understanding*, vol. 70, no. 3, pp. 307-319.

- [11]. Trenkle J.M., Vogt R.C. (1993), "Word Recognition for Information Retrieval in the Image Domain", *Symposium on Document Analysis and Information Retrieval*, pp. 105-122.
- [12]. He Y., Jiang Z., Liu B., Zhao H. (1999), "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.
- [13]. Lu Y., Zhang L., Tan C.L. (2004), "Retrieving Imaged Documents in Digital Libraries Based on Word Image Coding", *International Workshop on Document Image Analysis for Libraries*, CA, USA.
- [14]. Lu Y., Tan C.L. (2003), "Improved Nearest Neighbor Based Approach to Accurate Document Skew Estimation", *International Conference on Document Analysis and Recognition, ICDAR 2003*, Edinburgh, UK.
- [15]. Apostolico A., Giancarlo R. (1999), "Sequence Alignment in Molecular Biology", *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 47, pp. 85-115.
- [16]. Lopresti D., Zhou J. (1996), "Retrieval Strategies for Noisy Text", *Proc. of the Fifth Annual Symposium on Document Analysis and Information Retrieval*, LA, NV, pp. 255-269.
- [17]. Lopresti D. (2001), "A Comparison of Text-based Methods for Detecting Duplication in Scanned Document Databases", *Information Retrieval*, vol. 4, no. 2, pp. 153-173.
- [18]. Mallat S. (1998), "A Wavelet Tour of Signal Processing", San Diego, CA: Academic.
- [19]. Mallat S. (1989), "A Theory for Multiresolution Signal Decomposition: the Wavelet Representation", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674-693.
- [20]. Mckinley T. (1996), "Acrobat Capture vs. OCR: Apples and Oranges", *Intelligent Imaging*.
- [21]. Lu Y., Zhang L., Tan C.L. (2004), "A Search Engine for Imaged Documents in PDF Files", *27th Annual International ACM SIGIR Conference*, UK.