

# A Fast Keyword-Spotting Technique

Linlin Li, Shijian Lu and Chew Lim Tan  
Department of Computer Science, National University of Singapore  
Kent Ridge, Singapore 117543  
{lilinlin,lusj,tancl}@comp.nus.edu.sg

## Abstract

*In order to capture the content of an imaged document but avoid the time-consuming full-scale OCR which is fragile to handle touching characters, a fast and segmentation-free keyword spotting method is proposed in this paper. The keyword spotting method is based on word shape coding technique. The proposed coding scheme has little ambiguity, and can be swiftly executed. It is a promising technique to boost better document image retrieval. The strength of the proposed method is demonstrated in a document filtering experiment. The experimental results show that document filtering based on the proposed method is more than 20 times faster than the one based on OCR, and has comparable filtering accuracy.*

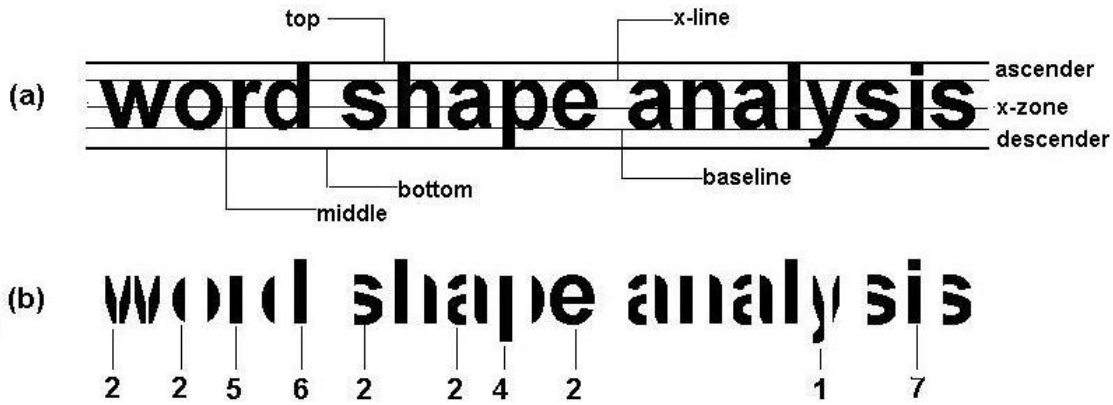
## 1. Introduction

To date, many efforts have been made to build digital libraries which digitize high-volume archives of paper documents (patent, legal tomes, historical documents) to provide the public with free and easy on-line access. These digital libraries store scanning images, which keep visual information such as layout and decorations. However, this leads to difficulties in document retrieval, because traditional text information retrieval techniques totally fail when documents are simply presented as raw bit-maps. A feasible solution is OCR, but current OCR softwares are not always able to provide accurate and reliable document image transcriptions. Based on character-segmentation, OCR performance degrades dramatically when touching adjacent characters appear frequently in an image. Furthermore, OCR technique requires very long execution time. Therefore, for information retrieval applications such as document filtering which are based on locating a few important keywords appearing in a document, it is a waste to resort to a full scale OCR. More importantly, for these databases with large volumes of document images, the time complexity makes it impractical to convert all images in to text by OCR.

A technique known as word shape analysis is proposed as an alternative to full-scale OCR. The technique is supposed to be faster and more reliable when document images are of bad quality. A complete survey of word shape analysis techniques could be found in [4] [6]. To date, word shape coding techniques could be roughly divided into two groups, both of which represent a word image as a whole unit instead of recognizing each character. The first group [3] is based on analyzing pixel-level features of the whole word image, such as intensity autocorrelation and moments. In these approaches, each word image in a document is represented by feature vectors. These approaches are language independent and very roust to poor image quality, but they require appropriate training sets. Besides, the vector is difficult to be indexed. The second group is based on word shape coding [7] [8] [5]. Word shape coding encodes a word image into a sequence of predefined symbols. The symbol set is often smaller than the character set and is easier to be recognized. Each word has a unique corresponding symbol string, while one symbol string may be mapped to several real words because of the reduced symbol set, which is referred as ambiguity. For a language, the limited number of character arrangement may help to avoid ambiguity or reduce it to an acceptable level, which will be discussed later in this paper. Compared with the first group, these approaches have advantages such as that they are easy to form the query, easy to index and training free. On the other hand, the drawbacks are language dependent and not robust as the first group.

We propose in this paper a fast and segmentation-free keyword spotting technique. Keyword spotting is to locate the occurrences of a given keyword from an image. The proposed method has two components: word shape coding and similarity estimation. It is directly related to retrieval applications like Boolean retrieval and document filtering, and it is promising to facilitate better document image retrieval with more sophisticated IR models.

This paper is organized as the following. In sections 2 and 3, the related works and the proposed keyword spotting method are introduced in detail. Section 4 introduces



**Figure 1. (a) A text image showing the text line parameter positions: top, x-height, baseline, and bottom, and the zones defined by them. (b) Decomposing “word shape analysis” into strokes.**

the similarity measure for code strings. The strength of the keyword spotting technique will be illustrated in a document image filtering application in section 5. Section 6 is a conclusion section.

## 2. Related Works

Word shape coding is an important word spotting approach, which takes a word image as input, and outputs a sequence of word shape codes. It maps the character set of a language to a smaller symbol set based on character shape features. Many word shape coding schemes have been proposed and been applied in various applications like language identification and document image similarity estimation. Three coding schemes which are considered representative will be explained in detail in this section. The scheme proposed by Tan et al [8] is based on the vertical bar pattern. 3 different codes are employed in total. Vertical bars are extracted by pairing up local minimum and local maximum pixels located on the contour of the word. Further, vertical bars are classified into three categories depending on whether they have ascenders or descenders. Bars protruding into neither descender nor ascender zones are coded as ‘m’; bars protruding into only descender zones are coded as ‘q’; bars protruding into only ascender zones are coded as ‘b’. This coding scheme causes heavy ambiguity. From their experiment, it is 2-6 times faster than commercial OCR software for different datasets. Spitz’s method [7] has 6 codes in total, and is character based, which means it has a one to one map from real English characters to the shape codes. In this method, character cells are firstly detected by connected component analysis. Each character cell then is classified by the presence of features like ascender/descender, the number of components and deep east-

ward concavity. This scheme is claimed as three or two orders of magnitude as fast as commercial OCR. Lu’s method [5] is stroke-based, and has a complex code set which has 29 codes. Firstly, straight strokes and traversal strokes from the word image are extracted by maximum run-length analysis. Strokes lie in vertical or diagonal directions are considered as straight strokes, and the residue are traversal strokes. Each stroke is described by a two-tuple  $(\sigma, \varpi)$ , where  $\sigma$  is based on ascender/descender attribute, and  $\varpi$  is based on the shape of a stroke. There are only 29 possible combinations in English; hence each of the 29 possible two-tuples are represented by a code each. With 29 codes, this coding method has little ambiguity, and it is reported to be about 3 times faster than OCR.

Aiming for spotting single keyword fast in a document image, these schemes have their shortcomings. Tan’s method has heavy ambiguity, and therefore it is suitable for imaged document similarity estimation, but is not able to handle spotting with a few keywords as queries. Spitz’s scheme has similar problems. Besides, it also relies on correctly character-segmentation as OCR, therefore is not able to deal with touching characters. Lu’s scheme doesn’t have a desirable speed.

## 3. Methodology

In this section, the word shape coding scheme is elaborated. The coding scheme is stroke-based, and has 8 codes in total. A word image is firstly decomposed into a sequence of strokes. A shape code is assigned to each stroke. The decomposition of word image into strokes depends on the pixels lying on the middle line as shown in figure 1 (a). If a pixel on the middle line is OFF, all pixels in the column are turned OFF, and otherwise pixels remain the same state.

Figure 1 (b) is an example where the phrase “word shape analysis” is decomposed into strokes. Each stroke is separated by several blank columns. In real applications, imperfect printing or scanning make word images to have blur intersection, and therefore the strokes are too wide to be decomposed by pixels on the middle line. In this case, we use a middle zone instead of the middle line. The middle zone is defined as a rectangular zone lying on the middle line, with a width (number of pixels) equal to half an average vertical stroke width. Vertical stroke widths are collected by horizontal run length analysis, and the average vertical stroke width is estimated by finding the peak in the histogram.

In order to detect ascender and descender features. The top line, x-line, baseline and bottom line (shown in figure 1 (a)) are extracted for each text line by examining the horizontal projection. These four lines define the boundaries of three significant zones on each text line. The area between the bottom and the baseline is the descender zone; the area between the baseline and the x-line is the x-zone; and the area above the x-line is the ascender zone. However, this detection method fails when the text line is too short or when the whole text line contains no ascender or descender. The coding of strokes is based on the presence of straight vertical/non-straight vertical strokes, ascender and descender and the number of components, as shown in table 2 (“1+” in the column “Number of Components” means “more than one components”). Examples of coded strokes could be found in figure 1(b).

**Table 1. The mapping of strokes to shape codes**

Code	Vertical Straight	Descender	Ascender	# of Components
1	NO	YES	NO	1+
2	NO	NO	NO	1+
3	NO	NO	YES	1+
4	YES	YES	NO	1
5	YES	NO	NO	1
6	YES	NO	YES	1
7	YES	NO	YES	2
8	YES	YES	YES	2

Each English character could be represented with a code string. Table 2 shows a complete list of English characters in both lower and upper cases with the corresponding word shape code strings. It is worth to be noted that, in different font character g may have different code strings. For example, character ‘g’ in Times New Roman is encoded as “11”, while in Arial it is encoded as “14”. Since no valid code string for any English word contains “11” except words which has character ‘g’ of Roman, all “11” appearing

**Table 2. The codes for characters in English**

Char.	Codes	Char.	Codes	Char.	Codes
a	25	H	66	g	14/11
bk	62	m	555	q	24
ceszY	2	nu	55	h	65
vAV	22	d	26	i	7
fitEFJLPT	6	p	42	r	5
CSZ	3	N	626	w	2222
BDKR	63	M	6226	j	8
GOQUX	33	W	2222	y	12

a code string is replaced by “14”. The code strings for an English word are concatenating code string for each character in order. For example, the code string for “word shape analysis” is “2222222526 26525422 255525612272”.

#### 4. Similarity Estimation

In order to measure the similarity between two code strings *query* and *str*, an approximate matching based on minimum edit distance is proposed as below:

$$1 - \frac{MinEditDis(query, str)}{10 * \log_{10}(length(query))} \geq \delta$$

where *length* is the number of codes in a code string ( $length(query) > 1$ ); the value of  $\delta$  is from 0.7 to 1; and *MinEditDis* is the function to calculate Levenshtein distance. Levenshtein distance or edit distance between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character. In our experiment, the cost for each operation equals to 1, and hence the return value of *MinEditDis* is a non-negative integer. The formulation shows that a higher  $\delta$  allows lower edit distance, which means strings are more similar and vice versa.

#### 5. Experiments and Results

Two experiments will be carried out for different emphasis in this section. The first one is word spotting on self-prepared image dataset. The dataset has 300-dpi binary images scanned 23 printed-pages with various character sizes and fonts. In this experiment, the focus is whether the coding scheme works well on different fonts and sizes under a well controlled situation. The second one is document filtering application on ISRI DOE2&3 collections. It is designed to demonstrate the strength of the proposed method in document image retrieval domain. ISRI DOE2&3 are public image collections from ISRI [1], which contains 1670

scanned pages as well as the associated ground truth text. These images are generated from various ways including directly scanning from journal pages, scanning from first or later generation photocopies. Therefore the dataset has unexpected fonts, character sizes, noise as well as skew.

Because the coding is sensitive to skew, images must first be deskewed [2]. Manual zone information is provided. In case where these is headlines, footnotes or drawings zones, they are removed from the images. In each text zone, word bounding boxes are extracted by examining the projection profile. Each query will be translated into two code strings: lower case and capital initial.

### 5.1. Keyword spotting

The first experiment is designed to show that the proposed keyword spotting is able to deal with different fonts and sizes. There are 23 pages which have 174221 words in total. Each page has a randomly chosen character size from 10 to 24 points, and a font out of four, which are Dotum, Century, Times New Roman, and Arial. These fonts are chosen because they are widely-used and representative subset of the major fonts which exhibit the range of characteristics of machine-generated text. Style is also an important issue. However, it is easy to see that the method can deal with Bold style, but not Italic style. After all pages are tokenized, the approximate matching algorithm is used to compare queries and strings in the documents. 50 keywords were generated, which appear at least 15 times in the dataset. There are 1845 appearances for all the keywords. The precision of spotting one keyword  $P$  is defined as the number of correctly detected keywords over the number of detected words, while the recall of spotting one keyword  $R$  is defined as the number of correctly detected keywords over the number of actual keyword appearances.  $F1$  is defined as  $F1 = \frac{2*P*R}{P+R}$ . The average recall, precision and  $F1$  measure of the keyword spotting for several thresholds is shown in table 3.

From table 3, the proposed word spotting method achieves the best precision/recall value as 96.22%/90.08%, while OCR has a performance as 98.33%/96.08%. The result may suggest the proposed word spotting is robust to different fonts and character sizes. Besides, table 3 also shows it is very easy to adjust the precision/recall pair according to different application requirements. Lower threshold brings high recall but low precisions, because the coding scheme itself has ambiguity among different words, approximate matching with a loose constraint will aggravate the situation. On the contrary, higher threshold brings high precision and adequate recall. When  $\delta$  is around 0.92, the average f1 achieves the highest value. This may be because the threshold is low enough to compensate coding mistakes (the coding scheme assigned a stroke a wrong code) , while

high enough to avoid bringing more ambiguity.

**Table 3. The keyword spotting performance**

$\delta$	0.7	0.88	0.9	0.92	1
P	0.3020	0.9294	0.9399	0.9622	0.9994
R	0.9601	0.9223	0.9133	0.9008	0.7205
F1	0.4100	0.9169	0.9173	0.9205	0.7612

### 5.2. Document image filtering

Document filtering task is one of the most important document retrieval applications. It emphasizes on speed as well as on indexing methods that enable very fast processing of documents against profiles. The second experiment is to simulate a real document filtering task. The document filtering is based on keyword matching, that is, If the query appears in a document, this document is considered as relevant. 50 single-keyword queries are manually generated from the ground truth data. The average recall and precision measure of the document image filtering task for several thresholds is shown in table 4. From the table, document filtering performance based on the proposed method achieves the best precision/recall value as 91.00%/77.01%, while the one based on OCR has a best performance as 93.7%/88%. On the average, it took 0.145 seconds to process a document image by the proposed method, while it took 2.99 seconds to process a document image by OCR. The proposed method generated comparable accuracy, yet was 20 times faster than OCR.

Two factors that may affect the keyword spotting performance are ambiguity and coding errors. The former means that a code string may map to several words and the later means assigning a wrong code to a stroke. Presumably, ambiguity brings higher recall but lower precision; on the contrary, coding errors bring lower recall. In table 4, when the performance is the best, precision (91%) is much higher than recall (77%). This phenomenon may indicate that the coding errors dominate the performance other than ambiguity.

**Table 4. The document filtering performance based on the keyword spotting**

$\delta$	0.70	0.80	0.88	0.9	1
P	0.5928	0.8889	0.9100	1.0000	1.0000
R	0.8048	0.7804	0.7701	0.5612	0.5612
F1	0.6827	0.8289	0.8341	0.7189	0.7189

### 5.3. Speed issue and ambiguity

An important issue for keyword spotting is speed. Both OCR and Coding processes include deskewing (the very OCR program that was employed in the experiment needs deskewing when the skew angle is larger than 0.5 degree.), zoning and recognising/coding. Since zoning does not affect the experiment performance, only deskewing and recognising/coding time are taken in account. As shown in table 5, in the experiments it took 0.73 seconds and 3 minutes to code all images in both datasets respectively, while a commercial OCR spent 3.06 seconds and 62 minutes respectively to transcript those images. Hence, the keyword spotting is more than 20 times faster than OCR, and yet has comparable performance. The coding speed is expected to be further improved by code optimization.

**Table 5. Running time comparison for OCR and coding**

Dataset	No. of pages	OCR time	Coding time
Self-prepared	23	3.06s	0.73s
ISRI DOE	1245	62min	3min

The coding ambiguity is quantified by estimating collision rate of the coding scheme. The collision rate is defined as the difference between the number of words and the number of corresponding identical codes over the number of words as the formula below:

$$collision\ rate = \frac{\#\ of\ words - \#\ of\ codes}{\#\ of\ words}$$

A lexicon of 55,000 most frequently used English words is employed. For comparison purpose, the collision rate for three other word shape coding schemes besides the one we proposed are also demonstrated in table 6. It is shown that inherited ambiguity makes Tan's [8] and Spitz's [7] coding schemes not suitable for single keyword spotting. The proposed coding scheme has a collision rate of 0.0619%, and Lu's [5] coding scheme also has little ambiguity.

**Table 6. The collision rate for four word shape coding schemes**

Coding Scheme	Stop-Words 120	Non-Stop-Words 54880	<i>n</i> times faster than OCR
Huang's	0.4622	0.4231	2-6
Spitz's	0.4351	0.2366	2-3 orders of magnitude
Ours	0.2304	0.0619	20-40
Lu's	0.0083	0.0023	3

## 6. Conclusion and future works

This paper introduces a new keyword spotting technique, which avoids the difficulties of separating touching adjacent characters in a word image and extensive computation during recognition. The experiment results show that it is very fast and yet generates comparable accuracy. It is promising to act as an alternative to full-scale OCR in some document image retrieval applications.

However, there are still two issues which need further exploration. The first issue is the sensitivity of skew and noise. From the experiment, a good performance shows when it works on real images of good quality (300 dpi, small skew, and little salt and pepper noise). More experiments are still necessary for degraded images in order to quantify the sensitivity of skew and noise of the proposed method, and to design more robust word shape coding schemes. The second issue is ambiguity. Since our ultimate goal is to boost better document image retrieval techniques, the effect of the ambiguity on different document retrieval applications is also worth detailed discussion.

## 7. Acknowledgement

This research is supported by A\*STAR grant 0421010085 and NUS URC grant R252-000-202-112.

## References

- [1] <http://www.isri.unlv.edu/ISRI/>.
- [2] D. S. Bloomberg, G. E. Kopec, and L. Dasari. Measuring document image skew and orientation. *Proc. SPIE, Document Recognition II*, 2422:302–316, 1995.
- [3] F. Chen, L. Wilcox, and D. Bloomberg. Detecting and locating partially specified keywords in scanned images using hidden markov models. *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 133–138, Oct. 1993.
- [4] D. Doermann. The indexing and retrieval of document images: A survey. *Computer Vision and Image Understanding: CVIU*, 70(3):287–298, 1998.
- [5] Y. Lu and C. L. Tan. Information retrieval in document image databases. *IEEE transactions on knowledge and data engineering*, 16(11):1398–1410, November 2004.
- [6] M. Mitra and B. Chaudhuri. Information retrieval from documents: A survey. *Information Retrieval*, 2(2-3):141–163, May 2000.
- [7] A. L. Spitz. Using character shape codes for word spotting in document images. *Proceedings of the Third International Workshop on Syntactic and Structural Pattern Recognition*, 1994.
- [8] C. L. Tan, W. Huang, S. Y. Sung, Z. Yu, and Y. Xu. Text retrieval from document images based on word shape analysis. *Applied Intelligence*, pages 257–270, 2003.