

# Keyword Spotting in Document Images through Word Shape Coding

Shuyong Bai, Linlin Li and Chew Lim Tan  
School of Computing, National University of Singapore  
{baishuyo, lilinlin, tanc1} @ comp.nus.edu.sg

## Abstract

*With large databases of document images available, a method for users to find keywords in documents will be useful. One approach is to perform Optical Character Recognition (OCR) on each document followed by indexing of the resulting text. However, if the quality of the document is poor or time is critical, complete OCR of all images is infeasible. This paper build upon previous works on Word Shape Coding to propose an alternative technique and combination of feature descriptors for keyword spotting without the use of OCR. Different sequence alignment similarity measures can be used for partial or whole word matching. The proposed technique is tolerant to serifs, font styles and certain degrees of touching, broken or overlapping characters. It improves over previous works with not only better precision and lower collision rate, but more importantly, the ability for partial matching. Experiment results show that it is about 15 times faster than OCR. It is a promising technique to boost better document image retrieval.*

## 1. Introduction

In today's digital age, thousands of documents and books are being digitized and stored as document images daily. Modern technology has made it possible to produce, store and transmit these document images efficiently. The traditional way of searching through these images is to perform OCR and convert them to machine-readable characters. However, performing OCR on large volumes of document images might not be cost efficient and practical.

One reason is that OCR is prone to character segmentation errors. Touching adjacent characters such as "rn", "lc", "vv" are sometimes recognized wrongly as "m", "k", "w" (and vice versa) respectively. The recognition rate of OCR software typically decreases dramatically when touching adjacent characters appear frequently in the image.

Thus, human correction of the OCR results is needed and that can prove to be infeasible for large volume of documents. OCR is a time-consuming process and a major bottleneck for information retrieval systems. Myers [1] reported in their OCR-based information extraction system that the total processing time was dominated by the character recognition process. As a result, document images do not normally go through the OCR process and is stored as document images. Nowadays, on the Internet, we can find many documents in image format, such as e-books, journals and patent documents. Many digital libraries such as ACM, IEEE keep scanned document images without their respective text equivalents.

If the goal is to retrieve some relevant documents from a large document image database and the exact words are not needed, then performing OCR on the entire document corpus is too expensive. Thus, a keyword-spotting technique is proposed to enable a user to search the images for keywords and filter out the relevant documents. This is done by first converting each word into a Word Shape Code (WSC). A similarity measure is then used to allow for either partial or whole word matching. This technique is about 15 times faster than OCR as it does not require any time-consuming template matching. It is also tolerant to certain degrees of degraded images such as broken, overlapping or touching characters.

The remainder of this paper is organized as follows: Section 2 surveys previous related works in document image retrieval. Section 3 describes the feature extraction and word shape code. Section 4 presents the similarity measure for the word codes. Section 5 will show the experimental results of the proposed word shape coding method. Section 6 is the conclusion.

## 2. Related Works

In recent years, a number of attempts have been made by researchers to avoid the use of character recognition for various document image retrieval applications. For example, Nakayama uses character shape codes for content word detection and document

image categorization, without character recognition. In his approach, he used connected components to identify characters and converts each character image into one of the seven codes using features such as ascenders, descenders, south and eastward concavity.

In later works, Spitz take a character shape approach for keyword spotting [2], language identification [3] and document image categorization. A set of 5 character shape codes encode whether or not the character image fits between the baseline and x-line or if it has an ascender or descender. Similarly, each connected component is converted into a code.

The above techniques rely on character segmentation and annotate each connected component with a single character code. The limitation of such an approach is that it is prone to character segmentation errors. Document images of low quality often contain many touching or broken characters, severely affecting the accuracy of the resultant shape codes. To avoid the difficulties of separating touching characters in a word image, we use a segmentation-free approach that treats each word image as a single, indivisible component and attempts to recognize the features of the word as a whole. By doing so, the word recognition process is more tolerant to character segmentation errors.

In previous works on Word Shape Coding, Li [4] uses a stroke-based method for keyword spotting. A word image firstly decomposed into a sequence of strokes by using pixels lying on the middle line. Each intersection with the middle line is then assigned a number '1' to '8' based on a combination of features. However, the method is too over-reliant on the middle-line and can often lead to wrong coding as it is not font-invariant. By using vertical strokes, it is also unable to handle italics font style.

Lu [5] uses the character extremum points to convert a word into a sequence of numbers '1', '2', '3' followed by the number of horizontal cuts of the word. However, with only 3 character codes to represent all words, the coding scheme has high ambiguity where many different words are mapped to similar word code. Due to its high-ambiguity, it is also not able to perform partial matching.

In Lu's [6] other work, he uses a combination of topological character shape features including character ascenders, character holes and water reservoirs for keyword spotting and document filtering. However, with his method, common and important characters in an English word such as 'm', 'n', 'r', and 'u' are not coded. For example, in the word "number" or "tunnel", only the letters "ber" and "tel" are being extracted with the rest ignored. This leads to high ambiguity, high collision rate and low precision when performing word spotting. As such, the method is also unable to perform partial keyword spotting.

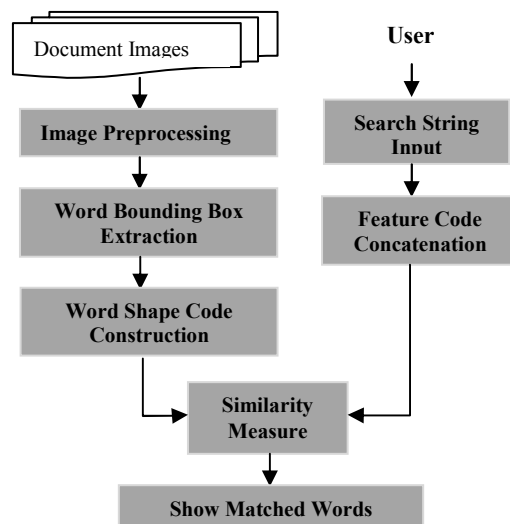


Fig 1. System diagram for keyword spotting

### 3. Proposed Technique

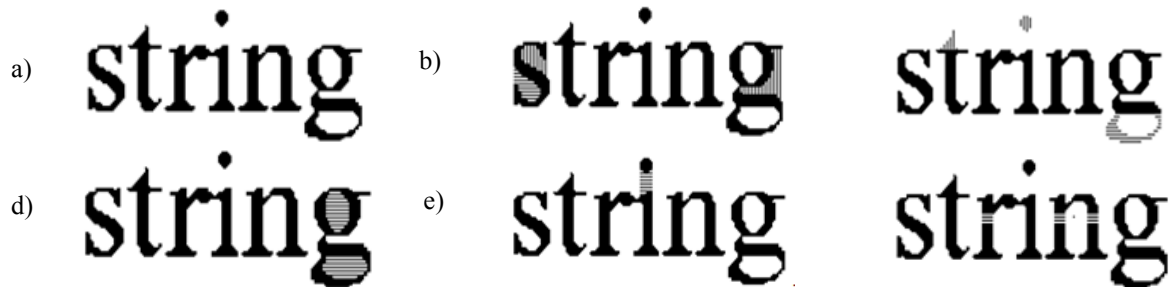
In our paper, we propose a fast keyword spotting approach with the ability to for partial matching. This technique can also be applied in other areas of document image retrieval such as document image categorization and language identification. Figure 1 highlights the steps for the keyword spotting process. The main focus here will be the word shape code and similarity measure process.

Archived document images often suffer from various types of degradation. Thus, it is assumed that noise removal and skew estimation and correction are done. Textline and word bounding boxes are then extracted through a page segmentation process.

#### 3.1 Word Shape Feature Extraction

For each word bounding box, a total of 7 different features are detected and the word converted into a WSC that describes the features in a left to right manner. The features used are character ascenders, descenders, deep eastward and westward concavity, holes, i-dot connectors and horizontal-line intersection. Figure 2 shows the features on a binarized word image.

In order to extract these features in a single pass through the word bounding box, we make use of run-length connected components and sandwiched vertical black and white runs. Sandwiched vertical black or white runs are maximal sequences of black or white pixels in a column that has the dual of its run color before and after it. A vertical run is specified by 3 integers  $(x, y, z)$  where  $x$  is the column,  $y$  and  $z$  are the row numbers of the first and last pixels respectively.



**Fig 2. The feature descriptors. a) original image b) deep eastward concavity shaded in vertical lines and deep westward concavity in horizontal lines. c) ascender shaded in vertical lines and descender in horizontal lines d) character holes e) i-dot connector f) horizontal-line intersection shaded.**

To extract character holes, we make use of run-length connected components by grouping together sandwiched white runs adjacent to each other. Two runs,  $(x_1, y_1, z_1)$  and  $(x_2, y_2, z_2)$ , are adjacent if the following constraint is satisfied:

$$|x_1 - x_2| = 1 \ \& \ y_1 - 1 \leq y_2 \ \& \ z_1 + 1 \geq z_2$$

More formally, we can define a sandwiched white run block as a maximal sequence of sandwiched white runs  $R_1, R_2, \dots, R_n$  such that  $R_i$  is adjacent to  $R_{i+1}$  and  $y_i - 1$  and  $z_i + 1$  are black pixels for all runs,  $1 \leq i \leq n - 1$ .

Using the above described method, features such as character holes, deep eastward or westward concavity can be easily found by simply by checking if all pixels to the left (right) of the first (last) white run block are all black. In order to avoid shallow concavities caused by noise and serifs, the number of runs in the white run block,  $n$ , for identifying concavities should exceed a heuristic threshold of 0.18 of the x-height.

Old documents typically have broken characters like the 'a' and 'g' shown in Figure 3, making it difficult to properly extract the correct features. To increase the chances of correctly extracting the features, adjustments are made to merge components together to form larger components so that would not be filtered out later. Instead of looking for run blocks that are directly next to the current white run to group together, run blocks within a distance of 3 pixels should be considered. Should more than 1 sandwiched white run blocks be within that distance, the tiebreak condition shall be the minimum amount of non-overlap (max amount of overlap) calculated as follows:

$$\min\{abs(y_c - y_1) + (abs(z_c - z_1))\}$$

where  $(x_c, y_c, z_c)$  is the current run-length and  $(x_l, y_l, z_l)$  is the run-length that the current run-length is trying to connect to. White runs that are more than 60% of the x-height are not considered as they tend to be caused



**Figure 3. Word image with broken characters**

by serifs instead of broken characters. With these adjustments, the features of word image in Figure 3 can be properly extracted.

Ascenders (descenders) can be identified by black run blocks that starts above (ends below) the x-line (baseline). Horizontal-line intersection are horizontal black runs that cross the imaginary line that is 2/5 between the x-line and baseline and do not have any other features above, below or directly beside the black run. For example, the 's' and 't' in Figure 2f do not have horizontal-line intersection feature as it already has concavity and ascender feature lying above the horizontal-line intersection feature respectively.

The reason why we do not use the middle-line intersection is that it tends to be error-prone. For example, the middle-line that intersects a character 'a' might vary from font to font. Thus, by using a line that is slightly above the middle-line, it is more tolerant to different fonts.

To extract all the features, the word bounding box is scanned vertically column by column to look for white run blocks and black run blocks that lie above or below the x-line or baseline respectively. For each column, we keep track of its horizontal-line pixel to look for horizontal-line intersections. A filtering step is then performed to filter out non-relevant features.

### 3.2 Word Shape Coding

Based on the position and type of the features, a sequence of feature codes is annotated to each word. The x-centroid of each white or black run block is first calculated by the following formula:

$$C_x = \frac{\sum_{i=1}^n (z_i - y_i) * x_i}{\sum_{i=1}^n (z_i - y_i)}$$

where  $(x_l, y_l, z_l)$  to  $(x_n, y_n, z_n)$  corresponds to the leftmost and rightmost run in the white run block.

The features are then grouped together based on their x-centroid positions. Features with x-centroids that are within a specific threshold of 0.34 of the x-

height are grouped together and assigned a unique feature code. Standalone features such as ascenders, descenders, character holes, deep eastward concavity, deep westward concavity, i-dot connector and middle-line intersection are coded by the characters ‘l’, ‘n’, ‘o’, ‘x’, ‘c’, ‘i’ and ‘r’ respectively. Table 1 shows the proposed coding scheme for the 52 Roman letters.

There are three ways the feature code of a character can be derived. Firstly, the alphabet ‘t’ only has one ascender feature and is thus coded by its feature code ‘l’. On the other hand, the letter ‘x’ has 2 features – a deep westward concavity followed by a deep eastward concavity – that are not within the threshold specified and is thus annotated as ‘xc’. Lastly, letters such as ‘e’ has two features – hole and eastward concavity – whose x-centroids are almost vertically aligned, and within the specified threshold. These 2 features are thus grouped together and assigned a unique code ‘e’.

It is interesting to note that the character ‘g’ has two types of typography. In san-serif, it has 3 vertically aligned features and is written as ‘g’; whereas in serifs, it has 4 vertically aligned features and is written as ‘g’. In both cases, their combinations of features are within the predefined threshold and are grouped together. Since no other characters share their combination of features, they are uniquely identified and coded as ‘g’.

The word “string” in Figure 2 is annotated by the shape code “slrriirg” which is a concatenation of its individual feature codes based on Table 1. Note that this proposed word coding scheme is tolerant to character segmentation - the ‘r’ and ‘i’ in Figure 2 are touching at the x-line position but they can still be properly annotated. It is also tolerant to certain degrees of broken characters or overlapping characters.

#### 4. Similarity Measure

After the word shape coding process, each word image is described by a concatenation of feature codes. We make use of a globalised sequence alignment similarity measure for keyword spotting.

Given a query shape code,  $S_1 = [a_1 a_2 a_3 \dots a_m]$  and word shape code,  $S_2 = [b_1 b_2 b_3 \dots b_n]$ , the score  $GS(S_1, S_2)$  can be computed by dynamic programming. Let  $M$  be a matrix and  $M(i, j)$  be the score of the optimal alignment between  $a_1 a_2 \dots a_i$  and  $b_1 b_2 \dots b_j$ . The base conditions are:

$$\forall i, j : \begin{cases} M(i, 0) = -0.6 * i \\ M(0, j) = -0.6 * j \end{cases}$$

The general recurrence formula relation is:

$$M(i, j) = \max \begin{cases} M(i-1, j-1) + \varepsilon(a_i, b_j) \\ M(i-1, j) - 0.6 \\ M(i, j-1) - 0.6 \end{cases}$$

for  $1 \leq i \leq m, 1 \leq j \leq n$

**Table 1. Feature codes of the 52 letters.**

Character	Feature Code	Character	Feature Code
a	a	x	xc
b	b	y	yr
c	c	z	z
d	d	A/P	A
e	e	B	B
f	f	C/G	C
g	g	D/O	O
h	lr	E	E
i	i	F	f
j	j	H/U/V	ll
k	lc	I/J/L/T	l
l/t	l	K	K
m/w	rrr	M/N/Y	lrl
n/u/v	rr	Q	Q
o	o	R	R
p	p	S	S
q	q	W	lll
r	r	X	XC
s	s	Z	Z

where cost of matching,  $\varepsilon(a_i, b_j)$ , is defined as:

$$\varepsilon(a_i, b_j) = \begin{cases} 1 & \text{if } a_i = b_j \\ -0.3 & \text{if } a_i \neq b_j \end{cases}$$

For whole word matching, the normalized similarity score is in the range (0, 1):

$$GS_{WholeWord}(S_1, S_2) = \frac{M(m, n)}{M^*(m, m)}$$

where  $M^*(m, m)$  is the similarity score between the query string  $S_1$  and itself.

For partial word matching, the base condition is changed in order to ignore non-matching prefixes:

$$\forall i, j : \begin{cases} M(i, 0) = -0.6 * i \\ M(0, j) = 0 \end{cases}$$

The recurrence formula remains the same while the similarity score is changed to the following in order to ignore any non-matching suffixes:

$$GS_{PartialWord}(S_1, S_2) = \frac{\max_{1 \leq j \leq n} (M(m, j))}{M^*(m, m)}$$

If the normalized similarity score is greater than a predefined threshold,  $\delta$ , then the word that corresponds to  $S_2$  is classified as a matched word.

**Table 2. Time Needed for Word Shape Coding**

	1 PAT Doc. (10 pages)	10 Text pages
Omnipage v16	31.210 s	58.152 s
Proposed method (Layout Analysis + Word Shape Coding)	2.065 s (1.395 s + 0.670 s)	4.405 s (2.560 s + 1.485 s)

**Table 3. Partial Spotting with PAT Dataset**

$\delta$	0.80	0.85	0.9	0.95	1.00
P	84.09%	93.63%	99.72%	100%	100%
R	99.80%	98.81%	98.78%	98.78%	98.78%
$F_1$	91.27%	96.15%	99.25%	<b>99.38%</b>	<b>99.38%</b>

**Table 4. Whole Word Spotting with PAT Dataset**

$\delta$	0.80	0.85	0.9	0.95	1.00
P	69.08%	72.54%	85.55%	93.66%	100%
R	97.92%	97.92%	97.72%	87.90%	86.23%
$F_1$	81.01%	83.34%	91.23%	90.69%	<b>92.61%</b>

**Table 5. Partial Word Spotting with UW1**

$\delta$	0.80	0.85	0.9	0.95	1.00
P	83.25%	92.14%	97.54%	99.51%	99.52%
R	94.51%	90.11%	89.55%	83.01%	81.46%
$F_1$	88.52%	91.11%	<b>93.37%</b>	90.51%	89.59%

## 5. Experiments and Results

The time taken to perform the keyword spotting was benchmark on a 2 sets of documents with our proposed method and a commercial OCR, Omniapge v16. The first being a patent image of about 10 pages which consist of figures and complex layouts while the second set consists of 10 pages of double-column, full-text documents. Docstrum analysis was used for page segmentation (layout analysis). The results in Table 2 show that our method is about 15 times faster.

In another experiment, 2 sets of document datasets are used to determine the performance of our coding scheme. The first dataset has 50 patent documents (PAT) totaling 636 pages with moderately good quality texts. The dataset has a total of 645156 words of which 322578 are unique. The second dataset has 30 pages of real world document images from the University of Washington (UW1) dataset.

40 words of length between 4 to 12 characters are randomly chosen. For each word, both partial and whole word spotting are performed and their precision (P) and recall (R) rates are recorded. Table 3 and 4 summarizes the average precision and recall rate of the 40 words using different thresholds with the PAT dataset. Table 5 shows the performance of partial word spotting on the UW1 dataset.

From the experimental results, a threshold of 1.00 yields the best  $F_1$  measure for PAT dataset. For whole word spotting, when searching for “example”, matched results such as “examples” are counted as false positives. This explains the low precision rate when a low threshold is used. For partial word spotting with the UW1 dataset, the best  $F_1$  measure is when  $\delta = 0.9$ .

**Table 6. Comparison with other coding schemes**

	Collision	Precision	Recall	$F_1$
Spitz [2]	26.1%	70.2%	91.5%	79.5%
Lu [6]	16.8%	91.5%	84.3%	87.8%
Proposed	<b>6.1%</b>	100%	86.2%	<b>92.6%</b>

For good keyword spotting, the collision rate (different words mapped to the same code) should be low. Table 6 compares the collision rate using the 12Dicts English Wordlist of about 40,000 words which is calculated by  $\frac{\# \text{ of words} - \# \text{ of code}}{\# \text{ of words}}$ . It also compares the performance of our proposed coding scheme with whole keyword spotting technique from Spitz [2] and Lu [6] on the PAT dataset. The results show that our coding scheme outperforms previous coding schemes in both the collision rate and  $F_1$  measure.

## 6. Conclusion

This paper improves upon previous work on keyword spotting without OCR. Experimental results show that it is robust. Unlike previous methods, it is able to perform partial and whole word matching. Thus, it is a promising technique as an alternative to full-scale OCR when the full text is not required. There are other areas of document image retrieval such as document filtering, document image categorization and language identification in which the proposed word shape code scheme can also be used.

## Acknowledgements

This research is supported in part by IDM R&D grant R252-000-325-279.

## References

- [1] G. K. Myers and P. G. Mulgaonkar, "Automatic extraction of information from printed documents," in *Proc. of 18th Annual ACM SIGIR* 1995, pp. 328-335.
- [2] A. L. Spitz, "Using Character Shape Codes for word Spotting in Document Images," *Shape, Structure and Pattern Recognition*, pp. 382-389, 1995.
- [3] A. L. Spitz, "Determination of the Script and Language Content of Document Images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pp. 235-245, 1997.
- [4] L. Li, S. Lu, and C. L. Tan, "A Fast Keyword-Spotting Technique," in *Proc. Of the 9th ICDAR*, 2007, pp. 68-72.
- [5] S. Lu and C. L. Tan, "Retrieval of machine-printed Latin documents through Word Shape Coding," *Pattern Recognition*, pp. 1799-1809, 2008.
- [6] S. Lu, L. Li, and C. L. Tan, "Document image retrieval through word shape coding," *IEEE Trans. On PAMI*, vol. 30, no. 11, pp. 1913-1918, Nov. 2008.