Journal of Universal Computer Science, vol. 17, no. 1 (2011), 48-63
submitted: 8/4/10, accepted: 12/12/10, appeared: 1/1/11 © J.UCS

An OCR Free Method for Word Spotting in Printed Documents: the Evaluation of Different Feature Sets

Israel Rios
(Pontifical Catholic University of Parana, Curitiba, Brazil
isrios@ppgia.pucpr.br)

Alceu de Souza Britto Jr
(Pontifical Catholic University of Parana, Curitiba, Brazil
alceu@ppgia.pucpr.br)

Alessandro Lameiras Koerich
(Pontifical Catholic University of Parana, Curitiba, Brazil
alekoe@ppgia.pucpr.br)

Luis Eduardo Soares Oliveira
(Federal University of Parana, Curitiba, Brazil
lesoliveira@deinf.ufpr.br)

Abstract: An OCR free word spotting method is developed and evaluated under a strong experimental protocol. Different feature sets are evaluated under the same experimental conditions. In addition, a tuning process in the document segmentation step is proposed which provides a significant reduction in terms of processing time. For this purpose, a complete OCR-free method for word spotting in printed documents was implemented, and a document database containing document images and their corresponding ground truth text files was created. A strong experimental protocol based on 800 document images allows us to compare the results of the three feature sets used to represent the word image.

Keywords: word spotting, document retrieval, word recognition
Categories: I.5, I.7

1 Introduction

With the advances in information technology observed in the last years, it is usual to find large volumes of information available in digital format. A large amount of this information is composed of scanned document images. Due to the large volume, there is the urgency to provide fast access methods to this information. However, the current tools for indexing and searching in large databases are not prepared to deal with this type of data. Moreover, the use of OCR-based methods has shown itself an expensive option of the computational point of view [Doermann, 98]. An interesting alternative is the group of methods that aim to make possible the word spotting in document images without using OCR. In such an approach, the methods have as advantage a small execution time, and the robustness to noisy documents [Balasubramanian, 06], [Lu, 04], [Lu, 02], [Rath, 03].

A good example of such an approach is the method proposed in [Lu, 04]. The authors have proposed a feature string matching method based on a dynamic programming algorithm and an inexact feature string matching. The feature set employed to represent word bitmap images is the Left-to-Right Primitive String (LRPS), which is a code string sequenced from the leftmost of a word to its rightmost. Line and traversal features are used to extract the primitives of a word image. The authors have reported experimental results, where precision and recall rates are evaluated, on various document images such as scanned books, student theses, and journal/conference papers downloaded from the Internet, as well as on the UW document images. Some interesting results were observed. The method achieves a precision ranging from 89.06% to 99.36% and a recall ranging from 85.67% to 99.19%.

Another interesting method is presented by Rath and Manmatha [Rath, 03] to deal with handwritten words. The authors have employed very simple features based on projection profile, upper word profile, lower word profile and column transitions to represent handwritten word bitmap images. Since the word image is represented with shape features that result from sampling a function of the image at each image column, DTW (Dynamic Time Warping) was applied to word image matching. Precision of 72.56% and recall of 65.17% were reported for a document manuscript database.

In both works, the key point is the feature set used to represent a word image as a string of features. The objective of this paper is to compare these feature sets, and a third one, originally proposed for character recognition [Arica, 00], by using the same experimental protocol. For this purpose, we have developed a complete word spotting method which is OCR–free and capable of searching words in a scanned document image as a string in ASCII format.. Similar to the work of Lu and Tan [Lu, 04], the proposed framework has the property of finding words inside of words through the use of a feature string matching algorithm which is able to provide a measure of similarity between two strings of features and to carry out a partial correspondence between them. Moreover, a significant improvement in terms of time reduction is obtained in the segmentation of the document image into words.

This paper is organized as follows. Section 2 describes the implemented method. Section 3 presents the experiments used to evaluate the different feature sets. Conclusions and future works are presented in Section 4.

2 Method Overview

The implemented method for word spotting in printed documents is illustrated in Figure 1 and encompasses several algorithms: pre-processing, segmentation of the document image into word images, feature extraction from the word images, conversion of ASCII queries into descriptors, and word matching. With this framework, we may evaluate specific strategies for segmentation, feature extraction, and word matching.

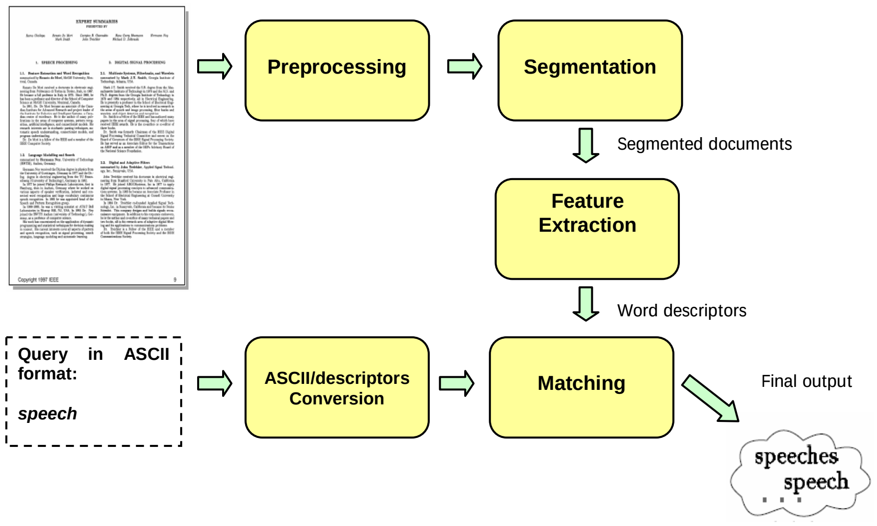


Figure 1: An overview of the implemented word spotting method

2.1 Preprocessing

The original image of a document is first binarized by using the Otsu method [Otsu, 78] followed by a smoothing process [Suen, 92] to reduce the noise in the contour of the character strokes which may be produced during the acquisition process. The masks in Figure 2 and their respective rotations by 90^o, 180^o and 270^o were used in the smoothing process. In these masks the code "1" represents the foreground, code "0" represents the background, while the code "?" represents "don´t care". Figure 3 shows an example of character "h" before and after the smoothing process.

?	1	?
1	0	1
?	1	?

?	1	?
1	0	1
0	0	1

1	1	0
1	0	0
1	1	0

0	0	?
0	1	1
0	0	1

0	0	0
0	1	1
0	0	?

Figure 2: Set of masks used for contour smoothing

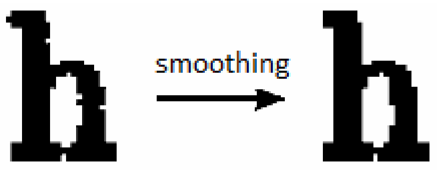


Figure 3: The effect of the smoothing process

2.2 Segmentation

The first step in the segmentation process consists of finding all connected components (CCs) in the binary image of the document. For each CC, shape features are calculated such as: the ratio between height and width, and the ratio between black and white pixels inside the connected component bounding box (CCBB). These features are used in a filtering process which is dedicated to eliminate table lines, figures and all kind of graphics present in the document. The CCs that go through this filter have their position and dimension stored in linear data structures (LDS). In order to accelerate the access to each CC, one LDS is created for each vertical and horizontal strip (50 pixels large) as shown in Figure 4, and the procedure used to store the CCs is based in the following rules:

1) a CC must belongs, at least, to a vertical and a horizontal strip;
2) CCs that belongs to more than one strip must belong to more than one LDS;
3) if a CCBB overlaps other CCBB both must be merged and re-inserted into the corresponding LDS.

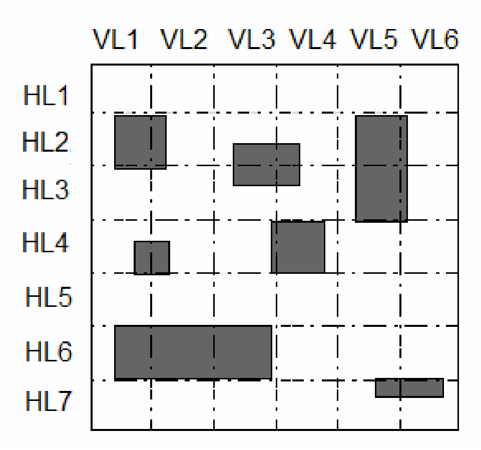


Figure 4: A document, its CCs in dark gray, and the vertical (VL1..VL6) and horizontal (HL1..HL7) linear data strutures related to the document strips

Different from the method proposed in [Breuel, 02], we have considered a modified strategy to search for empty spaces in the document. The empty spaces may represent the spaces between the text blocks, a text block and a table or a figure, and so on. Basically, the original method uses a divide-and-conquer strategy to find empty rectangles in the document image. However, we have considered first the division of the rectangles that have the highest height/width ratio. With this modification, we can divide the rectangles without leave common regions among them and with this we can favor the column detection as shown in Figure 5.

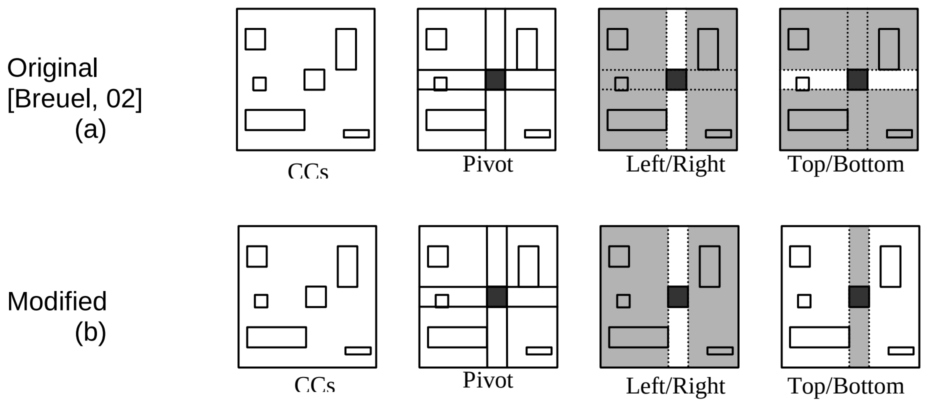


Figure 5: Searching for vertical and horizontal empty rectangles in a document page: a) original Breuel method; b) modified method in which the priority is the rectangles with highest height/width ratio

2.3 Feature Sets

We have evaluated three different feature sets. The left-to-right primitive string (LRPS) feature set proposed by Lu and Tan [Lu, 04][Lu, 02], the Arica and Yarman-Vural (AYV) descriptors [Arica, 00] and the upper/lower profiles and transition count (ULTC) feature set proposed by Ratha and Manmatha [Rath, 03].

2.3.1 Left-to-Right Primitive String (LRPS)

The LRPS word descriptors were originally proposed for information retrieval in document image databases. Each feature in this set is a pair of attributes (σ, ω) where σ is the Line-or-Traversal Attribute (LTA) and ω is the Ascender-and-Descender Attribute (ADA). The first attribute is calculated taking into account an analysis of the straight lines in the word image and the number of transitions that does not belong to straight lines. The second attribute is obtained through the analysis of the feature position (straight line or a transition) by considering the ascending and descending lines. These attributes provide structural information of the word image. They are scale and translation invariant. A detailed description of these attributes may be found in [Lu, 04] and [Lu, 02]. Figure 6 provides an overview of the LPRS features.

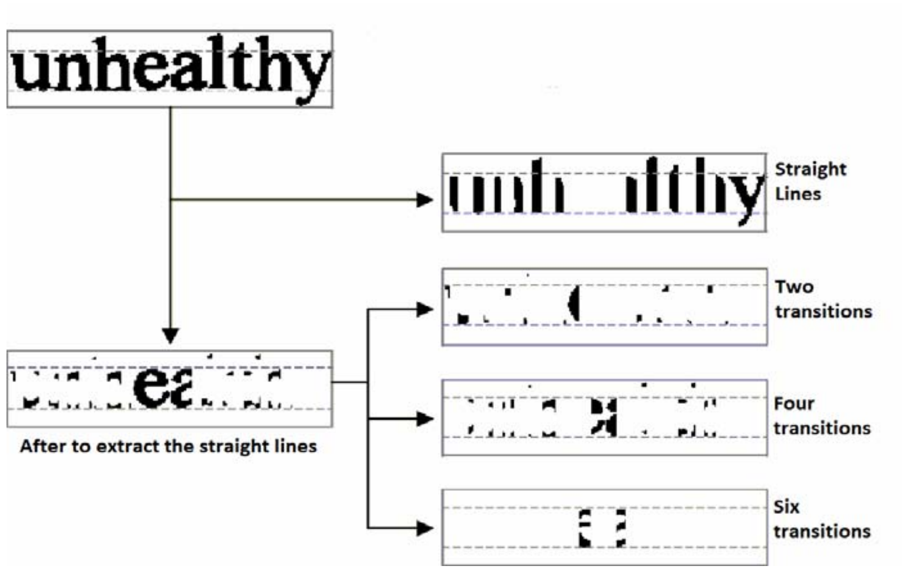


Figure 6: The LRPS features proposed in [Lu, 04]: based on straight lines and transitions

2.3.2 Arica and Yarman-Vural Descriptors (AYV)

The *AYV* descriptors are based on the Arica and Yarman-Vural features [Arica, 00] used for isolated character recognition. We have adapted the features proposed by the authors for the representation of printed word images. For this purpose, different from the original method, only the vertical columns were take into account, since we have to employ them for both the words and the isolated characters. In addition, in our modified AYV features no size normalization method is considered. Figure 7 illustrates the descriptors computed for two columns of the character “a”.

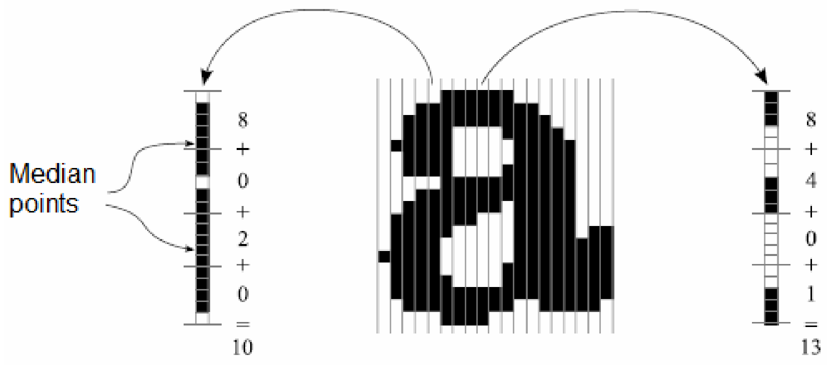


Figure 7: The AVY features proposed in [Arica, 00] - computed as 10 and 13 for columns 5 and 9 of a character “a”

For each column, the following steps are employed:

1) Divide the column into four regions with the same size;
2) Each region receives the code 2^i , where i = region_index – 1;
3) Compute the median point M of each sequence of black pixels;
4) Sum the region code in which M lies to the final descriptor.

Thus, the descriptor for each column is the sum of the code regions in which the median point of the sequence of black pixels are located.

2.3.3 Upper/Lower Profiles and Transition Count (ULTC)

This feature set is composed of projection profile, upper word profile, lower word profile and column transitions as shown in Figure 8.

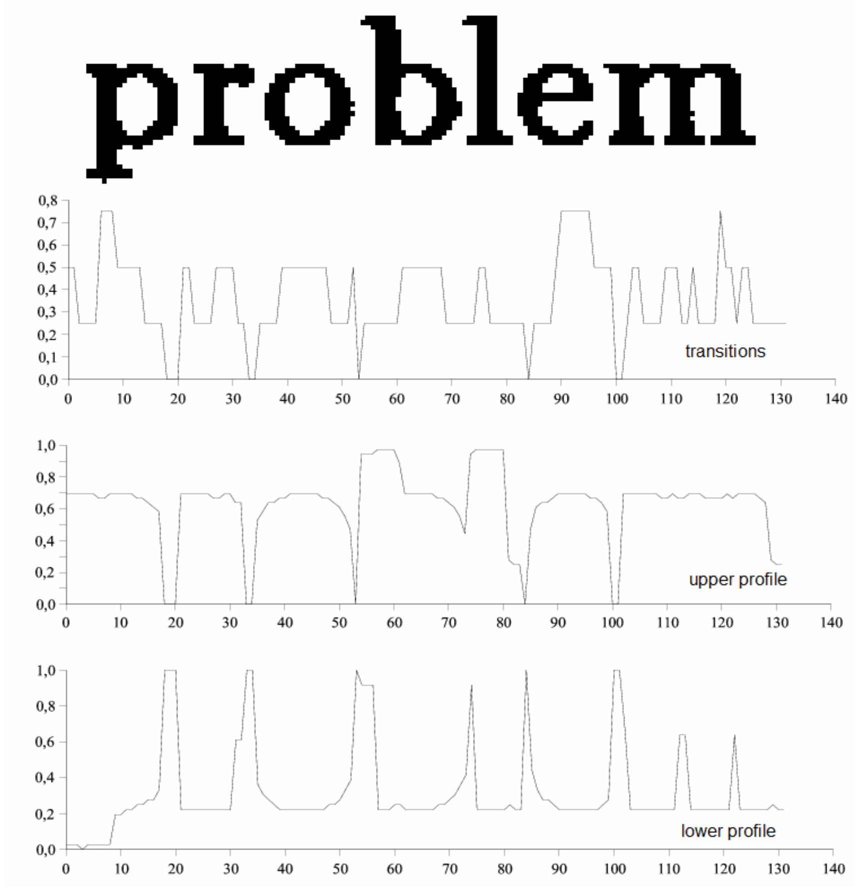


Figure 8: The ULTC features proposed in [Rath, 03]: transitions, upper and lower profiles computed for the word "problem"

The profile based features are calculated such as in [Rath, 03]. The projection profile is the sum of the pixel intensities for each word column. The upper word profile is calculated based on the distance from the top of the bounding box to the closest black pixel in that column. Similarly, the lower word profile is calculated from the bottom of the bounding box. All these features are normalized in the interval [0, 1]. These features are very robust to noise; however, they are not enough to discriminate some printed characters, for instance: “a” and “o”.

To deal with this situation, column local features based on the number of transitions from white to black pixels and vice-versa are added to the feature set. For this feature extraction method, we have used two conversion tables to generate the word descriptor. These tables are responsible to convert a character in features. One table considers font with serif and the second table considers font without serif. With these two conversion tables, it was not necessary to remove the serifs. The adoption of these two tables has provided some improvements in the method performance as observed in our experiments.

2.4 ASCII/Descriptor Conversion

A word descriptor corresponds to a sequence of features extracted from a word which may be originally provided in ASCII format, i.e, as an ASCII string of characters. To calculate the word descriptor from an ASCII string it is necessary to use a conversion table (see Figure 9).

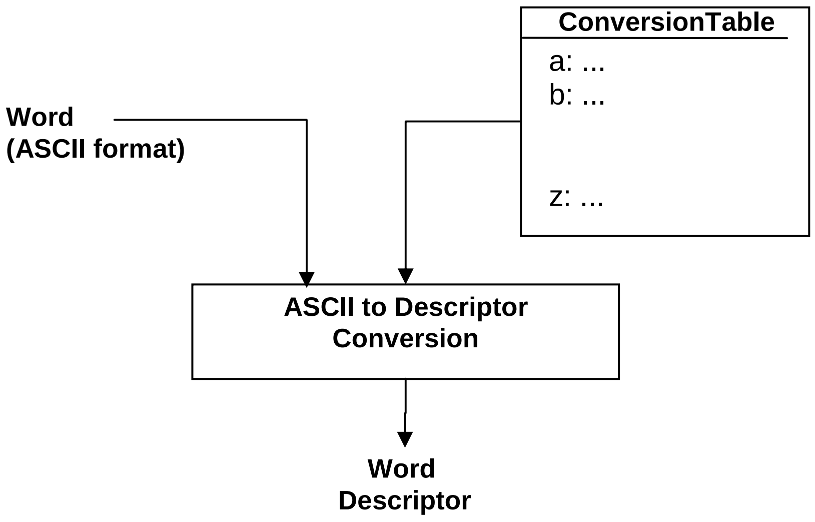


Figure 9: Scheme used to obtain the word descriptor from a word in ASCII format

In [Lu, 04], the authors use a table manually created from the observation of each possible character. In our approach, we have implemented an automatic strategy in which the conversion table is created from an image containing character models: upper and lower cases. With this scheme, it is easier to create the conversion table. In addition, this allows us to create different tables by considering distinct fonts. All character models are synthetically created in a common image editor. Thus, they are not specific for a certain kind of document. The procedure employed to convert an ASCII string *P* representing a word is composed of the following steps:

a) for each character C_i in *P* gets the corresponding descriptor D_i in the respective conversion table;
b) add the character descriptor D_i in the word descriptor *DW*;
c) add the descriptor "&" to represent the space between adjacent characters in *P*.

2.5 Word Matching

The inexact string matching algorithm proposed by Lopresti and Zhou [Lopresti, 96] is used to compute the similarity between two strings. However, we have adapted the algorithm to consider the different feature sets. For the ULTC features – since each column is represented by a feature vector instead of a single discrete value, the distance between the feature vector "a" representing a column of a word in the document and the feature vector "b" representing a column of the searched word is calculated as:

$$D_{UTLC} = \sum_{j=1}^{N} (a[j] - b[j])^2 \tag{1}$$

where *j* is the index and *N* is the dimension of the feature vectors.

For the AYV features, since each column is represent by a single discrete value, the distance between columns is calculated as:

$$D_{AYG} = \begin{cases} 2 \quad if \quad (v_a = v_b) \\ 0 \quad if \quad (v_a \neq v_b) \quad and \quad (v_a = 0 \quad ou \quad v_b = 0) \\ -2 \quad if \quad (v_a \neq v_b) \end{cases} \tag{2}$$

where v_a is a discrete value representing a column of a word in the document image and v_b is a discrete value representing a column of the searched word.

The original strategy proposed in [Lopresti, 96] is used to compute the distance for the LPRS features. The output of the algorithm after comparing both descriptors is a similarity measure, denoted as S , between the word image extracted from a document and the converted ASCII string representing the query word. This similarity measure is compared with a threshold value λ , which is experimentally defined (from 0.65 to 0.85). When $S \geq \lambda$, it means that the searched word is present in the analyzed descriptor. Thus, the corresponding word is selected and its address (document, page and paragraph) is stored. It is possible to observe in the experiments that the value of λ is in directly proportional to the precision and inversely proportional to recall. Precision and recall are the evaluation metrics and both of them are described in Section 3.

3 Experimental Results

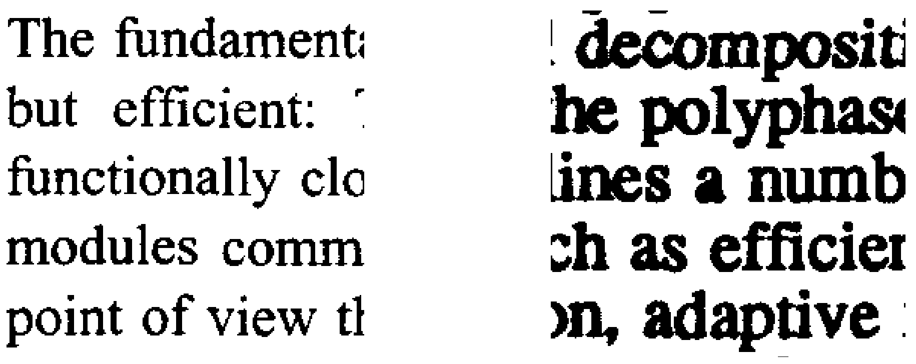
In this section we describe document image database that was used to evaluate the proposed method and the experiments undertaken to evaluate the proposed method and the different feature sets previously described.

3.1 Document Database

The document image database created for our experiments is composed of 865 papers in PDF format, digitized with 300 dpi and published in the ICASSP´97. Most of documents contain four pages in two-column format with text, equations, graphics and tables.

To create a textual version of each document keeping the page layout, the OCR available in the Acrobat [Acrobat, 06] was used. In addition, this textual version was submitted to the Xpdf [Xpdf, 05] to obtain for each word its position in the page, the page number and the document file name. A visual verification of each textual document was carried out. This process does not succeed to obtain the textual version for some low quality documents – which corresponds to about 1% of the initial amount of documents. To obtain a copy of the created database is just necessary to contact the authors.

Figure 10 shows two parts of distint documents, where we can observe some problems, such as: fragmented and touching characters, and a significant variance in terms of stroke thickness, while Figure 11 shows a sample of a document page available in the database.

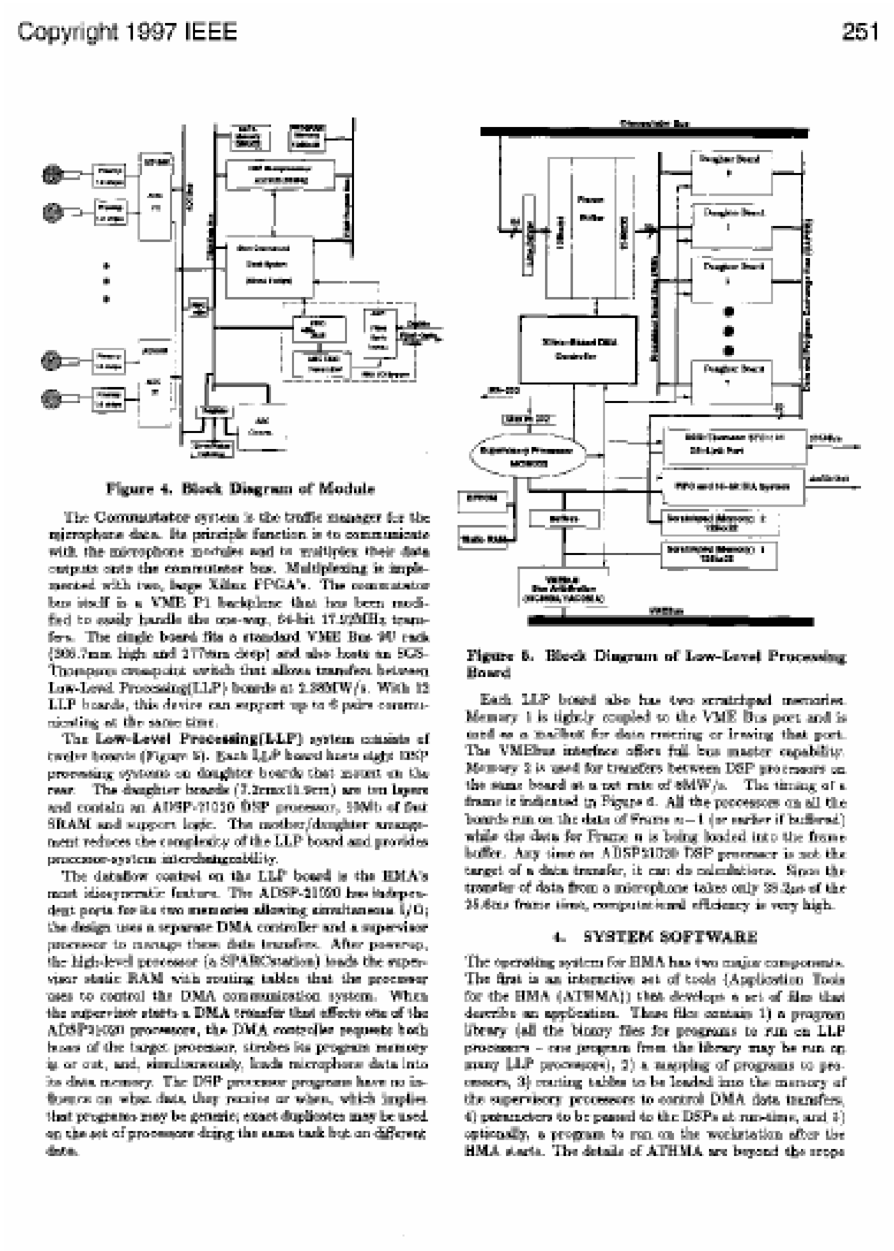


The fundament
but efficient:
functionally clo
modules comm
point of view tl

decompositi
he polyphas
lines a numb
ch as efficier
on, adaptive

(a) (b)

Figure 10: Sample of the variance in the database: (a) and (b) are two parts of distint documents



Copyright 1997 IEEE 251

Figure 4. Block Diagram of Module

The Commutator system is the traffic manager for the microphone data. Its principle function is to communicate with the microphone modules and to multiplex their data outputs onto the commutator bus. Multiplexing is implemented with two, large Xilinx FPGA's. The commutator bus itself is a VME P1 backplane that has been modified to easily handle the one-way, 64-bit 17.92MHz transfers. The single board fits a standard VME Bus 9U rack (366.7mm high and 277mm deep) and also hosts an SGS-Thompson crosspoint switch that allows transfers between Low-Level Processing(LLP) boards at 2.38MW/s. With 12 LLP boards, this device can support up to 6 pairs communicating at the same time.

The **Low-Level Processing(LLP)** system consists of twelve boards (Figure 5). Each LLP board hosts eight DSP processing systems on daughter boards that mount on the rear. The daughter boards (7.2cmx11.9cm) are ten layers and contain an ADSP-21020 DSP processor, 1MWb of fast SRAM and support logic. The mother/daughter arrangement reduces the complexity of the LLP board and provides processor-system interchangeability.

The dataflow control on the LLP board is the HMA's most idiosyncratic feature. The ADSP-21020 has independent ports for its two memories allowing simultaneous I/O; the design uses a separate DMA controller and a supervisor processor to manage these data transfers. After powerup, the high-level processor (a SPARCstation) loads the supervisor static RAM with routing tables that the processor uses to control the DMA communication system. When the supervisor starts a DMA transfer that affects one of the ADSP21020 processors, the DMA controller requests both buses of the target processor, strobes its program memory in or out, and, simultaneously, loads microphone data into its data memory. The DSP processor programs have no influence on what data they receive or when, which implies that programs may be generic; exact duplicates may be used on the set of processors doing the same task but on different data.

Figure 5. Block Diagram of Low-Level Processing Board

Each LLP board also has two scratchpad memories. Memory 1 is tightly coupled to the VME Bus port and is used as a mailbox for data entering or leaving that port. The VMEbus interface offers full bus master capability. Memory 2 is used for transfers between DSP processors on the same board at a net rate of 8MW/s. The timing of a frame is indicated in Figure 6. All the processors on all the boards run on the data of Frame $n-1$ (or earlier if buffered) while the data for Frame n is being loaded into the frame buffer. Any time an ADSP21020 DSP processor is not the target of a data transfer, it can do calculations. Since the transfer of data from a microphone takes only 39.2μs of the 25.6ms frame time, computational efficiency is very high.

4. SYSTEM SOFTWARE

The operating system for HMA has two major components. The first is an interactive set of tools (Application Tools for the HMA (ATHMA)) that develops a set of files that describe an application. These files contain 1) a program library (all the binary files for programs to run on LLP processors – one program from the library may be run on many LLP processors), 2) a mapping of programs to processors, 3) routing tables to be loaded into the memory of the supervisory processors to control DMA data transfers, 4) parameters to be passed to the DSPs at run-time, and 5) optionally, a program to run on the workstation after the HMA starts. The details of ATHMA are beyond the scope

Figure 11: Sample of a document page available in the database

The experimental results are reported in terms of precision (Pr) and recall (Re). The precision is the percentage of the number of correctly searched words over the number of all searched words, while recall is the percentage of the number of correctly searched words over the number of words that should be searched. In order to make easy the comparison of the feature sets previously described, we also use the

F_1 metric which is described in Equation (3), which is calculated in terms of both the precision and the recall.

$$F_1 = \frac{2 \operatorname{Re} \operatorname{Pr}}{\operatorname{Re} + \operatorname{Pr}} \tag{3}$$

The words to be searched in our experiments were selected based on their frequency in the database. Instead of using a random process, we decide to select the 50 most frequent words in the database after removing the stop words. The document database was organized into two subsets: training (50 document images) and testing (815 document images). Table 1 shows the 10 most frequent words in the database and the respective number of occurrences.

Word	**# of occurrences**	**Word**	**# of occurrences**
Speech	5,364	Figure	4,512
Signal	5,312	Noise	3,969
using	5,223	Filter	3,723
algorithm	4,954	Copyright	3,425
model	4,544	number	3,378

Table 1: The 10 most frequent words in the database

3.2 Segmentation

As explained before, the segmentation step consists in finding text blocks, lines and words from the connected components. The word segmentation in a text line is done based on the median distance, denoted as M, between adjacent characters. Two connected components are considered to be part of the same word if the horizontal distance between them is less than αM. The parameter α was experimentally defined on the training dataset as 2.

During the experiments, we have compared the segmentation method proposed in [Breuel, 02] with a modified method in which a different strategy to search the empty spaces in the document was used, as described in Section 2.2. Basically, we gave priority for the division of the rectangles that present the highest height/width ratio to favor the column detection. Table 2 shows the significant reduction in the number of iterations to find the first 40 empty spaces in a document page.

Method	**# of iterations**
Original Segmentation Method	21,393
Modified Segmentation Method	643 (reduction of 96.9%)

Table 2: Average number of iterations in the segmentation process

3.3 Different Feature Sets

Table 3 summarizes the experimental results of the proposed method by considering the LRPS feature set for different similarity thresholds (λ), without using the smoothing process of our preprocessing step.

Similarity (S)	**0.65**	**0.70**	**0.75**	**0.80**	**0.85**
Precision (%)	35.68	55.54	71.31	89.00	95.23
Recall (%)	55.92	45.40	35.40	22.58	12.63
F_1	0.436	0.500	0.473	0.360	0.223

Table 3: LRPS feature set: results (without preprocessing) using different similarity thresholds

We also consider the best similarity parameter observed in Table 3 with the use of the preprocessing step (see Table 4).

Similarity (*S*)	**0.70**
Precision (%)	69.15
Recall (%)	42.98
F_1	0.530

Table 4: Best LRPS feature set setup with preprocessing

Table 5 presents the experimental results considering the AYV features and the use of different similarity thresholds (λ) without smoothing process. Table 6 shows the results for the AYV features but now using the smoothing in the preprocessing step.

Similarity (*S*)	**0.65**	**0.70**	**0.75**	**0.80**	**0.85**
Precision (%)	19.18	38.92	61.86	85.87	94.62
Recall (%)	31.77	23.12	15.11	6.85	2.55
F_1	0.239	0.290	0.243	0.127	0.050

Table 5: AYV features (without preprocessing) using different similarity thresholds

Similarity (*S*)	**0.70**
Precision (%)	40.0
Recall (%)	21.8
F_1	0.283

Table 6: Best AYV setup with preprocessing

Similar experiments were done by considering the ULTC features. Tables 7 and 8 present the results, while in Table 9 we have the comparison of the best results achieved by using each feature set.

Similarity (*S*)	**0.65**	**0.70**	**0.75**	**0.80**	**0.85**
Precision (%)	46.61	65.89	81.42	91.98	96.78
Recall (%)	80.48	70.35	55.63	36.95	19.01
F_1	0.590	0.680	0.661	0.537	0.318

Table 7: ULTC features (without pre-processing) using different similarity thresholds

Similarity (*S*)	**0.70**
Precision (%)	64.84
Recall (%)	71.63
F_1	0.681

Table 8: Best ULTC setup with pre-processing

3.4 Discussion

During the experiments, it was possible to observe that the value of the similarity measure between two strings (λ) is in direct proportional to the precision and in inverse proportional to recall.

Features	**Precision**	**Recall**	F_1	**Sim**
LRPS	69.15%	42.98%	0.530	0.75
AYV	38.92%	23.12%	0.290	0.70
ULTC	65.89%	70.35%	0.681	0.70

Table 9: Best result of each feature set

The experimental results have also shown that the pre-processing provide some improvement when using the LRPS and ULTC feature sets. In both cases, the impact in the recall is positive, but with a small loss in terms of precision.

With the scheme of first selecting the biggest rectangles in terms of height/width ratio during the divide-and-conquer strategy used in the original segmentation algorithm proposed in [Breuel, 02], we have observed a significant reduction (96.9%) in terms to execution time. In addition, the automatic strategy used to generate the conversion table allows us to easily adapt the method for documents with different fonts.

As one can see in Table 9, the best results were achieved by using the ULTC feature set, when the similarity threshold is 0.70. The results of LRPS and ULTC are similar in terms of precision. However, the recall provided by the ULTC features (70.35%) is significantly better than that provided by the LRPS features (42.98%). An additional advantage of the ULTC features is the simple way that it combines global and local information of the word. The AYV feature set presented the worst results. In fact, we observed that the column information provided by this feature set was not enough to represent the word shape variability. Maybe, this feature set can be used in combination with other methods in a future work.

It is not possible to compare the obtained results with those reported in the literature for the LRPS and ULTC features, since the experimental protocol is different. However, we observed that using our database they do not succeed to achieve similar performances.

4 Conclusions

We have implemented a complete OCR-free word spotting method and evaluated three different feature sets by considering the same experimental protocol. In addition, a document database containing document images and their corresponding ground truth text files was created.

Some specific contributions were done at each stage of the developed word spotting method, such as: a significant reduction in terms of the processing time in the segmentation process; an automatic scheme to generate the conversion tables; and the comparison of three feature sets under the same conditions.

Further work may be done by considering the combination of feature sets in the proposed method. In order to make it, we plan to implement some methods for feature selection. In addition, we plan to evaluate the efficiency of the best features for word spotting in a database of historical documents and also to compare the proposed method with OCR-based word spotting approaches.

Acknowledgements

This research has been supported by The National Council for Scientific and Technological Development (CNPq) grants 302358/2008-0.

References

[Acrobat, 06] ACROBAT. Adobe, 2006. Available at: <http://www.adobe.com>.

[Arica, 00] Arica, N., Yarman-Vural, F. T. One-dimensional representation of two-dimensional information for HMM based handwriting recognition. *Patt. Recog. Letters*, 21, 6-7 (June 2000), 583–592.

[Balasubramanian, 06] Balasubramanian A., Meshesha M., Jawahar C., Retrieval from document image collections, Proc. of the International Workshop on Document Analysis Systems, (Nelson, NewZealand), pp. 1–12, 2006.

[Breuel, 02] Breuel, T. M. Two geometric algorithms for layout analysis. *Proc. of the International Workshop on Document Analysis Systems*, (London, UK, 2002), 188–199.

[Doermann, 98] Doermann D., “The Indexing and Retrieval of Document Images: A Survey,” Computer Vision and Image Understanding (CVIU), vol. 70, no. 3, pp. 287–298, 1998.

[Lopresti, 96] Lopresti, D. P., Zhou, J. Retrieval strategies for noisy text. *Proc. Annual Symp. on Document Analysis and Information Retrieval*, (Las Vegas, NV, 1996), 225–269.

[Lu, 02] Lu, Y., Tan, C. L. Word spotting in chinese document images without layout analysis. *Proc. of 16th Intl Conf. on Pattern Recognition* Quebec, Canada, (2002), 57–60.

[Lu, 04] Lu, Y., Tan, C. L. Information retrieval in document image databases. *IEEE Trans. Knowl Data Eng.*, 16, 11 (2004), 1398–1410.

[Otsu, 78] Otsu, N. A threshold selection method from gray-level histogram. *IEEE Trans. Systems, Man, and Cybernetics*, 8 (1978), 62–66.

[Rath, 03] Rath, T. M., Manmatha, R. Features for word spotting in historical manuscripts. *Proc. Intl Conf. Docum. Analysis and Recognition*, Edinburgh, Scotland, (2003), 218–222.

[Suen, 92] Suen, C. Y., Nadal, C., Legault, R., Mai, T.A., Lam, L. Computer recognition of unconstrained handwritten numerals. *Proc. of the IEEE,* 80, 7 (1992) 1162–1180.

[Xpdf, 05] XPDF: A pdf viewer for X. Glyph & Cog, 2005. Available at: <http://www.foolabs.com/xpdf/>.