Automatic Synthesis of Historical Arabic Text for Word-Spotting

Majeed Kassis
Department of Computer Science
Ben-Gurion University of the Negev
Beer-Sheva, Israel
majeek@cs.bgu.ac.il

Jihad El-Sana
Department of Computer Science
Ben-Gurion University of the Negev
Beer-Sheva, Israel
el-sana@cs.bgu.ac.il

***Abstract*—We present a novel framework for automatic and efficient synthesis of historical handwritten Arabic text. The main purpose of this framework is to assist word spotting and keyword searching in handwritten historical documents. The proposed framework consists of two main procedures: building a letter connectivity map and synthesizing words. A letter connectivity map includes multiple instances of the various shape of each letter, since a letter in Arabic usually has multiple shapes depends in its position in the word. Each map represents one writer and encodes the specific handwriting style. The letter connectivity map is used to guide the synthesis of any Arabic continuous sub-word, word, or sentence. The proposed framework automatically generates the letter connectivity map annotation from a several pages historical pages previously annotated. Once the letter connectivity map is available our framework can synthesis the pictorial representation of any Arabic word or sentence from their text representation. The writing style of the synthesized text resembles the writing style of the input pages. The synthesized words can be used in word-spotting and many other historical document processing applications. The proposed approach provides an intuitive and easy-to-use framework to search for a keyword in the rest of the manuscript. Our experimental study shows that our approach enables accurate results in word spotting algorithms.**

I. Introduction

The ongoing extensive effort to digitize historical manuscripts have produced huge datasets, that shed a light on various aspects of life of past societies. To accesses and process these manuscripts, it is essential to provide a search and retrieve engines. However, since these documents are represented as images and current Optical Character Recognition (OCR) systems perform badly when applied to handwritten historical documents, keyword spotting technique stands as a practical alternative [16]. In keyword spotting, the retrieval is performed on the image domain, and aims to locate regions, in the image, that include patterns similar to the query image keyword.

The Arabic written heritage is mostly handwritten books. A typical book, i.e. a manuscript, includes hundreds of pages that are usually written by similar writing style, often by the same scribe. It is very hard to provide annotation for an entire book, even for a scholar who is studying that specific book. However, it's usually easy and acceptable to provide detailed annotation for several pages. Providing the query keyword, for word spotting or searching, is usually a challenging task since one should submit a pictorial representation of the keyword. Some approaches manually search for an instance of the keyword within the manuscript [33], while others manually assemble the keyword from the word fragments [30], [32].

Arabic script is written from right to left in a semi-cursive manner in handwriting as well as machine printing. A letter in Arabic usually has several shapes, according to its adjacent letters and its position within the word. Some letters interrupt the cursiveness of a word by prohibiting a connection to the following letters and splitting words into connected groups of letters called components or sub-words. We refer to these letters as *interrupting letters*.

In this work, we assume the existence of several annotated pages that provide correspondence between a set of pictorial representation of contentious sub-words and their textual counterpart. In an off-line process we analyze the input pages and generate a data-structure, *Letter Connectivity Map* (LCM), which is used to synthesize a pictorial representation of any input word in Arabic. An LCM includes multiple shapes for each form of the alphabet letters (letters at various locations in a word). Each map represents one writer and encodes a specific handwriting style. The letter connectivity map is used to guide the synthesis of any Arabic sub-word, word, or sentence. The proposed approach provide an intuitive and easy-to-use framework to search for a keyword in the rest of the manuscript. Our experimental study shows that our approach enables much more accurate results in word spotting algorithms.

The application of the proposed framework is beyond keyword searching and word spotting and can be utilized to automatically extend a small collection of annotated Arabic words into a comprehensive annotated database that includes all possible shapes of words and sub-words according to the input writing style.

The rest of the paper is organized as follows. We begin with related work. Then, we explain in detail the proposed framework, beginning with the generation of the letter connectivity map, then detailing the synthesis procedure. Next, we present experimental results and finally, we summarize and draw conclusions.

978-1-5090-1792-8/16 $31.00 © 2016 IEEE
DOI 10.1109/DAS.2016.9

II. Related Work

Several approaches have been developed to recognize isolated forms of Arabic sub-words without segmenting into characters [4], [15], [18]. Character-based recognition approaches store the various forms of each character, while segmentation free approaches are required to maintain large databases that store multiple shapes for each sub-word in the lexicon. These databases are used for training and recognition. The Arabic language has more than $100,000$ different sub-words, which make these databases quite large.

Keyword spotting aims to detect a word in an image and was initially proposed in [11] for printed and handwritten text, respectively. The core of any word spotting procedure is a word-matching algorithm, which measures the distance between pictorial representations of words. Word-matching algorithms roughly fall into two categories: pixel-based and feature-based. Pixel-based matching approaches measure the similarity between the two images on the pixel domain using various metrics, such as Euclidean Distance Map, XOR difference, Scott and Longuet-Higgins distance, Hausdorff distance, or the Sum of Square Differences [14], [22], [26]. Feature-based matching approaches extract features from the images to be compared and measure the similarity on the feature space [1], [13], [31], [34].

Word-Spotting algorithms can also be categorized by whether they need a segmentation step, or not. Some algorithms are segmentation free [7], [8], [25], [27]. While other methods rely on a segmentation step [10], [20].

The Dynamic Time Warping technique was implemented and evaluated using various sets of features [23], [29] and yielded better results than competing techniques [17]. Rath and Manmatha [23] preprocessed segmented word images to create sets of one-dimensional features, which were compared using DTW. Rath and Manmatha [24], describe an approach called word spotting which involves grouping word images into clusters by using image matching to find similarity. They automatically build an index that links "interesting" words to their locations. In order to compute image similarity values, they use and compare a number of different techniques including dynamic time warping. The word similarity values are then used for clustering using both k-means and agglomerative clustering techniques.

Some approaches spot words within lines, thus require the document to be segmented into lines or connected components in a preprocessing step [3], while others work directly on unsegmented pages and treat the spotting task as an image retrieval task [12], [19]. Hidden Markov Models [2], and Neural Networks [28] were used by many researches for keyword searching and spotting tasks.

Recently Saabni and El-Sana [32] presented a system to synthesize a comprehensive database for on-line and off-line Arabic script recognition. Their system can generate multiple shapes for a given word. Sets of prototypes representing the various ways of writing each letter are used to synthesize a word. These sets are extracted manually for each writing style in a tedious, expensive, and time consuming manner, which limits the utilization of the systems. In contrast, our approach relies on the letter connectivity map, which is generated in a fully automatic manner.

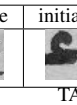
isolate	initial	medial	final
			

TABLE I
Four possible forms for the letter ع from left to right: isolated form, initial form, medial form, and final form.

Fig. 1. Ligature example for the letter ل from left to right: final form, letter itself, ligature

III. Automatic Synthesis

The input to our framework is a small number of document images (pages) and their corresponding annotation. We assume these document images have the same writing style; i.e., written by the same writer. The input images together with their annotations are processed to generate a data structure, which is used with a script model, as seen in Table II, to generate any Arabic word.

The presented framework consists of two main procedures: building the *Letter Connectivity Map* (LCM) and synthesizing Arabic words. The first procedure automatically segments the images, extracts the pictorial representation of the letters in each sub-word, and populates the LCM. The second procedure synthesizes a pictorial representation of an input text according to a given LCM. It fetches letters in the appropriate forms and aligns them to generate the pictorial representation of the input text according to the LCM writing style. Next, we describe the two procedures in detail.

A. Letter Connectivity Map

A typical letter in Arabic has up to four different forms, classified into isolated and contextual. A Contextual form may be connected at the beginning of the sub-word, at the middle, or at its end. Due to this characteristic, the letters are split into four connectivity groups: (i) isolate, (ii) initial, (iii) medial, and (iv) final. *The Script Model* defines the connectivity, and the possible shapes of each of the alphabet letters in the Arabic script as presented in Table II. This model, which is generated once per language, may differ a bit for other languages, such as Urdu and Persian.

Let us define the form of a letter, l, as $form(l)$, which can have four values: isolate, initial, medial, and final. Let us also define the isolate, initial, medial, and final shape of a letter l as $isolate(l)$, $initial(l)$, $medial(l)$, and $final(l)$, respectively. We define the letter represented by a given shape, s, as $letter(s)$, where s could be in any of the four forms. An example of the four different forms of a letter can be seen in Figure I

The *Letter Connectivity Map* (LCM) has an entry for each letter of the 28 letters in the language. Each entry includes

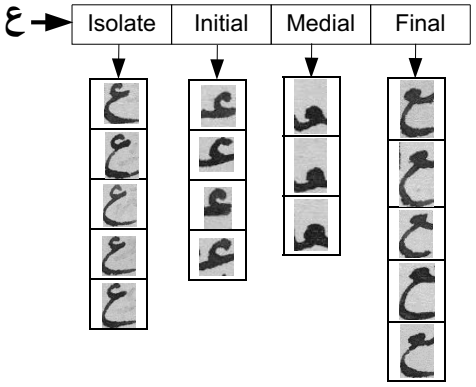


Fig. 2. An example for an LCM entry: the shapes in the lists of the entry of the letter ع.

Letter	Initial	Medial	Final	Letter	Initial	Medial	Final
ا	N/A	No	Yes	ض	No	No	Yes
ب	No	No	Yes	ط	Yes	Yes	Yes
ت	No	No	Yes	ظ	Yes	Yes	Yes
ث	No	No	Yes	ع	No	No	No
ج	No	No	Yes	غ	No	No	No
ح	No	No	Yes	ف	No	No	Yes
خ	No	No	Yes	ق	No	No	Yes
د	N/A	N/A	Yes	ك	No	No	Yes
ذ	N/A	N/A	Yes	ل	No	No	Yes
ر	N/A	N/A	Yes	م	No	No	Yes
ز	N/A	N/A	Yes	ن	No	No	Yes
س	No	No	Yes	ه	No	No	No
ش	No	No	Yes	و	N/A	N/A	Yes
ص	No	No	Yes	ي	No	No	Yes

TABLE II
THE ARABIC SCRIPT MODEL. EACH LETTER MIGHT LOOK THE SAME WHEN IT IS ISOLATED, AND WHEN IT IS IN A CONTEXTUAL FORM. N/A IS WHERE SUCH FORM IS NOT POSSIBLE IN THE LANGUAGE.

pointers to four lists, representing the four forms (isolated, initial, medial, and final) of a letter, and one scalar (which is only relevant for the interrupting letters). Each list stores various instances of $isolated(l)$, $initial(l)$, $medial(l)$, and $final(l)$, respectively. Figure 2 illustrates one entry of the LCM. The map is populated from the given annotated pages, and used to guide the generation of the continuous sub-words. Populating the isolated forms of each letter in the map is done in a straightforward manner, as sub-words consisting of one letter do not require extraction. They are detected through their corresponding text annotation and added to the appropriate entries in the letter connectivity map.

The initial form, $initial(l)$, of some Arabic letters resembles its isolate form, $isolated(l)$; i.e., $initial(l) = isolated(l)$ plus a connecting ligature, see Figure 1. Similarly the final form of some letters resemble the isolate form. The *Script Model* stores the information to determine these letters. We take advantage of this property and assign the isolate shapes also to the initial or the final list of these letters. To complete the LCM, we traverse the sub-word on an increasing order of their length, i.e. by their number of letters, and utilize sub-words that include letter shapes already in the map. For example, sub-words of size two that include initial shapes already in the LCM are used to determine the final shape of letters, which are not in the LCM. The discovered shapes are added to the LCM, populating the LCM in an incremental manner. Next we explain this process in detail.

A typical sub-word ω of length n is represented as $s_0 s_1 s_2 .., s_{n-1}$, where s_i is a shape of a letter. Let us denote the set of continuous sub-words that consist of n letters by W^n, e.g. W^1 is the set of the isolate letters.

We traverse the sub-words in an incremental manner, the set W^1 is used to incorporate the isolate letters, and the initial and final shapes that resemble the isolate forms. Next we proceed to process the set W^2, sub-words of two-letters. For each sub-word $\omega_i = s_0 s_1 \in W^2$, we consult with the LCM to determine whether at least one of the shapes already in the LCM. We used the stored shapes to split the sub-word to two shapes, as illustrated in Table V, and store them in the appropriate entries – s_0 will be added to the initial list of the entry $letter(s_0)$ and s_1 will be added to the final list of the entry $letter(s_1)$. If none of the two shapes is in the LCM, we delay processing this sub-word to a later stage, until at least one of the shape enter the LCM.

Once the W^2 set is processed, the LCM will hold a nonempty lists for the initial and finals forms of each letter, at the best. In any case, the medial form lists for each letter in the map will be empty, due to the nature of the Arabic language. To complete this group, we continue extracting letters from three-letter sub-words. For each sub-word $\omega_i = s_0 s_1 s_2 \in W^3$, we query the existence of s_0 in the initial list of $letter(s_0)$ as well as s_2 in the final list of the entry $letter(s_2)$ in the LCM (this is easy to perform as we have the textual representation of the sub-word, from the annotated pages). If both shapes exist in the LCM, we subtract them from ω_i, add s_0 and s_2 to the appropriate lists, and store s_1 to the medial list of the entry $letter(s_1)$, as illustrated in Table IV. If at least one of the shapes s_0 or s_2 does not exist in LCM, we delay processing this sub-word. Processing sub-words that consist of more than three letters is done in a similar manner; i.e., they contain one initial-form letter, one final-form letter, and two or more medial-form letters.

B. Shape Extraction

In this section we discuss the extraction of a letter shape from a sub-word image given the textual representation of this sub-word. Let $\omega = s_1..s_{n-1} \in W^n$ and $u = l_0...l_n$ be a sub-word image and its textual representation, respectively. The shape extraction procedure determines the shape image s_i that corresponds to the letter l_i. Since ω is continuous sub-word image, the main challenge is to determine the location and the boundary of the image of each shape. To extract the shapes from a sub-word image, we iteratively subtract instance shapes of the extremes (head or tail) letters from ω, until we obtain a residual ω_r consisting of one letter. In each iteration an extreme letter, l_i, of the textual representation ω that have a corresponding shape, ρ_i, in the LCM is selected, matched

sub-word	Right	Left	sub-word	Right	Left
					
					

TABLE III

TWO-LETTER SUB-WORDS, THEIR EXTRACTED LETTERS, AND THE DETECTED ANCHOR REGIONS. WHERE 'RIGHT' FOR RIGHT ANCHOR, AND 'LEFT' FOR LEFT ANCHOR.

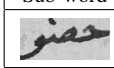
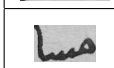
Sub-word	R	R&L	L	Sub-word	R	R&L	L
							
							

TABLE IV

THREE-LETTER SUB-WORDS, THEIR EXTRACTED LETTERS, AND THE DETECTED ANCHOR REGIONS. WHERE 'R' DENOTES RIGHT ANCHOR, AND 'L' DENOTES LEFT ANCHOR.

with s_i, and subtracted from ω. The newly extracted shape instance, s_i, is added to LCM and the residual ω is used for the next iteration.

To extract the shape s_i from ω, we define the *template list*, t_i, as the list $form(s_i)$ of the entry $letter(s_i)$. We search for an instance of each member of t_i in ω, using template matching [5], to determine the position of s_i. This produces a list of coordinates, one coordinate per match. To find the best match, we apply k-means to cluster the coordinates, and choose the largest cluster. Then, we choose the coordinate closest to the centroid of the group. This coordinate and the width of its corresponding template image are used to extract the letter from the sub-word.

To be able to synthesize sub-words of the different letter images, we need to detect the *anchor regions*, which determine the alignment of the forms of adjacent letters in a sub-word. In Arabic, the anchor regions of initial and final forms are in the left and right sides of the letter image, respectively. The medial form has two anchor regions, on in each side of the letter image. For two adjacent letters, l_j and l_{j+1}, in ω, the anchor regions are found in the left side for l_j, and in the right side of l_{j+1}. We scan the first and the last columns of the extracted letter l_j and l_{j+1}, respectively, and detect a ink region (text fragments), shared between the two letters. We add this region to the candidate anchor region list.

Usually one candidate is detected for two adjacent letters. In some cases, due to the presence of diacritics, we might detect more than one candidate. In such a case, we chose the anchor region which has the largest ink size, and mark it as the anchor region. Once the detection of anchor regions for the extracted letters is complete, we add the extracted letters with valid anchor regions to the letter connectivity map.

The final step is to order the extracted letter images in each entry in a decreasing order of their confidence. We adopt the Radial Descriptor [9] to measure the distance between the letter images in each entry, and generate a distance matrix D, where D_{ij} is the distance between the images i and j. We define the confidence of image i as the sum of its corresponding column i in the matrix. Using the confidence rating, the best candidates are in the top of the list and the worst ones are in the bottom.

Examples of anchor detection can be seen in Table III and Table IV. Extraction examples can be seen in Table V and in Table VI.

C. Synthesis

The synthesis procedure generates pictorial representation of an input text. The procedure retrieves the appropriate letters

Sub-word	Final	Initial	Sub-word	Final	Initial
					
					

TABLE V

SEGMENTATION RESULTS SAMPLE OF TWO-LETTER SUB-WORDS. FOR EACH SUB-WORD, WE DETECTED THE FINAL LETTER, AND THE REMAINDER LETTER IS ASSIGNED AS AN INITIAL LETTER.

from the letter connectivity map combining them into sub-words to generate words.

1) sub-word: sub-word synthesis aims to generate a pictorial representation of a given sub-word text. For a given sub-word text $\tau = l_0 l_1 ... l_n$, we retrieve the top candidate for each l_i from the appropriate entry; i.e., l_0 will be retrieved from the initial entry, l_n from the final entry, and $l_1, l_2, ..., l_{n-1}$ from the medial entries of the corresponding letters.

We define the stroke distance δ_{ij} between two consecutive letters i and j as the difference between the heights of the shared anchor regions. The best synthesis will minimize the sum of δ_{ij} between the adjacent letters as formulated in Eq. 1. We select the set of letter images that minimizes this sum.

To generate the pictorial representation of the sub-word τ, we place the selected images according to the order of the corresponding letters in the textual representation. We align each two adjacent letters such that their anchor regions have the maximal overlap possible.

$$\Delta = \sum_{i=0}^{n-1} \delta_{i,i+1} \tag{1}$$

2) Word: Recall that an Arabic word is a sequence of sub-words. For a given textual representation of a word, it is easy to determine its sub-words. We generate a pictorial image for each sub-word, and align them to generate the pictorial representation of the word. The alignment process aims to

Sub-word	Final	Medial	Initial	Sub-word	Final	Medial	Initial

TABLE VI

SEGMENTATION RESULTS SAMPLE OF THREE-LETTER SUB-WORDS. FOR EACH SUB-WORD, WE DETECTED THE FINAL LETTER AS WELL AS THE INITIAL LETTER, AND THE REMAINDER LETTER IS ASSIGNED AS A MEDIAL LETTER.

align the sub-words horizontally as well as vertically. Vertical alignment places the sub-words in the appropriate order along the baseline, and horizontal alignment ensures proper spacing between adjacent sub-words.

The synthesized sub-words are already in the appropriate orientation. We only need to place them on the baseline such that the mean squared distance between the anchor points of the sub-word and the baseline is minimized.

In Arabic script, six letters interrupt the continuity of a word, and form multiple sub-words. In our framework, we assume the existence of several annotated images. We utilize these input to learn the distance between adjacent sub-words for each interrupting letter. We calculate the average distance for each one and store it in the LCM. For each two adjacent sub-words, we determine the interrupting letter and place them apart according to the distance stored in the LCM.

3) Sentence: To synthesize a sentence, we synthesize its constituting words, and place them in the appropriate order on a defined baseline. Similar to sub-word alignment we place each word on the baseline while minimizing the mean squared distance from the anchor points of this word. We learn the spacing between adjacent words from the annotated input images while considering the ending letter of the first word and the beginning letter of its following word, and store these information in the letter connectivity map. These values are used to determine the horizontal space between two adjacent words in the sentence.

The synthesized sentence does not occupy the whole image, and black areas may be present. We generate a synthetic background image from the original dataset, and randomly select a pixel from the background image to fill these black pixels.

IV. Experimental Results

We have implemented this framework in C++ using OpenCV [21], and applied it on various datasets collected from Harvard's Open Collections Program [6]. In the paper we report the results on three datasets, each consists of ten annotated pages written by the same writer, in Arabic language.

We processed each of the three datasets, independently, to generate three letter connectivity maps, which were used to synthesize texts of various content and length. Each dataset contributes 3500 sub-words on average. For a give LCM, the synthesis of an input text is done in real time. Let us refer to these sub-words as *original* sub-words.

To evaluate our framework, we need to assess the quality of the synthesized sub-words. For each writer, we randomly selected n sub-words and for each sub-word we randomly chose $2k$ images, half of them are original and the other half are synthetically generated by our system. Samples of the original and synthetic sub-words used in our tests can be seen in Table VII

We applied the Radial Descriptor [9] to generate a occurrence histogram for the patterns (images), and use these histograms to calculate the distance between any two patterns. We compute the inter variance between the synthetic and original pictorial representation of each sub-word and the intra variance among the original instances of each sub-word. For inter-variance, we calculated the pairwise distance between the synthetic pictorial representations and the original ones, for each sub-word. The intra variance is the pairwise distance among the original instances of the same sub-word. We calculate the average distance, as well as the variance of these distances. The distance values are normalized between 0, where the images are identical, and 1, for the maximum distance found in the set.

Figure 3 reports the results of the experiments for three different writers, where $n = 8$ sub-words and $k = 20$ images for each sub-word. As can be seen, for the majority of the sub-words, the synthesized pictorial representation are as good as the original sub-word images. The variance for the tested sub-word remains very low. These results indicated that using our approach for word-spotting will result in just as accurate results as manually selecting a specific instance of a sub-word. Our system provides a fast, robust, and easy to use method, where users only needs to input the requested text, and thus it saves the user the hassle of the manually for a pictorial instance of the query text.

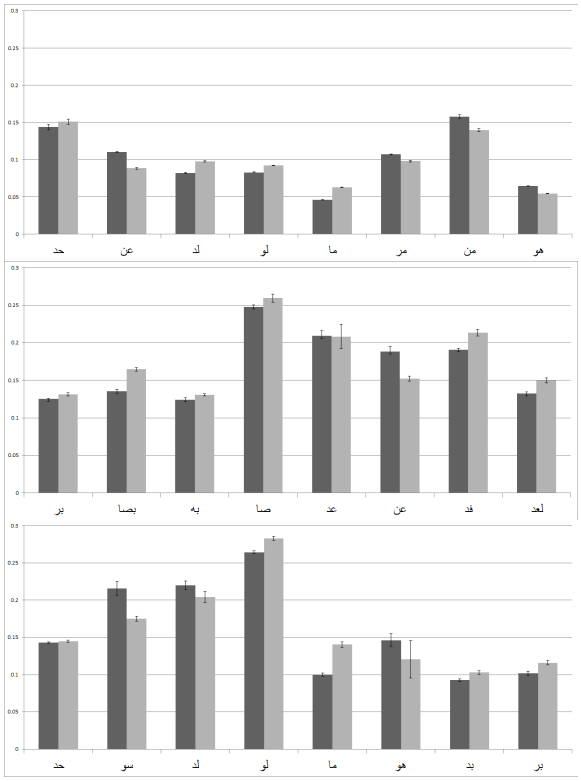


Fig. 3. Comparison Results for the three different writing styles, For each writer, we chose 8 different sub-words. In each chart, the left bar for each sub-word is the average distance and variance for the original pictorial images, while the right bar for each sub-word is the average distance and variance for the synthetic pictorial images. Values are normalized to the range of [0,1], where 0 is identical, and 1 is very distant. Please note that the charts have been cropped to [0,0.3] to save space.

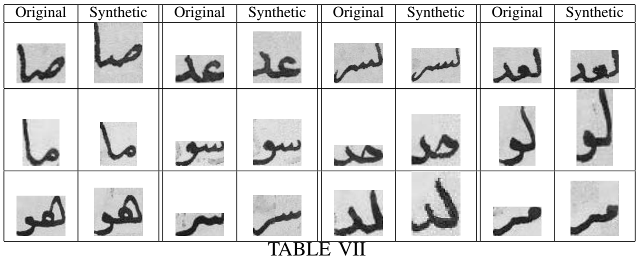


TABLE VII

Sample sub-word used in our tests. Left column is original image, and to its right is a synthetic image generated using our system.

V. Conclusions, and Future Work

We presented a novel framework for automatic and efficient synthesis of historical handwritten Arabic text. The main purpose of this framework is to assist word spotting and keyword searching in handwritten historical documents.

We have showed in our experimental results that the synthesized sub-words are on par in quality and variance with the original sub-words, which deems them applicable for word-spotting algorithms. The benefit from having the ability to execute word-spotting without the need to manually find the pictorial image of a sub-word and cut it for search is a big benefit to word-spotting.

References

[1] D. J. A. Bhardwaj and V. Govindaraju. Script independent word spotting in multilingual documents. In *in 2nd Int'l Workshop on Cross Lingual Information Access*, pages 48–54., 2008.

[2] V. F. A. Fischer, A. Keller and H. Bunke. Hmm-based word spotting in handwritten documents using subword models. In *in 20th Int'l Conf. on Pattern Recognition*, pages 3416–3419, 2010.

[3] M. A. Aleksander Kolcz, Joshua Alspector. A line-oriented approach to word spotting in handwritten documents. *Pattern Analysis and Applications*, 3(2):153 – 168, 2000.

[4] F. Biadsy, R. Saabni, and J. El-Sana. Segmentation-free online arabic handwriting recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 25(07):1009–1033, 2011.

[5] K. Briechle and U. D. Hanebeck. Template matching using fast normalized cross correlation. In *Aerospace/Defense Sensing, Simulation, and Controls*, pages 95–102. Int. Society for Optics and Photonics, 2001.

[6] J. CETIS. Open educational resources–opportunities and challenges for higher education. 2008.

[7] B. Gatos, T. Konidaris, K. Ntzios, I. Pratikakis, and S. J. Perantonis. A segmentation-free approach for keyword search in historical typewritten documents. In *International Conference on Document Analysis and Recognition*, pages 54–58. IEEE, 2005.

[8] B. Gatos and I. Pratikakis. Segmentation-free word spotting in historical printed documents. In *International Conference on Document Analysis and Recognition*, pages 271–275. IEEE, 2009.

[9] M. Kassis and J. El-Sana. Word spotting using radial descriptor. In *International Conference on Frontiers in Handwriting Recognition*, pages 387–392, 2014.

[10] T. Konidaris, B. Gatos, K. Ntzios, I. Pratikakis, S. Theodoridis, and S. J. Perantonis. Keyword-guided word spotting in historical printed documents using synthetic data and user feedback. *International Journal of Document Analysis and Recognition (IJDAR)*, 9(2-4):167–177, 2007.

[11] S. S. Kuo and O. E. Agazzi. Keyword spotting in poorly printed documents using pseudo 2-d hidden markov models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(8):842–848, 1994.

[12] Y. Leydier, F. Le Bourgeois, and H. Emptoz. Omnilingual segmentation-free word spotting for ancient manuscripts indexation. In *Proceedings. Eighth International Conference on Document Analysis and Recognition*, pages 533–537Vol.1, 29 Aug.-1 Sept. 2005.

[13] Y. Leydier, F. Lebourgeois, and H. Emptoz. Text search for medieval manuscript images. *Pattern Recognition.*, 40(12):3552–3567, 2007.

[14] Y. Lu and C. L. Tan. Word spotting in chinese document images without layout analysis. In *International Conference on Pattern Recognition*, volume 3, pages 57–60, 2002.

[15] S. S. Maddouri and H. Amiri. Combination of local and global vision modelling for arabic handwritten words recognition. In *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, pages 128–135. IEEE, 2002.

[16] R. Manmatha and W. Croft. Word spotting: Indexing handwritten archives. *Intelligent Multimedia Information Retrieval Collection*, pages 43–64, 1997.

[17] R. Manmatha and T. Rath. Indexing handwritten historical documents - recent progress. 2003.

[18] N. Mezghani, A. Mitiche, and M. Cheriet. Bayes classification of online arabic characters by gibbs modeling of class conditional densities. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(7):1121–1131, 2008.

[19] R. F. Moghaddam and M. Cheriet. Application of multi-level classifiers and clustering for automatic word spotting in historical document images. In *ICDAR*, pages 511–515, 2009.

[20] N. Nikolaou, M. Makridis, B. Gatos, N. Stamatopoulos, and N. Papamarkos. Segmentation of historical machine-printed documents using adaptive run length smoothing and skeleton segmentation paths. *Image and Vision Computing*, 28(4):590–604, 2010.

[21] L. OpenCV. Computer vision with the opencv library. *GaryBradski & Adrian Kaebler-O'Reilly*, 2008.

[22] T. Rath, S. Kane, A. L. and. Partridge, and R. Manmatha. Indexing for a digital library of george washingtons manuscripts: A study of word matching techniques. *CIIR Technical Report, University of Massachusetts Amherst.*, 2002.

[23] T. Rath and R. Manmatha. Word image matching using dynamic time warping. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings*, volume 2, pages II–521–II–527vol.2, 18-20 June 2003.

[24] T. M. Rath and R. Manmatha. Word spotting for historical documents. *International Journal of Document Analysis and Recognition (IJDAR)*, 9(2-4):139–152, 2007.

[25] L. Rothacker, M. Rusinol, G. Fink, et al. Bag-of-features hmms for segmentation-free word spotting in handwritten documents. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 1305–1309. IEEE, 2013.

[26] J. L. Rothfeder, S. Feng, and T. M. Rath. Using corner feature correspondences to rank word images by similarity. *Computer Vision and Pattern Recognition Workshop*, 3:30, 2003.

[27] M. Rusinol, D. Aldavert, R. Toledo, and J. Lladós. Browsing heterogeneous document collections by a segmentation-free word spotting method. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 63–67. IEEE, 2011.

[28] J. S. S. Fernandez, A. Graves. An application of recurrent neural networks to discriminative keyword spotting. In *in 17th Int'l Conf. on Artificial Neural Networks, ser. Lecture Notes in Computer Science*, volume 4669, pages 220–229, 2007.

[29] C. H. S. N. Srihari, H. Srinivasan and S. Shetty. Spotting words in latin, devanagari and arabic scripts. *Indian Jounal of Artificial Intelligence*, 16(3):2–9, 2006.

[30] R. Saabni and J. El-Sana. Keyword searching for arabic handwritten documents. pages 716–722, 2008.

[31] R. Saabni and J. El-Sana. Word spotting for handwritten documents using chamfer distance and dynamic time warping. In *Document Recognition and Retrieval XVIII*, 2011.

[32] R. Saabni and J. El-Sana. Comprehensive synthetic arabic database for on/off-line script recognition research. *International Journal on Document Analysis and Recognition (IJDAR)*, 16(3):285–294, 2013.

[33] S. Srihari, H. Srinivasan, P. Babu, and C. Bhole. Handwritten arabic word spotting using the cedarabic document analysis system. In *Proc. Symposium on Document Image Understanding Technology (SDIUT-05)*, pages 123–132, 2005.

[34] A. F. S. T. Adamek, N. E. Connor. Word matching using single closed contours for indexing historical documents. *Journal on Document Analysis and Recognition*, 9(2):153–165, 2007.