AN EFFECTIVE APPROACH TO OFFLINE ARABIC HANDWRITING RECOGNITION

Jafaar Alabodi and Xue Li

School of Information Technology and Electrical Engineering, University of Queensland, Brisbane, Qld 4072, Australia

ABSTRACT

Segmentation is the most challenging part of the Arabic handwriting recognition, due to the unique characteristics of Arabic writing that allows the same shape to denote different characters. In this paper, an off-line Arabic handwriting recognition system is proposed. The processing details are presented in three main stages. Firstly, the image is skeletonized to one pixel thin. Secondly, transfer each diagonally connected foreground pixel to the closest horizontal or vertical line. Finally, these orthogonal lines are coded as vectors of unique integer numbers; each vector represents one letter of the word. In order to evaluate the proposed techniques, the system has been tested on the IFN/ENIT database, and the experimental results show that our method is superior to those methods currently available.

KEYWORDS

Offline character recognition; segmentation; skeletonization; binarization

1. INTRODUCTION

The Arabic world has thousands of years of recorded literatures. Their digitization is a pending challenge to information system developers. On the other hand, the writing remains a preferred method for the vast majority to express ideas and to exchange information. New tools are being invented to facilitate integration between traditional writing and digital documents. Those tools include digital pens, digital panels, personal digital assistant PDA's, computer hardware and mobile phones. All these tools rely on touch-sensitive screens and special pens to allow users to hand-write their text as an input method.

To recognize handwritten text, most approaches go through a few stages as a process. These stages start from the handwriting on a paper to the last stage which displays the result text on the screen or saves it as a text file (see Figure. 1).

The accuracy of recognition can be greatly affected by the quality of input text. Pre-processing is used to filter out noise and transform the image into an intermediate representation with features preserved.

Pre-processing has two major stages, binarization and skeletonization. Binarization converts images from greyscale into binary images. Skeletonization is a process that reduces the width of a pattern shape to just a single pixel. The objective of skeletonization is to find the median axis of a character. The median axis is defined as a smooth curve or set of curves that follows the shape of a character equidistantly from its contours [1]. Thinning is the process of transforming a pattern

DOI : 10.5121/ijaia.2013.4601

from one form to another with less thickness while maintaining the connectivity of the original pattern [2].

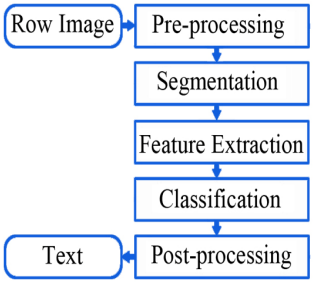


Figure. 1. Stages of character recognition process.

The purposes of skeletonization of a binary image are:

- To reduce the amount of data that needs to be processed.
- To reduce overall processing time.
- To extract features such as junction-points, end points, and connectivity between the components for the further processing needs.

The next problem in recognizing cursive handwriting is the segmentation of words into characters [3]. Segmentation splits the text areas into pieces of the image in order to recognize the characters or words. Many systems do not segment a text image into characters or recognizable isolated characters [4]. Other approaches recognize sub-words [5]. The problems with this segmentation technique include misplaced segmentation, over segmentation, or under segmentation (as illustrated in Figure. 2). In this paper, in order to avoid segmentation problems, we propose a new segmentation technique, which is independent of handwriting style and size, and is based on the geometrical features of Arabic handwriting characters. Experimental results show that our proposed method performs with higher accuracy than existing methods.

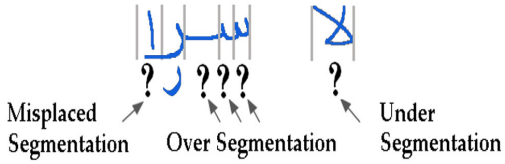


Figure. 2. Common problems in character segmentation.

The main idea of this paper is to preserve the cursive writing properties, so we design a new coding scheme to record the geometrical features of pixels. Analogue to the idea of using codons in the genetic code for DNA sequencing of the human genome [6], our coding scheme is used to represent the geometric information on how pixels are connected to each other. The connections between pixels are denoted by integer vectors. Thus, the pattern recognition problem becomes a vector sequencing problem.

This paper is organized as follows. Section 2 presents the overview of related work in Arabic handwriting recognition. Section 3 describes our proposed algorithms and operations for a complete system. Section 4 presents the experimental results and evaluations. Section 5 presents the conclusions of this research.

2. Related Work

There are a few well known approaches used in offline Arabic handwriting recognition [7]. In the following discussion, we focus on the approaches used in the life cycle of the offline Arabic handwriting recognition process. The weaknesses of current approaches are discussed.

2.1. Pre-processing

In most binarization algorithms, a threshold is used. If a pixel has a smaller value than that threshold, it is considered as background; otherwise it is considered as writing. Threshold methods are divided into two categories: global and local thresholds. In the global thresholding method, a single threshold value is computed and is applied to all pixels in the image. In local thresholding, different values of different threshold values for different local areas are applied.
Otsu's method [8] is widely used in binarization. This method selects a threshold based on the minimization of the within-group variance of the two groups of pixels separated as a result of a global threshold. This method can be used to clean a document image that has a simple background. Although this method is not designed especially for handwriting image binarization, the Otsu's method does not need any user defined parameter. This contrasts with other algorithms that require prior knowledge about the number of peaks in the histogram. Figure. 3 illustrates an example resulted from the Otsu's method.

$$p(i) = \frac{(number\ of\ pixels\ with\ gray\ level\ value\ of\ i)}{(total\ number\ of\ pixels)} \quad (1)$$

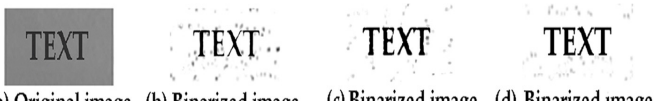
(a) Original image (b) Binarized image

Figure. 3. Otsu's algorithm applied to an image.

A global threshold technique may not be sufficient for binarizing handwriting images with complex backgrounds. In such cases, local methods are more useful. Niblack's local average method [9] is frequently used to decide the local thresholds. The formula is given in (2).

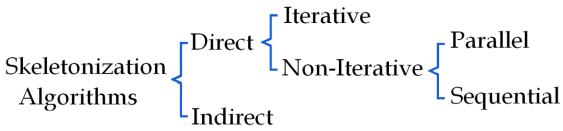
$$T(x,y) = M(x,y) + k.\sqrt{V(x,y)} \quad (2)$$

Where M(x, y) is the local mean, k is a user-defined parameter to determine how much of the total print object boundary is taken as a part of the given object, V(x, y) is the local variance. V(x, y) is computed in a moving (w x w) sliding window, w is the size of the sliding window. As shown in Figure. 4 (b) the resulting binary image for default values of w and k parameters are not useful because of noisy characters. The recommended value of w is 15 and a typical value for k is -1. These parameters are image-dependent; generally small values of w lead to noisy results and inconsistent stroke width and large values cause some characters to merge or split.



(a) Original image (b) Binarized image with k =-1 and w = 15 (c) Binarized image with k =-1 and w = 30 (d) Binarized image with k =-0.5 and w = 30

The skeletonization techniques can be divided into two major categories [10], namely directly and indirectly removing pixels from a given image pattern. Direct reduction methods can be either iterative or non-repetitive. The iterative direct techniques compute skeletons by iteratively deleting removable boundary pixels either sequentially [11] or in parallel [12] (see Figure. 5), until there are no further changes to the image.



2.2. Segmentation

It has been noted that segmentation can be performed before or after pre-processing. The segmentation approaches can be categorized into two classes [13]:

2.2.1. Holistic strategies

There is no need to segment words into individual characters because the recognition is performed word by word. In this way, we need to segment a line of text into words, which is difficult in Arabic language because the space between letters within a word may even be greater than the space between the words.

2.2.2. Analytical strategies

In this approach, words are segmented either implicitly as part of a word classifier or explicitly as a lexicon filter. The explicit strategy tries to isolate single letters, which are then individually recognized [14]. While in implicit segmentation, the recognition is performed at an intermediate level, not at the word or character level. The algorithm is usually based on Hidden Markov Model (HMM) because the text image (word text image) is converted into a sequence of small size units (a sequence of observations). Each unit may be a part of a letter. A number of successive units can belong to a single letter.

To the best of our knowledge, we have not seen any algorithm that can segment handwritten Arabic words into characters with a high level of accuracy. Therefore, most well-known handwriting recognition systems have used implicit segmentation [15]. After explicit segmentation, each character can be recognized by a classifier such as HMM or Artificial Neural Networks ANN.

3. Proposed Algorithms

Our proposed algorithms are described according to the offline handwriting recognition process described in Figure. 1.

3.1. Binarization

The main idea behind the algorithm is to distinguish between the colour of the background of the image, and the colour of the text in the image. The threshold selected using the average of pixel values taken from the four corners of the image as well as from the centre of the image. The following criteria are used to determine the threshold:

- The binarized resulting image should not have any noise in non-text areas.

- The text is scanned using a scanner with 300 dpi resolution.

The focus of this algorithm is not to lose any part of the text pixels regardless of the level of convergence between the text colour and the background colour. The proposed algorithm performs the following steps:

Calculate the threshold T1 for all pixels in every corner of the image.

$$T1 = \sum_{c=0}^{w} \sum_{r=0}^{h} (c, r) / (w \times h) \quad (3)$$

Where w is the number of the pixels in the width of the image divided by 10, (to avoid the central pixels). And h is the number of pixels in the height of the image divided by 10. Choose the minimum threshold value.

Calculate the threshold T2 for all pixels in the centre of the image. That is because the number of background pixels is usually much larger than the pixels on the foreground and the mean value of all pixels will be always closer to the background pixel values. Then the algorithm compares T2 to T1 to select the minimum value T3. For all pixels in the image, assign a white colour to all pixels with a grey level value higher than T3, and black colour of all pixels with a grey level value lower than T3 (Figure. 6).

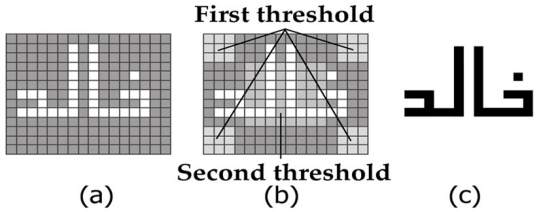


Figure. 6. (a) Original image. (b) Obtaining the thresholds. (c) Binarized image.

3.2. Skeletonization and Thinning

The input image of this algorithm is a binary image. The algorithm will extract skeletons from an image that consists of single pixels of the skeleton and remove all contour pixels of the image. The thinning algorithm is listed as Algorithm 1.

In this algorithm, we denote the spatial relationship between pixels by assigning an integer number to each pixel. The idea is that for any given pixel, there are eight directions for eight possible adjacent pixels. So the geometric property of any pixel can be considered within a 3 x 3 matrix of pixels. Subsequently, relationships of all pixels can be recorded by using a 3 × 3 sliding window. This window is moving through every pixel in the image at one pixel per step to determine the number of black pixels from its surrounding 8-neighbor pixels. As it is shown in Figure 7-a, each neighbour of a given pixel is assigned by a special number. The property of these 8 integer numbers is that each one of them can uniquely define a certain neighbouring pixel. There are 256 unique patterns of surrounding pixels as calculated in Formula 4.

$$\sum_{i=1}^{8} C_{8}^{i} = 2^{8} = 256 \quad (4)$$

This indicates that there are 256 unique states of all possible geometric combinations of the surrounding pixels for any given pixel (see Figure. 7-a).

We use the following example to illustrate how a unique combination is calculated (See Figure. 7-b). When we have a number 145, then the pixels are at above, below, and top-left, as 145 = 1 (above) + 16 (below) + 128 (top-left).

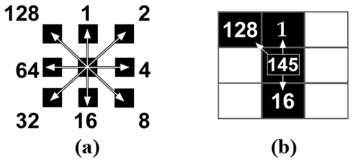


Figure. 7. Coding scheme for neighbouring pixels.

Algorithm 1 Skeletonization.

Input:

A binary image I [r, c] is the binary input image having R rows and C columns, and the image background is represented by zeros.

Output:

A skeleton image of I image.

1: **For** each pixel from left to right
2: **For** each pixel from up to down
3: If b [r, c] is a black pixel
4: **Calculate** a value of b depending on the number and the position of the 8 black pixels surrounded it.
5: **Determine** whether b is located at the external or the internal boundary of the handwriting depending on its value which obtained from the previous step, then if so, change b to the grey colour.
6: **End for;**
7: **End for;**
8: **For** each pixel from left to right
9: **For** each pixel from up to down
10: If b [r, c] is a grey pixel
11: **Remove** b [r, c] by changing it to white.
12: **End for;**
13: **End for;**
14: **Repeat** 1 to 13 until no removable pixels are left.
15: **Return** a skeleton image.

As shown in the Figure. 8, they are examples of relationships between black pixels.

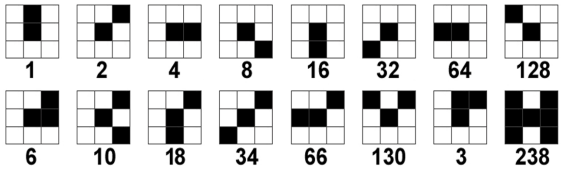


Figure. 8. Different cases of black pixels value example.

In the algorithm, processing pixels starts vertically from the top-left corner of the image. If the black pixel located at the border of the handwriting, the algorithm will change the colour of this pixel to grey. Afterwards all grey pixels are deleted. As long as there is a removable pixel, the algorithm will repeat the steps. Figure. 9 shows a practical example and illustrates how to use this algorithm.

Figure. 9. Skeletonization of "H" letter.

The algorithm considers two pre-prepared lists (see Figure. 10 and Figure. 11). Two pixels are connected if and only if they are adjacent to each other through one of 8 directions in the image. The removability is considered for the central pixels listed in Figures 10 -11. Figure. 10 contains the geometry values of removable pixels, removing a central pixel would not cause the loss of connectivity, while Figure. 11 contains the geometry values of essential pixels for preserving the skeleton, removing any central pixel would result in two separate sets of pixels.

When evaluating the effectiveness of skeletonization algorithm, we consider the connectivity preservation and the width of the skeleton. We also consider the geometric information preservation. Compared with existing algorithms, we found that our algorithm is robust in reducing the noise.

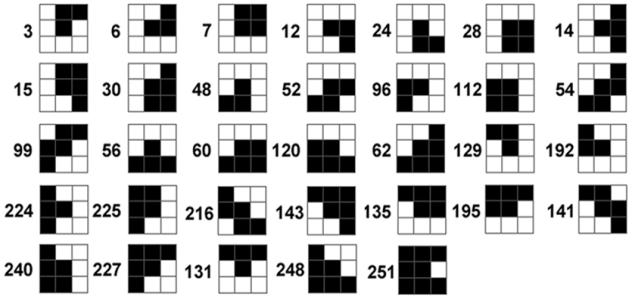


Figure. 10. Removable pixels.

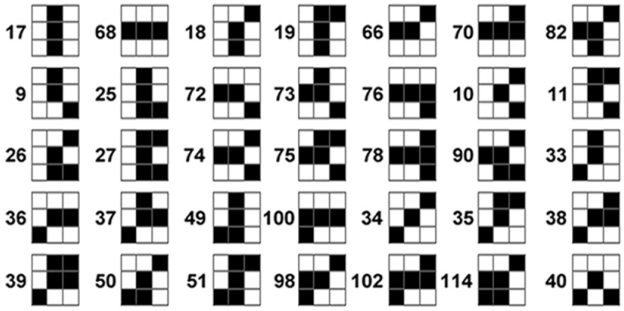


Figure. 11. Non-removable pixels.

3.3. Skew Correction

To correct image skew, we need to detect the baseline which is an imaginary horizontal line in which all words in one line are written on it. Most letters in the Arabic language are connected with other letters depending on their position within the word.

Algorithm 2 Baseline Detection.

Input:
A skeleton image S for an Arabic handwriting text.

Output:
Angle, the position of correct baseline

1: **Rotate** the image to the right 1 degree then calculate the maximum value (peak) of horizontal histogram.
2: **Rotate** the image to the left 1 degree then calculate the maximum value (peak) of horizontal histogram.
3: **Compare** the results of the previous two steps; lead us in the correct baseline direction.
4: **Repeat**:
5: **Rotate** image 1 degree to the direction that found in step3.
6: **Find** the max value in the horizontal histogram.
7: **If** the new max value > last max value then
8: **Let** last max value = new max value.
9: **Save** the angle value.
10: **End IF**
11: **Add** 1 to a counter.
12: **While** the counter < threshold value.
13: **Return** the angle and the max value of the horizontal histogram for that angle.

The horizontal projection method is used to find the number of all black pixels of every row in the image.

The horizontal projection is defined as:

$$hp(i) = \sum pixel(i, j) \qquad (5)$$

Where hp(i) is the horizontal projection of the image for row i, and the pixel (i , j) is a black pixel at row i, column j.

3.4. Segmentation

After having completed the process of the skew correction, we apply the process of horizontal projection to the whole image. This process determines the line space that is used to divide the image horizontally, see Figure. 12. The next step is to perform a vertical projection operation on the image of each line to diagnose the spaces between the words within the line, see Figure. 13.

Figure. 12. The horizontal projection.

Figure. 13. The vertical projection.

3.5. Word Recognition

To recognize letters from input handwritten image, we firstly convert the handwriting images into a canonical representation which has only horizontal and vertical pixels. By doing so, both efficiency and effectiveness of recognition can be achieved (Figure. 14 and Figure. 15).

To convert diagonal pixels into horizontal and vertical pixels we change the diagonal pixel of the given pixels in the same row (horizontal) and the same column (vertical).

We examine pixels from the top left of the image to the bottom right in a column-first fashion. In this case, the value of the first black pixel in the location (i.e., column 1, row 9) is 18. Because the pixel that is on the top right is not located on the same row and column of the pixel being processed, we should follow this direction (northeast). The last black pixel will be at the location (column 5, row 6) as in Figure. 14 (a). Then we delete all the black pixels between (column 1, row 9) and (column 5, row 6). For each of the deleted pixels we draw a black pixel in column 1 because it was the first column. And we draw another black pixel in the row 6 because it was the first row as in Figure. 14 (b) - (c).

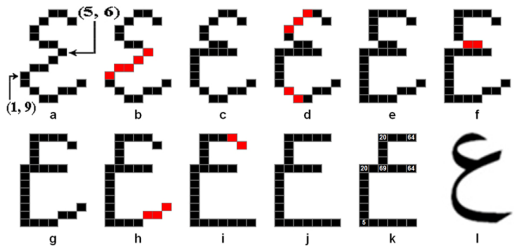


Figure. 14. A regression process used to recognize an Arabic letter. The vector is (64, 64, 20, 69, 20, 5, 0) for the geometric features of this letter as listed in Table 1.

Algorithm 3 Determination of the letter ع.

Input:

A skeleton image and integer value V = 69, V represent a value of the black pixel that lies at the intersection of two lines of the letter ع.

Output:

Boolean value result = True, if and only if the pixels represent a letter ع.

1: **If** (in the right side of V there is a black pixel with value 64)
   And {(in the top side of V there is a black pixel with value 20 is found) and (in the right side of it there is a black pixel with value 64 is found)}
   And {(in the left side of V there is a black pixel with value 20 is found) and (in the down side of it there is a black pixel with value 5 is found) and (in the right side of it there is another black pixel with value (64 or 65) and in the top side of it is 16)}
2: result = true
3: **Else**
4: result = false
5: **Return** result

All Arabic letters are recognized by using algorithms more or less as same as Algorithm 3.
Algorithm 4 is used to generate the recognized letters based on the vectors in Table 1. When a given vector has no match with anyone in Table 1, an error occurs.

Algorithm 4 Characters Recognition.

Input:

A skeleton image s which obtained after applying all the previous algorithms.

Output:

Arabic text; corresponds to what exists in the input image.

1: **Change** the colour of start, corner, intersection, and end pixels to the grey.
2: **Collect** all the grey pixels in a vector, from every column just take the first 2 grey pixels.
3: **If** there is no dot up or down on the letter
Add 0 to the end of the vector
4: **Else**
5: **If** 1 or 2 or 3 dots up of the letter
Add dots number to end of the vector
6: **Else**
7: **If** 1 or 2 dots down of the letter
Add (- dots number) to end of the vector
8: **End** if;
9: **End** if;
10: **Compare** this vector with stored vectors for Arabic letters and get the corresponding text.
11: **Return** text.

TABLE 1: Vectors of Canonical Arabic Letters.

ا	{16, 1, 0, 0, 0, 0, 0}	و	{80, 81, 20, 5, 0, 0, 0}
ب	{16, 65, 16, 5, 0, 0, -1}	ي	{64, 80, 20, 5, 16, 5, -2}
ت	{16, 65, 16, 5, 0, 0, 2}	ـا	{16, 5, 0, 0, 0, 0, 0}
ث	{16, 65, 16, 5, 0, 0, 3}	ـب	{16, 69, 16, 5, 0, 0, -1}
ج	{80, 65, 20, 69, 20, 5, -1}	ـت	{16, 69, 16, 5, 0, 0, 2}
ح	{80, 65, 20, 69, 20, 5, 0}	ـث	{16, 69, 16, 5, 0, 0, 3}
خ	{80, 65, 20, 69, 20, 5, 1}	ـج	{80, 69, 20, 69, 20, 5, -1}
د	{80, 65, 4, 4, 0, 0, 0}	ـح	{80, 69, 4, 20, 0, 0, 0}
ذ	{80, 65, 4, 4, 0, 0, 1}	ـخ	{80, 69, 20, 69, 20, 5, 1}
ر	{16, 2, 0, 0, 0, 0, 0}	ـد	{80, 69, 4, 4, 0, 0, 0}
ز	{16, 2, 0, 0, 0, 0, 1}	ـجـ	{80, 69, 20, 69, 0, 0, -1}
س	{16, 65, 16, 69, 20, 65, 0}	ـحـ	{80, 69, 4, 0, 0, 0, 0}
ش	{16, 65, 16, 69, 20, 65, 3}	ـخـ	{80, 69, 20, 69, 0, 0, 1}
ص	{80, 65, 20, 69, 20, 65, 0}	ـسـ	{16, 69, 16, 69, 16, 69, 0}
ض	{80, 65, 20, 69, 20, 65, 1}	ـشـ	{16, 69, 16, 69, 16, 69, 3}
ط	{80, 65, 16, 69, 20, 69, 0}	ـصـ	{80, 69, 20, 69, 16, 69, 0}
ظ	{80, 65, 16, 69, 20, 69, 1}	ـضـ	{80, 69, 20, 69, 16, 69, 1}
ع	{64, 64, 20, 69, 20, 5, 0}	ـعـ	{80, 69, 80, 69, 20, 5, 0}
غ	{64, 64, 20, 69, 20, 5, 1}	ـغـ	{80, 69, 80, 69, 20, 5, 1}
ف	{80, 81, 20, 5, 16, 5, 1}	ـفـ	{80, 69, 20, 69, 0, 0, 1}
ق	{80, 81, 20, 5, 16, 5, 2}	ـقـ	{80, 69, 20, 69, 0, 0, 2}
ك	{16, 68, 16, 5, 0, 0, 5}	ـكـ	{64, 0, 20, 69, 0, 0, 0}
ل	{16, 68, 16, 5, 0, 0, 0}	ـلـ	{16, 70, 0, 0, 0, 0, 0}
م	{80, 65, 20, 69, 20, 1, 0}	ـمـ	{84, 65, 84, 5, 0, 0, 0}
ن	{16, 65, 16, 5, 0, 0, 1}	ـنـ	{16, 69, 0, 0, 0, 0, 1}
ه	{80, 65, 20, 5, 0, 0, 0}	ـهـ	{20, 65, 80, 5, 0, 0, 0}

As a summary, we illustrate the process of character recognition in Figure 15. It should be mentioned that the lengths of orthogonal lines are ignored (i.e., regarded as only one pixel long) in our coding scheme. Figures. 15 and 16 illustrate an overview extraction for our technique process for two of Arabic words.

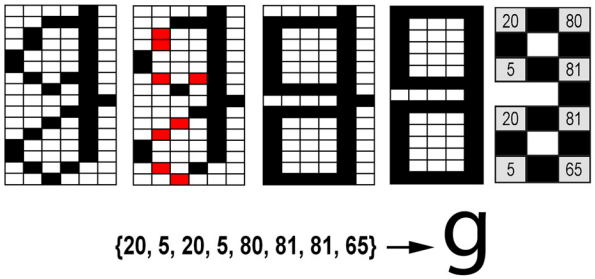


Figure. 15. The regression process for letter g.

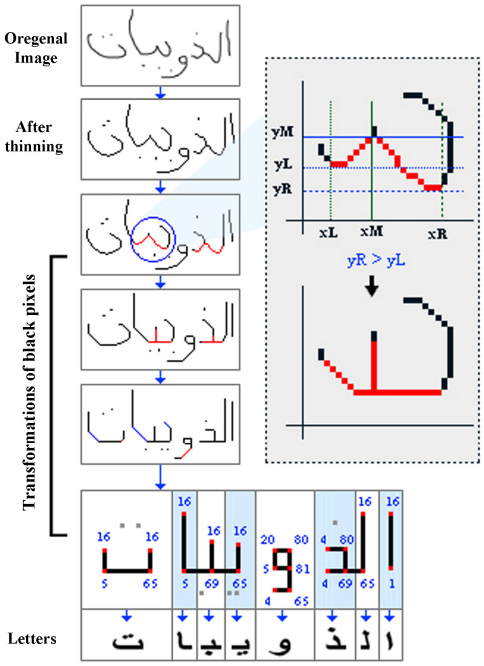


Figure. 16. An overview of our technique process for one of the Arabic words "الذوبيات".

4. Results of Experiments

Several experiments were done to measure the performance of the developed methods on different types of input data. We will describe the results of each of those methods.

4.1. Experiment Setup

The proposed algorithms are implemented in Visual Studio 2008. The PC used for experiments has an Intel(R) Core(TM)2 CPU 6400 @ 2.13GHz, 2 GB memory.

This section presents a comparison of the results with those of other researchers in the same area. There are two main parts in this section: the first presents the results of the experimental analysis of pre-processing algorithms, while the second part deals with character recognition results.

4.2. Evaluation of Pre-processing Algorithms

To evaluate the efficiency and the performance of our proposed pre-processing algorithms of binarization, skeletonization, and segmentation, the following tests are implemented. We applied the algorithms on 10 handwritten and 10 printed pages of Arabic words with different grey levels. A total of 2000 words are used in our experiments, and we split our data set into two parts. The 60% of the pages were randomly chosen as a training data set, and the remaining pages were used as the testing data set. The results show that our proposed algorithms (binarization, skeletonization and segmentation to lines and words) perform very well with respect to different styles of Arabic handwriting (see Table 2). The accuracy of segmentation is evaluated based on the number of correctly segmented lines and words and the total number of lines or words. For all images we tested, the binarization algorithm worked at a rate of 97% of accuracy, the performed well in these initial tests.

As seen from Table 2, the binarization and skeletonization algorithms achieved an accuracy of 98% and 99% respectively for the testing set for the handwriting pages, and 98% for the printed pages.

The segmentation of printed pages has better results than that of handwritten pages, as it can be seen in the result of page-to-line segmentation (Table 2).

TABLE 2: The Result of First Test.

Page No.		Accuracy							
		Binarization		Skeletonization		Segmentation to Lines		Segmentation to Words	
		Training	Testing	Training	Testing	Training	Testing	Training	Testing
Handwritten Pages	1	97%	97%	98%	98%	98%	100%	97%	99%
	2	97%	98%	99%	100%	98%	100%	98%	99%
	3	97%	97%	99%	100%	99%	99%	97%	98%
	4	99%	99%	98%	100%	99%	99%	98%	100%
	5	99%	100%	97%	97%	96%	99%	97%	99%
	6	96%	98%	98%	99%	98%	99%	97%	99%
	7	94%	98%	98%	98%	98%	99%	97%	99%
	8	95%	95%	98%	99%	98%	98%	99%	99%
	9	96%	99%	98%	100%	97%	97%	96%	98%
	10	98%	99%	98%	99%	99%	100%	98%	100%
Average		97%	98%	98%	99%	98%	99%	97%	99%
Printed Pages	1	95%	98%	96%	99%	99%	99%	98%	99%
	2	98%	99%	97%	98%	99%	99%	97%	99%
	3	99%	100%	97%	98%	99%	100%	97%	100%
	4	99%	99%	97%	97%	100%	100%	97%	100%
	5	96%	97%	98%	98%	99%	99%	98%	99%
	6	98%	99%	96%	99%	99%	100%	99%	99%
	7	97%	99%	98%	99%	100%	100%	99%	99%
	8	97%	99%	99%	100%	98%	99%	98%	99%
	9	96%	98%	99%	99%	99%	99%	98%	99%

	10	98%	99%	98%	99%	99%	100%	99%	100%
Average		97%	98%	97%	98%	99%	99%	98%	99%

We compared the performance of our binarization algorithm with two well-known binarization techniques (see in Table 3). We evaluated the following: Otsu's global thresholding method [8], Niblack's thresholding method [9], and our proposed method. Table 3 shows that our proposed algorithm has demonstrated the best performance regarding the final results.

TABLE 3: Result of Binarization Algorithms.

Type	Data Set	Words	Methods					
			Otsu		Niblack		Proposed	
			Correct Words	Rate	Correct Words	Rate	Correct Words	Rate
Hand	Training	1200	1056	88%	1116	93%	1176	98%
	Testing	800	664	83%	712	89%	784	98%
Print	Training	1200	1128	94%	1140	95%	1188	99%
	Testing	800	696	87%	720	90%	776	97%

4.3. Evaluation of Recognition Algorithm

In our experiments, we evaluate the efficiency and accuracy of our proposed algorithm using the demo version of the benchmark IFN/ENIT database of Arabic city names [16]. It was produced by the Institute for Communications Technology at the Technical University of Braunschweig (IFN). This includes 569 handwritten Arabic images. These images represent the names of Tunisian towns. The names of the towns might be having related sub-words, and they are of maximum three words.

We evaluate our proposed algorithm using 2160 images for the English capital letters and Arabic numbers, in the handwritings of different people. From the 1300 words in the training data set, the average accuracy of 93.7% is achieved. The average accuracy of 93.1% is achieved for the 860 words in the testing data set. These results are presented in Table 4.

TABLE 4: Results of Our Recognition Algorithm.

Data Set	Recognition					
	Training			Testing		
	Words/ Letter	Correct W/L	Rate	Words/ Letter	Correct W/L	Rate
IFN/ENIT- Words	356	328	92.1%	240	224	93.3%
English Letters	1300	1219	93.7%	860	799	93.1%

4.4. Comparison to Other Approaches

The IFN/ENIT database is available to the general public for research purposes and this makes system comparison possible. In this section, the performance of our system is compared with the other Arabic handwriting recognition systems, that use datasets a, b, and c of IFN/ENIT-database for training and set d for testing. Relevant results are presented in Table 5.

Table 5 summarizes the performance of current Arabic handwriting recognition systems tested on IFN/ENIT-database. The average accuracies of our technique are higher than that of other techniques tested. We can conclude that our technique improved the recognition of Arabic handwritten words and it is robust to different styles of personal Arabic handwriting.

TABLE 5: Comparison with Other systems.

Method	Features	Rate
Pechwitz et al. [16]	Skeleton directions, Pixel values	89.10%
Menasri et al. [17]	Graphemes	87.40%
Touj et al. [18]	Directional values, connection of graphemes	86.10%
Dreuw et al. [19]	Image slices and their spatial derivatives	92.86%
Natarajan et al. [20]	Percentile of intensity, energy, correlation, angle	89.40%
Benouareth [21]	Distribution, concavity and skeleton features	90.20%
Al–Hajj et al. [22]	Distribution and concavity features	90.26%
Our Method	**Correction diagonally connected pixels**	**93.30 %**

5. Conclusions

In this paper, we propose an effective and efficient approach for Arabic handwriting recognition based on a novel representation of geometrical features of Arabic characters. The main contributions of this paper can be summarized as follows:

An effective binarization algorithm that does not require user parameters is proposed. The resulting binarized images do not contain noise in non-text areas. The text is distinguished from the background with high accuracy.

These techniques collectively form a complete process for reliable Arabic handwriting recognition. Our experiments show that the proposed approach achieves significantly higher accuracy rates than existing approaches. Figure. 17 shows 28 images of different IFN/ENIT words. Although the proposed system is applied to Arabic handwritten pages, it can be adapted to other languages handwriting such as Latin language. Figure. 18 shows 42 images of different English letters and numbers and the recognition results. Our approach has demonstrated its potential and has achieved much better results than the existing approaches.

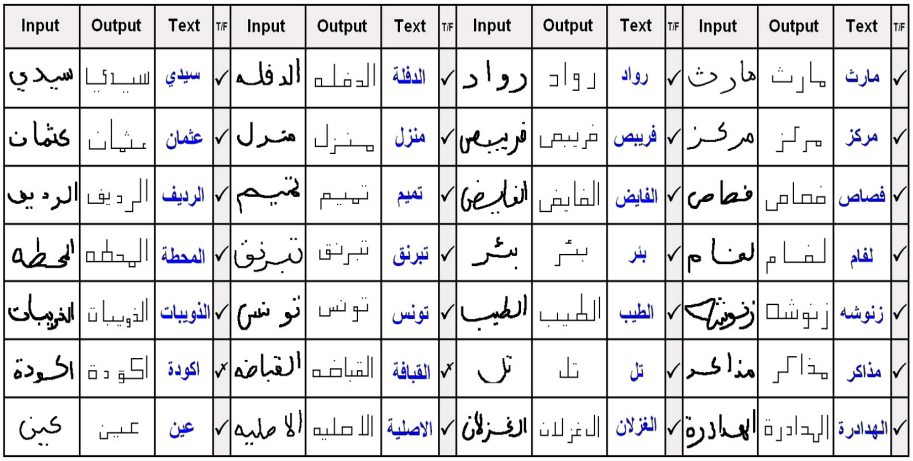


Figure. 17. Results of testing some IFN/ENIT words.

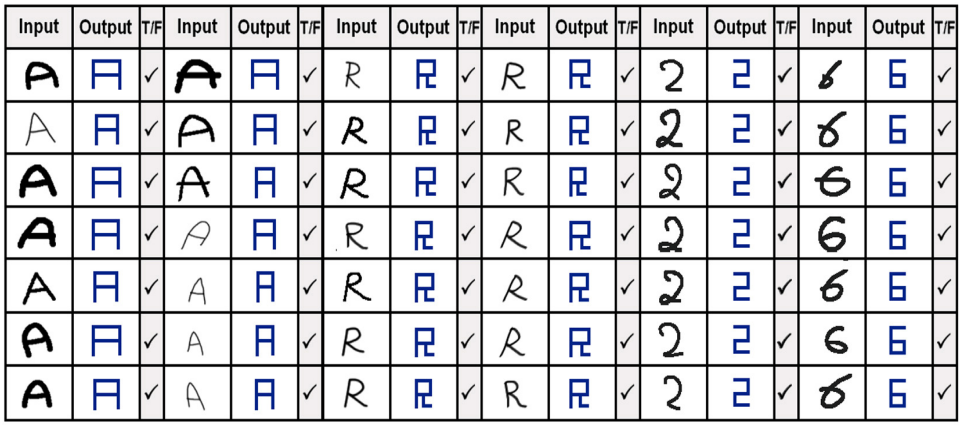


Figure. 18. Results of testing some English letters and numbers.

REFERENCES

[1] Kégl, B. and A. Krzyzak (2002). "Piecewise linear skeletonization using principal curves." IEEE Trans. Pattern Anal. Mach. Intell 24: 59–74.

[2] Harous, S. and A. Elnagar (2009). "Handwritten Character-Based Parallel Tinning Algorithms: A Comparative Study." University of Sharjah Journal of Pure and Applied Sciences 6: 81 - 101.

[3] EI-Badr, B. and S. A. Mahmoud (1995). "A survey and bibliography of Arabic optical text recognition." Signal Processing 41: 49-76.

[4] Zheng, L. (2006). "Machine Printed Arabic Character Recognition Using S-GCM." Proceedings of 18th International Conference on Pattern Recognition 2: 893–896.

[5] Mandana, K. and A. Amin (1999). "Preprocessing and Structural Feature Extraction for a Multi-Fonts Arabic / Persian OCR." in Document Analysis and Recognition ICDAR: Proceedings of the Fifth International Conference 213-216.

[6] F. Crick (1988). "Chapter 8: The genetic code, What mad pursuit: a personal view of scientific discovery." New York: Basic Books: 89–101.

[7] Mohamed, M. and P. Gader (1996). "Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques." IEEE Transactions on Pattern Analysis and Machine Intelligence 18(5): 548.

[8] Otsu, N. (1979). "A Threshold Selection Method from Gray Level Histograms." IEEE Trans. on Systems, Man and Cybernetics 9: 62-66.
[9] Niblack, W. (1986). "An Introduction to Digital Image Processing." Englewood Cliffs, N. J., Prentice Hall 115-116.
[10] P. Ahmed (1995). "A neural network based dedicated thinning method." Pattern Recognition Letters 16 (6): 585-590.
[11] N. J. a. S. Naccache, R. (1984). "SPTA: A Proposed Algorithm for Digital Pictures." IEEE Trans. on Systems, Man and Cybernetics SMC-14(3): 409-418.
[12] T. Y. a. S. Zhang, C. Y. (1984). " A Fast Parallel Algorithm for Thinning Digital Patterns." Comm. ACM 27(3): 236-239.
[13] Amin, A. (1998). "Off-Line Arabic Character Recognition: The State of the Art." Pattern Recognition Letters 31(5): 517-530.
[14] Vinciarelli, A. (2002). "A Survey on Off-Line Cursive Word Recognition." Pattern Recognition 35: 1433-1446.
[15] Lorigo, L. and V. Govindaraju (2006). "Off-line Arabic Handwriting Recognition: A survey " IEEE Transaction on Pattern Analysis and Machine Intelligence 28: 712-724.
[16] Pechwitz, M., V. Maergner, et al. (2006). "Comparison of Two Different Feature Sets for Offline Recognition of Handwritten Arabic Words." 10th International Workshop on Frontiers in Handwriting Recognition (IWFHR).
[17] Menasri, F., N. Vincent, et al. (2007). "Shape–based Alphabet for Off–line Arabic Handwriting Recognition." 9th International Conference on Document Analysis and Recognition (ICDAR): 969–973.
[18] Touj, S. M., B. Amara, et al. (2007). "A Hybrid Approach for Off-Line Arabic Handwriting Recognition Based on a Planar Hidden Markov Modeling." Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR): 964 - 968.
[19] Dreuw, P., S. Jonas, et al. (2008). "White–Space Models for Offline Arabic Handwriting Recognition." 19th International Conference on Pattern Recognition (ICPR).
[20] Natarajan, P., S. Saleem, et al. (2008). "Multi- lingual Offline Handwriting Recognition Using Hidden Markov Models." A Script-Independent Approach, Arabic and Chinese Handwriting Recognition LNCS 4768: 231-250.
[21] Benouareth, A., A. Ennaji, et al. (2008). "Semi–continuous HMMs with Explicit State Duration for Unconstrained Arabic Word Modeling and Recognition." Pattern Recognition Letters 29(12): 1742–1752.
[22] Mohamad, R. A., L. Likforman-Sulem, et al. (2009). "Combining Slanted- Frame Classifiers for Improved HMM-Based Arabic Handwriting Recognition." IEEE Transactions on Pattern Analysis and Machine Intelligence 31(7): 1165-1177.

Authors

Jafaar Alabodi. He is currently a Ph.D student at the University of Queensland. He received his Bachelor's degree in Computer Science from Al-Basra University, Iraq in 1985. He obtained a Master's Degree in Computer Science from the University of Queensland, Australia in 2007. His research interests include pattern recognition and distributed systems.

Xue Li. Is an Associate Professor in the School of Information Technology and Electrical Engineering at the University of Queensland, Australia. He graduated in Computer Science from Chongqing University, China in 1982. He obtained Ph.D degree in Information Systems from Queensland University of Technology in 1997. Dr Xue Li's research areas are Data Mining, Multimedia Data Security, and Intelligent Web Information Systems. He is a member of ACM and IEEE.