# OFF-LINE SYSTEM FOR THE RECOGNITION OF HANDWRITTEN ARABIC CHARACTER

Ahmed Sahlol[1] and Cheng Suen [2]

[1]Department of Computer Teacher preparation,
Damietta University, Damietta, Egypt
asahlol@encs.concordia.ca
[2]Department of Computer Science, Concordia University, Canada
parmidir@encs.concordia.ca

## ABSTRACT

*Recognition of handwritten Arabic text awaits accurate recognition solutions. There are many difficulties facing a good handwritten Arabic recognition system such as unlimited variation in human handwriting, similarities of distinct character shapes, and their position in the word. The typical Optical Character Recognition (OCR) systems are based mainly on three stages, preprocessing, features extraction and recognition.*

*In this paper, we present an efficient approach for the recognition of off-line Arabic handwritten characters which is based on structural, Statistical and Morphological features from the main body of the character and also from the secondary components. Evaluation of the accuracy of the selected features is made. The system was trained and tested with CENPRMI dataset. The proposed algorithm obtained promising results in terms of accuracy (success rate of 100% for some letters at average 88%). In Comparable with other related works we find that our result is the highest among others.*

## KEYWORDS

*Handwritten Arabic Characters, Feature extraction, Secondary component*

## 1. INTRODUCTION

The estimated number of historical Arabic manuscripts exceeds three millions [1], so we translate images of typewritten or handwritten text into machine-editable text encoded in a standard encoding scheme (ASCII). Handwritten text recognition of such as Arabic text is an active research problem [2].

Now OCR systems have expanded to recognize Latin alphabets, Japanese Katakana syllabic characters, Kanji (Japanese version of Chinese) characters, Chinese characters, Hangul characters, etc.

Work on Arabic OCR started in 1970s [3]. The first published work on Arabic OCR dates back to 1975 [5]. The first Arabic OCR system was made available in 1990s [6]. The recognition of Arabic handwriting presents some unique challenges and benefits to the researchers [7]. Although more than three decades have passed, there has been a lack of effort in the recognition of Arabic handwritten texts compared to the recognition of texts in other scripts [6].

The main problem encountered when dealing with handwritten Arabic characters is that characters written by different persons representing the same character are not identical but can vary in both size and shape. Unlimited variation in human handwriting styles similarities of distinct character shapes, character overlaps, and interconnections of neighboring characters. In addition, the mood of the writer and the writing situation can have an effect on writing styles.

## ▪ Arabic Writing System

Arabic is written from right to left and is always cursive. It has 28 basic characters. Thus, roughly the alphabet set can expand to 84 different shapes according to the position of the letter (beginning, middle, end or isolated) as well as according to the style of writing (Nasekh, Roqa'a, Farisi and others).

A character is drawn in an isolated form when it is written alone and is drawn in three other forms when it is written connected to other characters in the word. For example, the character Ain has four forms: isolated, initial, medial and final. See Fig. 1.

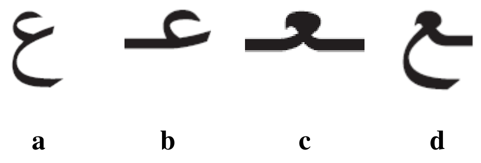


Figure 1. Different forms of Ain character: **a**. isolated, **b**. initial **c**. medial **d**. final

The secondary components are character components that are disconnected from the main body. Sixteen Arabic letters have from one to three secondary components (dots).

Additionally, some characters like alif (أ), kaaf (ك) can have a zigzag-like stroke called Hamza (ء). Those dots and Hamzas are called secondaries and they are located above the character primary part as in Alif (أ), or below like Baa (ب), or in the middle like Jeem (ج).

The type and position of the secondary components are very important features of Arabic letters. In the below Table1 we see letters can be only distinguished only by their secondary components. For example, Tah (ط) A1 and Thah (ظ) A2 differ only by the number of dots above the main body, also Seen (س) B1 and Sheen (ش) B2.

Another important kind of variations in drawing the secondary components appears mostly in drawing two or three dots. As shown in table1 Sheen (ش) B2, B3 and B4, the three dots come in three variations: isolated, connected and linked to the character, respectively. Also Taa (ت) C1, C2 can be drawn in two variations: two isolated dots or one short horizontal dashed line.

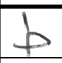
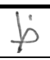
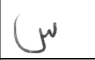
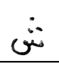
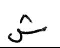
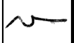
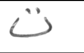
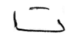
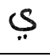
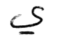
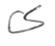
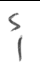
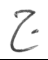
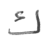
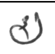
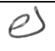
Another kind of differences between characters depended only on the position of the two dots; see Taa (ت) C1, Yaa (ي) D1 and Taa (ت) C2, Yaa (ي) D2 respectively.

There is also another classification challenge; that some characters which contain secondary components can also be written without those secondary components depending on "Roqa'a" writing styles. For example Yaa (ي) can be drawn with two isolated dots D1 or with short dashed line D2 or without any dots D3. Another example is Alif (أ) which can be drawn with hamza E1(default) or without it E2.

One recognition difficulty is due to some writers' styles which can join the secondary components of isolated and final forms with main body curves. Table 1 shows some examples: Samples F1 and F2 show how the one dot of isolated Geem (ج) is joined to the main character body.

Another difficulty in recognizing the secondary components comes due to quickly writing, as writers draw them connected to the main body. For example, Samples H2, H3 show hamza connected to Kaf's (ك) body.

Table 1: variations in drawing character secondary components

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| A |  |  | | |
| B |  |  |  |  |
| C |  |  | | |
| D |  |  |  | |
| E |  | | | |
| F |  | | | |
| H |  |  |  | |

Considerable work has been undertaken in the area of Arabic character recognition, their targeted in various ways to improve accuracy and efficiency but with limited success, this is due to the nature of Arabic characters and to the problems mentioned above.

El-Dabi et al. [8] presented a recognition system for typed Arabic text, which involves a statistical approach for character recognition. El-Sheikh et al [9] proposed algorithms to recognize Arabic handwritten characters; this system assumes that characters result from a reliable segmentation stage, thus, the position of the character is known a priori. Four different sets of character shapes have been independently considered (initial, medial, final, and isolated). Each set is further divided into four subsets depending on the number of strokes in the character. El-Khaly et al. [10] discussed an algorithm for the machine recognition of optically captured Arabic characters and their isolation from the printed text. Moment-invariant descriptors are investigated for the purpose of recognition of individual characters. Sabri Mohmoud [11] has used Fourier and contour analysis for the recognition of Arabic characters with acceptable recognition rates. The features of an input character are compared to the models' features using a distance measure. The model with the minimum distance is taken as the class representing the character.

If we look at a character as image from which we can extract much useful information, such information can be structural features such as loops, branch-points, endpoints, and dots or statistical which include pixel densities, histograms of chain code directions, moments, and Fourier descriptors. Many approaches and techniques have been proposed like [12] [13] [14] [15] [16] used loops, dots, curves, relative locations, height, sizes of parts of characters, loop positions and types, line positions and directions and turning points. Others like [17] [18] [19] used statistics from moments of horizontal and vertical projection. Histogram of slopes along contour is used by [20].

Artificial Neural Networks (ANNs) are the common seed in most if not all classifiers recognition. In this paper we use Neural Networks as a classifier like [21] which proposed a system for the recognition of handwritten Arabic characters recognition. Also Sherif and Mostafa [22] [23] presented a parallel design for back propagation Neural Networks approach in order to accelerate the computation process. But another kind of Neural network called Learning Vector Quantization (LVQ) was used in [24] for handwritten Arabic character recognition. While others [25] [26] used SVM (support vector machine) as a classifier for Arabic numeral.

In this paper, we propose a novel approach for extracting statistical, morphological and topological features of handwritten Arabic characters. We apply this technique in extracting moment features and show that this technique provides better feature sets that give higher recognition accuracies.

This paper is organized in nine sections. Section 2 describes the Binarization Algorithm used to convert grayscale image to be binary image, Section 3 introduces the normalization technique used for reducing the shape variation between the images. Section 4 describes some kind of noise removal Algorithms which have an important contribution during the classification stage. Section 5 illustrates different kind of feature extraction techniques used to extract useful features from character images. Section 6 describes the Classification phase; which analyzes the used classifier including its architecture, training and testing phase. Section 7 provides experimental results included classification accuracy and analysis. Finally, Section 8 describes the main conclusions and future work.

## 2. MATERIALS AND METHODS

The goal of this work is to develop a system that recognizes off-line Arabic handwritten characters. Features needed for the recognition process include different kinds of features. Neural network is then used to classify the characters based on the features that were extracted from the input character.

### 2.1 Binarization

Our purpose of this step is to convert the input image to binary image based on threshold.
Binary images may contain numerous imperfections. In particular, the binary regions produced by simple thresholding are distorted by noise and texture. Morphological image processing pursues the goals of removing these imperfections by accounting for the form and structure of the image.
Replace all pixels in the input image with luminance greater than level with the value 1 (white) and replace all other pixels with the value 0 (black).

Computes a global threshold (level) that can be used to convert an intensity image to a binary image is a normalized intensity value that lies in the range [0, 1].

We use Otsu’s method [27], which chooses the threshold to minimize the intraclass variance of the black and white pixels.

Otsu’s thresholding method involves iterating through all the possible threshold values and calculating a measure of spread for the pixel levels at each side of the threshold, i.e. the pixels that either fall in foreground or background. The aim is to find the threshold value where the sum of foreground and background spreads is at its minimum.

The algorithm assumes that the image to be thresholded contains two classes of pixels (e.g. foreground and background), then calculates the optimum threshold separating those two classes so that their combined spread (intra-class variance) is minimal.

we use Otsu method not only because it is a global binarization technique but also its short running time; the Mean running time for the Otsu's binarization method was 2.0 secs and this is one of the lowest running time (Original Sauvola algorithm [28] takes 12.6 secs).

## 2.2 Slant Correction

We make slant Correction for every character image to eliminate any slant in each character. The basic idea is to locate near-vertical strokes in the character and estimate the average slant of the character from these strokes. Then, the slant in a character is corrected by applying a shear transformation to the character.

## 2.3 Normalization.

Normalization is to regulate the size, position, and shape of character images, so as to reduce the shape variation between the images of same class. Denote the input image and the normalized image by f(x, y) and g(x□,y□), respectively, normalization is implemented by coordinate mapping

$$\begin{cases} x' = x'(x,y), \\ y' = y'(x,y). \end{cases} \quad (1)$$

we denote the width and height of the original character by W1 and H1, the width and height of the normalized character by W2 and H2, and the size of the standard (normalized) plane by L As seen in Figure 3. The standard plane is usually considered as a square and its size is typically 32 × 32 or 64 × 64, among others. We define the aspect ratios of the original character (R1) and the normalized character (R2) as

$$R_1 = \frac{\min(W_1, H_1)}{\max(W_1, H_1)}$$

and

$$R_2 = \frac{\min(W_2, H_2)}{\max(W_2, H_2)} \quad (2)$$

which is always considered in the range of [0, 1].

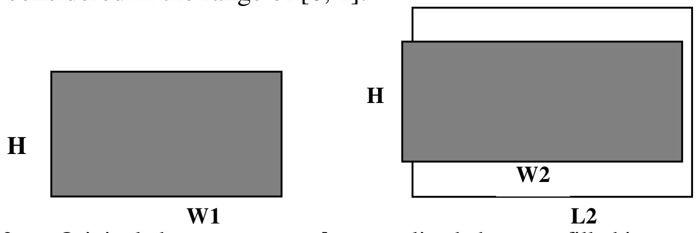


Figure 2. **a**. Original character **b**. normalized character filled in standard plane.

We use Linear Backward mapping method [29] where:

$$\begin{aligned} x &= x'/\alpha \\ y &= y'/\beta \end{aligned} \quad (3)$$

and α and β denote the ratios of scaling, given by:

$$\alpha = W_2 / W_1$$
$$\beta = H_2 / H_1 \qquad (4)$$

Where W1 and H1 are the horizontal span and vertical span of the strokes of the original character (size of minimum bounding box).

## 2.4 Noise removal

### 2.4.1 Statistical Noise removal

In addition to enhancement of character image by contrast and dynamic range modification, a character image can also be enhanced by reducing degradations that may be present. This area of image enhancement overlaps with image restoration.

Median filtering [30] is a nonlinear process useful in preserving edges in an image while reducing random noise.

The median is calculated by first sorting all the pixel values from the surrounding neighborhood into numerical order and then replacing the pixel being considered with the middle pixel value. (If the neighborhood under consideration contains an even number of pixels, the average of the two middle pixel values is used). Figure 4 shows the process of Median filtering.

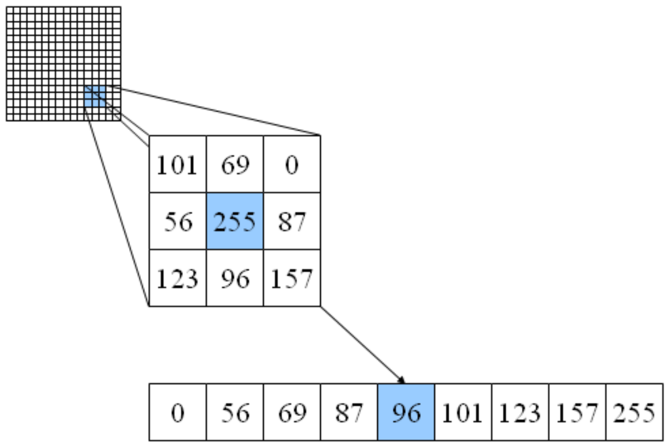


Figure 3. Median filtering process

A template of size 3x3, 5x5, 7x7,… etc is applied to each pixel. The values within this template are sorted and the middle of the sorted list is used to replace the template central pixel.

Median filtering can preserve discontinuities in a step function and can smooth a few pixels whose values differ significantly from their surroundings without affecting the other pixels [31]. In this paper, several filters were tested however a 3 × 3 median filter was chosen because it gave us the best result. The median filter is used to reduce noise in an image by considering each pixel within its neighboring pixels to decide whether or not it is representative of its surroundings. Then, it replaces the pixel value with the median of the values of the neighboring pixels.

### 2.4.2 Morphological noise removal

Fills isolated interior pixels (individual 0s that are surrounded by 1s), such as the center pixel in this pattern.

For each pixel p in the binary image I, check the two neighbors (as shown in the below figure) and decide whether P to be 0 or 1 if B(p) =4, where B(p) is the number of non-zero neighbors of p.

$$\begin{matrix}1 & 1 & 1\\ 1 & p & 1\\ 1 & 0 & 1\end{matrix} \quad \text{becomes} \quad \begin{matrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{matrix}$$

Figure 4. Filling Process

Bridges unconnected pixels, that is, sets 0-valued pixels to 1 if they have two nonzero neighbors that are not connected.

For each pixel p in the binary image I, check the two neighbors (as shown in the below figure) and decide whether P to be 0 or 1 if B (p1)>=2, where B (p) is the number of non-zero neighbors of p.

$$\begin{matrix}1 & 0 & 0\\ 1 & p & 1\\ 0 & 0 & 1\end{matrix} \quad \text{becomes} \quad \begin{matrix}1 & 1 & 0\\ 1 & 1 & 1\\ 0 & 1 & 1\end{matrix}$$

Figure 5. Bridging Process

Remove isolated pixels (individual 1s that are surrounded by 0s), such as the center pixel in this pattern.

For each pixel p in the binary image I, check the two neighbors (as shown in the below figure) and decide whether P to be 0 or 1, if B(p) =0, where B(p) is the number of non-zero neighbors of p.

$$\begin{matrix}0 & 0 & 0\\ 0 & p & 0\\ 0 & 0 & 0\end{matrix} \quad \text{becomes} \quad \begin{matrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{matrix}$$

Figure 6. Removing process

Another kind Morphological operation used in this paper is Dilation:

- **Dilation**

It is an operation that grows or thickens objects in a binary image The specific manner and extent of this thickening is controlled by a shape referred to as a structuring element. This Morphological technique exposes an image to a small shape or template; the structuring element is positioned at all possible locations in the image and it is compared with the corresponding neighborhood of pixels. Some operations test whether the element "fits" within the neighborhood, while others test whether it "hits" or intersects the neighborhood. Figure 8 shows how dilation works .

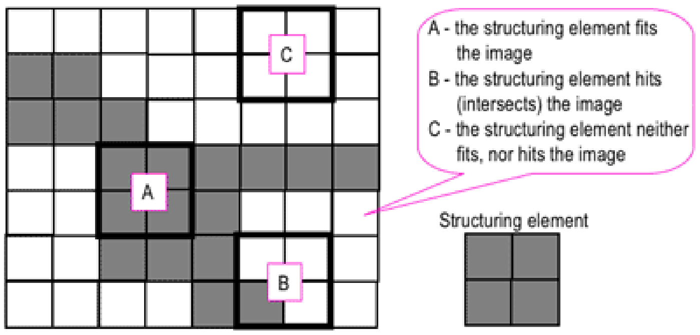

A - the structuring element fits the image
B - the structuring element hits (intersects) the image
C - the structuring element neither fits, nor hits the image
Structuring element
A
B
C

Figure 7. Exposing an image to a structuring element

(white and grey pixels have zero and non-zero values, respectively).

A morphological operation on a binary image creates a new binary image in which the pixel has a non-zero value only if the test is successful at that location in the input image.

The dilation of A by B is defined by:

$$A \oplus B = \bigcup_{b \in B} A_b \qquad (5)$$

The dilation is commutative, also given by:

$$A \oplus B = B \oplus A = \bigcup_{a \in A} B_a \qquad (6)$$

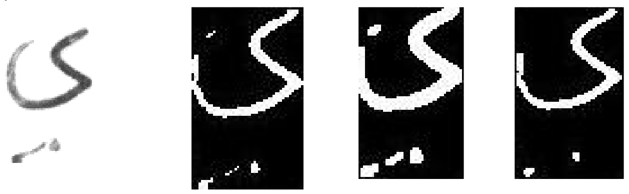
If B has a center on the origin, then the dilation of A by B can be understood as the locus of the points covered by B when the center of B moves inside A. The dilation of a square of side 10, centered at the origin, by a disk of radius 2, also centered at the origin, is a square of side 14, with rounded corners, centered at the origin. The radius of the rounded corners is 2.
The dilation can also be obtained by:

$$A \oplus B = \{z \in E \mid (B^s)_z \cap A \neq \varnothing\} \qquad (7)$$

where ∅ is the empty set and B is the st y of B, that is:

$$B^s = \{x \in E \mid -x \in B\} \qquad (8)$$

In this work we create a square of 2x2 of ones as a structuring element. Example of badly and for good dilation see fig. 9.



a b c d

Figure 8. Dilation for Yaa character **a**. Original image **b**. primary Preprocessed **c**. badly Dilated **d**. perfect Dilated

## 2.5 Feature Extraction

### 2.5.1 Structural Features:

#### 2.5.1.1 Upper and Lower profile.

The upper and lower profiles capture the outlining shape of a connected part of the character. Upper (or lower) connected part profile is computed by measuring the distance (pixel count) of each column group from the top (or, bottom) of the bounding box of the connected part to the closest ink pixel in that column group.

Algorithm: Upper (or Lower) Profile

Input: binary preprocessed character image

Output: feature vector, F, representing upper (or, lower) profile.

```
1. Read the image into a two-dimensional array.
2. Divide the width into g column groups.
3. for each column group
```

Compute the distance from the top (or, bottom) of the bounding box of the connected part to the closest ink pixel in that group by counting the number of white pixels.
Get the ratio of distance to the number of black pixels of each group. Figure 10 shows the upper and lower profiles of character "Faa".

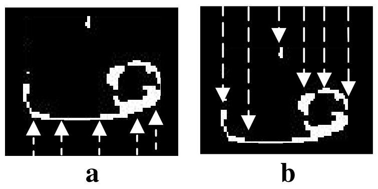


Figure 9. **a**. Upper profile **b**. Lower profile for character "Faa"

### 2.5.1.2 Horizontal and Vertical projection profiles.

Projection profile based feature extraction method delivers excellent results even in the absence of some important preprocessing steps such as smoothing and thinning. In fact, in this type of feature extraction it will be disadvantageous to apply the thinning process because there will be a huge loss of important information related to the count and position of white pixels present in the character image.

Vertical profile is the sum of white pixels perpendicular to the y axis. It is computed by scanning the character column wise along the y-axis and counting the number of white pixels in each column.

Similarly, for horizontal projection profile is sum of black pixels but it is perpendicular to the x axis. The character is traced horizontally along the x-axis. The row wise sum of number of white pixels present in each row. See fig. 10.

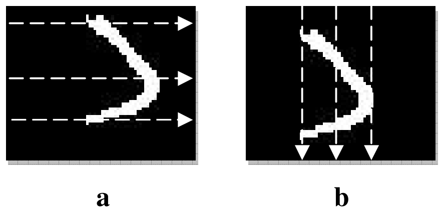


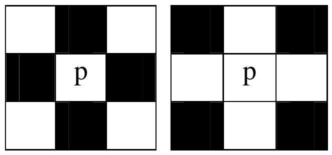
Figure 10. Character Daal projection profile **a**. horizontally **b**. vertically

## 2.5.2 Statistical Features:

### 2.5.2.1 Connected components.

We consider the important “middle ground” between the individual foreground pixels and the set of all foreground pixels. This leads to the notion of connected components, also referred to as the following objects:

A pixel p at coordinates (x,y) has two horizontal and two vertical neighbors whose coordinates are (x+1,y), (x-1,y), (x,y+1) and (x,y-1). This set of 4 neighbors of p denoted N4 (p) is illustrated in figure 11, where the four diagonal neighbors of p have coordinates (x+1,y+1), (x+1,y-1), (x-1,y+1) and (x-1,y-1) respectively.



**a** **b**

Figure 11. **a**. Pixel p and its 4 neighbors **b**. Pixel p and its diagonal neighbours

Connected component is an important feature because most of Arabic characters contain one or more connected components like (أ, ب, ت, ث, خ, ذ, ض, غ, ق, ك, ي), so this feature can be effective for distinguishing among those characters which contain connected components and others which don’t contain connected components.

## 2.5.3 Topological features:

### 2.5.3.1 End points.

Another feature that is useful is the number of end points in the character. Those points have only one neighbor and the other three neighbors are noise.

Algorithm: Get end points of a character image

Input: binary preprocessed character image

Output: feature vector, F, representing the most common end point of the character.

1. Read the character image, preprocess it (Binarization, normalization,...)
2. Dilate the image to join all its parts. Dilation essentially just adds pixels around existing white pixels.
3. Shrink the image to points (removes pixels so that objects without holes shrink to a point, and objects with holes shrink to a connected ring halfway between each hole and the outer boundary).
4. Find the pixel that is still white, which i is the row and j is the column of the white pixel in image such that image(i,j) is equal to 1. All the other pixels should be 0 in the image.
5. Gather i and j in one vector “G”.
6- Normalized each of the vector elements using Min–max method [31]:
Normalized value=each value-min(all vector values))/(max(all vector values)-min(all vector values).
7- Determine the most repeated values by calculating differences between adjacent elements.
8- Finally, determine the Most frequent values in the character array .
9- Put all the end points for the same character in a feature vector.
Notes about the algorithm:

-At the 2nd stage "Dilation". We did it twice because we used Otsu method in binarization.
-At the 3rd stage "shrinking". We shrank the character image until we reach the fewest number of white pixels. The below figure describes all the stages for the above algorithm.

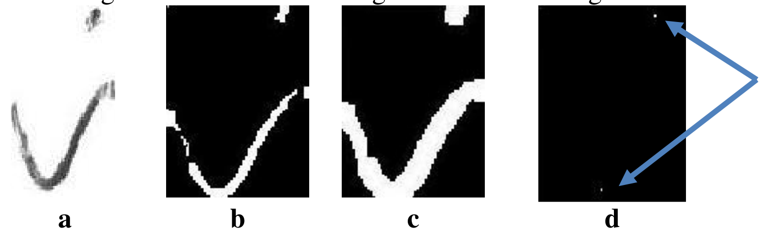


Figure 12. End point stages for Zaay character **a**. original **b**. preprocessed **c**. dilated d. end points

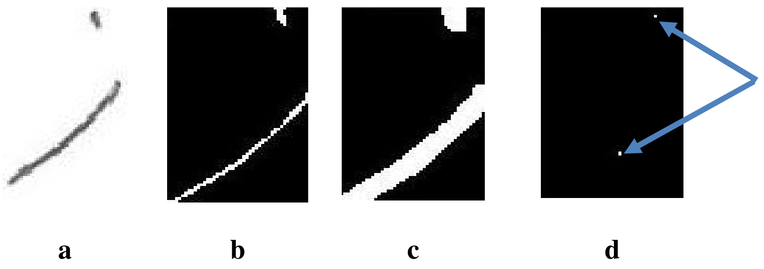


Figure 13. End point stages for another Zaay character **a**. original **b**. preprocessed **c**. dilated **d**. end points

It is obvious that the outer boundary points (end points) of the two Zaay characters are so close although they do not have the same shape. When we check the coordinates (step 4) of the end points of the two characters we find that the two characters have exactly the same coordinates of one from the two end points.

**2.5.3.2 Pixel Ratio.**

The character area is the total number of white (foreground) and black pixels (background) of the character. Pixel ratio is: (the number of white pixels / the number of black pixels).

**2.5.3.3 Height to Width Ratio.**

Since different people write same characters in different sizes, the absolute width and height are not reliable features for Arabic handwritten characters. However, some Arabic characters are wider than others. Therefore, the aspect ratio (height/width ratio) of the character is a useful feature.

- ## Feature Normalization

The simplest normalization technique is the Min–max normalization method [31]. Min–max normalization is best suited for the case where the bounds (maximum and minimum values) of the scores produced by a matcher are known. In this case, we can easily shift the minimum and maximum scores to 0 and1, respectively. However, even if the matching scores are not bounded, we can estimate the minimum and maximum values for a set of matching scores and then apply the min–max normalization.

We use this method because it has one of the best recognition performances. Also it is efficient but sensitive to outliers in the training data, but if the parameters of the matching score distribution is known this method would suffice.

When the minimum and maximum values are estimated from the given set of matching scores Min–max normalization retains the original distribution of scores except for a scaling factor and transforms all the scores into a common range [0, 1].

## 2.6 Classification

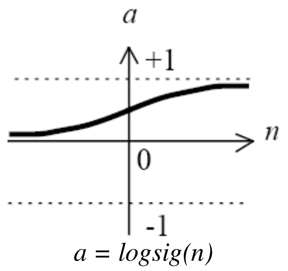
The classification stage is the decision making part of the recognition system.

The Feed Forward Neural Network (back propagation) (2-NN) [32] is one of the most powerful classifiers. The number of hidden units of ANN should be selected to be high enough to model the problem at hand but not too high to overfit. The number of hidden units is selected to have the best performance on the validation set.

### 2.6.1 Network Architecture

The input vectors are defined as a matrix called alphabet (133x1848), representing 1848 samples (66 character image x 28 "number of Arabic alphabet") of 133 elements (feature). The target vectors are also defined with variable called target. Each target vector is a 28-element vector with a 1 in the position of the letter. It represents, and 0's everywhere else. For example, the letter "أ" is to be represented by a 1 in the first element (as "أ" is the first letter of the alphabet), and 0's everywhere else. It is then required to identify the letter by responding with a 28-element target vector. The 28 elements of the target vector each represent a letter. To operate correctly, the network should respond with a 1 in the position of the letter being presented to the network. All other values in the output vector should be 0.

The neural network needs 133 inputs and 28 neurons in its output layer to identify the letters.
The network is a two-layer network. The log-sigmoid [33] transfer function at the output layer was picked because its output range (0 to 1) is perfect for learning to output Boolean values. The function generates outputs between 0 and 1 as the neuron's net input goes from negative to positive infinity see Fig.14.



*a = logsig(n)*

Figure 14. Log-sigmoid transfer function.

In building the network, the data was divided randomly into two categories. Training data consisted of 80% of the data (1540 characters"55 samples of each of the 28 character"). The remaining 20% of the data was assigned to the testing data (308 characters"11 samples of each of the 28 character"). Fig.15. shows the network architecture.

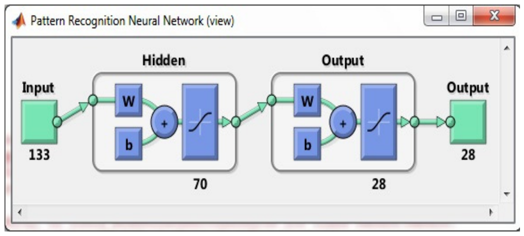


Figure 15. Network architecture

### 2.6.2 Network Training

To create a network that can handle noisy input vectors (characters) it is best to train the network on both ideal and noisy characters.

The network is trained to output a 1 in the correct position of the output vector and to fill the rest of the output vector with 0's.

Back propagation training method is followed. This method was selected because of its simplicity and because it has been previously used on a number of pattern recognition problems. The method uses in this work called principle of gradient descent [34].

This function is useful because the conjugate gradient algorithms have relatively modest memory requirements. Memory is important when working with large networks, and yet it is much faster than other algorithms [35].

The gradient descent updates the network weights and biases in the direction in which the performance function decreases most rapidly, the negative of the gradient. One iteration of this algorithm can be written as:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{g}_k \qquad (10)$$

where $\mathbf{x}_k$ is a vector of current weights and biases, $\mathbf{g}_k$is the current gradient, and $\alpha_k$is the learning rate. This equation is iterated until the network converges. The scaled conjugate gradient algorithm [34] is based on conjugate directions, but does not perform a line search at each iteration.

For evaluating the performance we choose to use the mean square error mse; the average squared error between the network outputs and the target outputs t. It is defined as follows:

$$F = mse = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - a_i)^2 \qquad (11)$$

In this paper we stop the network training is if the sum of squared error falls below 0.001.

### 2.6.3 Testing the Network

After many trials of eliminating, adding and modifying features and also adjusting network hidden layers (start with 20 then 35 …70 neurons) after 70 neurons network performance does not increase. A network of 70 neurons in hidden layer was able to predict about 88% of the input characters correctly (96% in the training phase and 88% in the testing phase).

## 3. RESULTS AND DISCUSSIONS

The test set size used in the experiments is 308 characters (11 different sample of each of the 28 character), Experimental results are presented in Table 2.

Table 2: Recognition rates for Arabic letters in this system.

| Character | Recognition Percentage | Character | Recognition Percentage | Character | Recognition Percentage |
|---|---|---|---|---|---|
| أ | 96% | ز | 80% | ق | 61% |
| ب | 87% | س | 78% | ك | 81% |
| ت | 69% | ش | 88% | ل | 96% |
| ث | 76% | ص | 100% | م | 92% |
| ج | 100% | ض | 72% | ن | 100% |
| ح | 94% | ط | 91% | ه | 97% |
| خ | 100% | ظ | 83% | و | 100% |
| د | 83% | ع | 66% | ي | |
| ذ | 88% | غ | 99% | | |
| ر | 89% | ف | 100% | | |

The achieved results are very promising. Experimental results showed that the proposed method give a recognition rate of about 88% for all letters although we get success rate of 100% for some letters.

As compared to the previous works we find that Aburas [36] system achieved 70% of recognition rate, also Khedher [37] system achieved 73.4% of recognition rate, Taani [38] system achieved 75.3% of recognition rate. We achieved 1% increase in recognition over the Abandah [39] system. All of those systems work on isolated handwritten Arabic characters.

In this work Statistical and structural features were combined from the main body or the boundary with other features to get high recognition accuracy.

These features include the letter form, secondary type (Hamza,dots) and number of those secondary components. The results shown in the previous table illustrate that higher recognition accuracies are achieved using the proposed feature extraction technique. Extracting features from the main body and secondary components provides more valuable features that exploit the recognition potential of the secondary components of handwritten Arabic letters (comparison between س and ش, ص and ض). These results also confirm the importance of the secondary components of the handwritten Arabic letters.

After many trials of eliminating, adding and modifying features and also adjusting neural network hidden layers

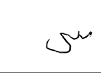
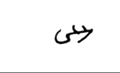
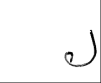
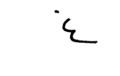
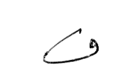
Some notes on the features:

-We tried many kind of features one example is a feature which extracts the Number of holes in the character because a lot of handwritten Arabic letters contain holes which can be very important feature to distinct letters like (ع, ي, ق, س) from others (أ, ب, د) which don't contain holes. Unfortunately this feature doesn't perform well and doesn't give me dependable results because the variation in writing Arabic handwritten characters, so we eliminate this feature from my feature list.

-Another kind of feature which has been examined is secondary component position. Unfortunately because of the variety of secondary components may take different positions and it is highly depending on writing styles, so this feature doesn't give me dependable results so we decide not to use this feature.

Our database of handwritten Arabic samples is CENPRMI [40] dataset. It includes Arabic off-line isolated handwritten characters. The database contains 11620 characters (about 332 samples for each character). These characters were written according to 12 different templates by 13 writers, with each template adopted by 5–8 writers. The 11620 character samples are divided into three groups 198, 67 and 67 for training, validating and testing purposes respectively. Table 3 shows collection of samples of isolated Arabic characters.

Table 3: Variation in characters from dataset

| Letter | Variations | | | |
|---|---|---|---|---|
| ن | | | | |
| س |  | |  | |
| ش | | | | |
| ك |  | | | |
| غ | | |  | |
| ف | |  | | |

As shown above, these samples show that loops are sometime introduced or filled and secondary objects are written in multiple styles, contains loops, under baseline, above baseline cavities, left and right directions, and different shapes and that makes the recognition process very hard.
We found that the Recognition rate is between 100% for easy to recognize letter forms and 61% for the hardest letter forms. Table 4 shows the ten letter forms that have the lowest classification rate.

Table 4: Worst 10 recognized characters

| No | Character | Recognition Rate | Often mistaken as |
|---|---|---|---|
| 1 | ق | 61% | ف |
| 2 | ع | 66% | ح |
| 3 | ت | 69% | ز , ث |
| 4 | ض | 72% | ص , ظ |
| 5 | ث | 76% | ق , ف |
| 6 | س | 78% | ص |
| 7 | ز | 80% | ر |
| 8 | ك | 81% | ح , ع |
| 9 | د | 83% | ك |
| 10 | ظ | 83% | ر , ش |

These ten letters are always drawn with loops or drawn with loops in some writing variations. There are substantial similarities among multiple Arabic letter form groups that have loops. Often the sole difference between such letters is a subtle difference in the loop’s shape. Moreover, letters with secondaries tend to have low recognition accuracies because the variations in drawing the dots or hamzas give inaccuracies in extracting the secondary type feature. Moreover that multiple dots in some writing styles may be isolated (Naskh writing style) or continued dash (Rekaa writing style). After careful examination of the samples that were incorrectly recognized, we concluded that most of these samples are hard to recognize even by a human expert reader. However, we think that the door is open to search for extracting new features that capture subtle differences in loop shapes and secondary types.

## 4. CONCLUSION

This paper presents an approach for extracting features to achieve high recognition accuracy of handwritten Arabic characters.

We present some useful techniques during the preprocessing phase including binarization, normalization and some noise removing methods. Our algorithm extracts useful features not only from the main body, but also from the secondary components of the character. It also overcomes some of handwritten characters variations.

Selecting proper features for recognizing handwritten Arabic characters can give better recognition accuracies, therefore our feature included statistical, morphological and topological features.

Although, there are some challenges with some characters, but the overall recognition rate is perfect especially when compared to other handwritten Arabic characters systems.

After examining the recognition accuracy of each character we found that the recognition rate is between 100% for the easiest recognized characters such as (و, ن and ص) and 61% and 66% for the hardest recognized characters such as (ق and ع), respectively.

We think that this is because those characters contain loops and also they can have different characteristics from one to other writing style. Characters with dots also tend to have low recognition accuracies because the variations in drawing the dots (dots can be linked to the main body of the character or even can be removed in some writing styles) give inaccuracies in extracting the secondary type feature.

Our future work includes increasing the efficiency of the proposed approach especially for the characters that were not recognized well by finding out other powerful feature, also including variations in writing the main body of the character and also the secondaries.

We hope also that we complete system for recognizing handwritten Arabic text passing through segmentation techniques for segmenting the words to characters.

## REFERENCES

[1] M. S. Khorsheed, Automatic recognition of words in Arabic manuscripts, PhD thesis, University of Cambridge, 2000.

[2] N. Arica, Yarman-Vural. F, Optical Character Recognition for Cursive Handwriting, IEEE Trans Pattern Anal Mach Intell, 24(6), 2002, 801–813.

[3] B. Al-Badr and Mahmoud, S. A, Survey and bibliography of Arabic optical text recognition. Signal Process. 41, 1, 1995, 49–77.

[4] A. Nazif, A system for the recognition of the printed Arabic characters. Master's thesis, Faculty of Engineering, Cairo University, 1975.

[5] V. Margner and H. El Abed, Databases and competitions: strategies to improve Arabic recognition systems. In Arabic and Chinese Handwriting Recognition, Lecture Notes in Computer Science, vol. 4768, Springer, 2008, 82–103.

[6] M. Cheriet, Visual recognition of Arabic handwriting: challenges and new directions. In Arabic and Chinese Handwriting Recognition, Lecture Notes in Computer Science, vol. 4768, Springer, 2008, 1–21.

[7] A. Amin, H. Bunke and P. Wang, Arabic character recognition, in Handbook of character recognition and document image analysis, Eds. World Scientific, 1997, 397-420.

[8] S. El-Dabi, R. Ramsis and A. Kamel, Arabic character recognition system: A statistical approach for recognizing cursive typewritten text, Pattern Recognition, Vol. 23, no. 5, 1990, 485- 495.

[9] T. S. El-Sheikh and S. G. El-Taweel, Real-time Arabic handwritten character recognition, Pattern Recognition, Vol. 23, no. 12 , 1990, 1323-1332.

[10] F. El-Khaly and M. A. Sid-Ahmed, Machine recognition of optically captured machine printed Arabic text, Pattern Recognition, Vol. 23, no. 11, 1990, 1207-1214.

[11] A.Sabri, Arabic character recognition using Fourier descriptors and character contour encoding, Pattern Recognition, Vol. 27, no. 6, 1994, 815-824.

[12] J. Ayman and M. Laheeb, Arabic Handwritten Characters Recognized by Neocognitron Artificial Neural Network, University of Sharjah Journal of Pure & Applied Sciences, Vol. 3, No. 2, 2006.

[13] A. Amin, H. Al-Sadoun and S. Fischer, Hand- Printed Arabic Character Recognition System Using an Artificial Network. Pattern Recognition, 29, 1996, 663-675.

[14] A. Amin, Recognition of Hand-Printed Characters Based on Structural Description and Inductive Logic Programming. Pattern Recognition Letters, 24, 2003, 3187-3196.

[15] G. Olivier, H. Miled, K. Romeo & Y. Lecourtier, Segmentation and Coding of Arabic Handwritten Words. Proc. 13th Int'l Conf.Pattern Recognition, 3, 1996, 264-268.

[16] I.S.I. Abuhaiba and P. Ahmed, Restoration of Temporal Information in Off-Line Arabic Handwriting, Pattern Recognition, 26, 1993, 1009-1017.

[17] I.S.I. Abuhaiba, S.A. Mahmoud and R.J. Green, Recognition of Handwritten Cursive Arabic Characters. IEEE Trans. Pattern Analysis and Machine Intelligence, 16, 1994, 664-672.

[18] M. Dehghan, K.Faez, M. Ahmadi and M. Shridhar, Handwritten Farsi (Arabic)Word Recognition: A Holistic Approach Using Discrete HMM. Pattern Recognition, 34, 2001, 1057-1065.

[19] H. Al-Yousefi and S.S. Udpa, Recognition of Arabic Characters. IEEE Trans. Pattern Analysis and Machine Intelligence, 14, 1992, 853-857.

[20] S. Abdelazeem and E. EL-Sherif, Arabic handwritten digit recognition. Int. J. Doc. Anal. Recog. 11, 3, 2008,127–141.

[21] G. Abandah and N. Anssari, Novel moment features extraction for recognizing handwritten Arabic letters. J. Comput. Sci. 5, 3, 2009, 226–232.

[22] K. Sherif and M. Mostafa, A Parallel Design and Implementation For Backpropagation Neural Network Using MIMD Architecture. IEEE, 1996, 1361- 1366.

[23] K. Sherif & M. Mostafa, A Parallel Design and Implementation For Backpropagation Neural Network Using MIMD Architecture. IEEE, 1996, 1472- 1475.

[24] M. A. Ali, Arabic handwritten characters classification using learning vector quantization algorithm. In Image and Signal Processing, Lecture Notes in Computer Science, vol. 5099, Springer, 2008, 463–470.

[25] S. A. Mahmoud and S. O. Olatunji,. Automatic recognition of off-line handwritten Arabic (Indian) numerals using support vector and extreme learning machines. Int. J. Imaging 2, A09, 2009.

[26] S. A. Mahmoud and S. Owaidah, Recognition of off-line handwritten Arabic (Indian) numerals using multi-scale features and support vector machines. Arab. J. Sci. Engin. 34, 2B, 2009, 429–444.

[27] Otsu, N., "A Threshold Selection Method from Gray-Level Histograms," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, No. 1, 1979, pp. 62-66.

[28] G. Lazzara and T. Géraud, Efficient multiscale Sauvola's binarization, Springer-Verlag Berlin Heidelberg, 2013.

[29] Cheriet. M, Character Recognition Systems, John Wiley, 2007, 36-38.

[30] S.L. Jae, Two dimensional signal and image processing, PRENTICE HALL PTR, Upper Saddle River, New Jersey. 07458, 1990.

[31] J. Hann, M.Kamber,. Data Mining: Concepts and Techniques, 3rd Edition. Morgan Kaufman Publishers, 2011.

[32] S. Daniel, K. Vladimír, P. Jiri, Introduction to multi-layer feed-forward neural networks, ELSEVIER, Chemometrics and Intelligent Laboratory Systems 39, 1997, 43-62.

[33] P. CHANDRA, Sigmoidal Function Classes for Feedforward Artificial Neural Networks, Kluwer Academic Publishers, 2003, 185–195.

[34] M. F. Moller, A scaled conjugate gradient algorithm for fast supervised learning, Neural Networks, Vol. 6, 1993, 525–533.

[35] M. H. Beale, T. Hagan and B. Demuth, Neural Network Toolbox 7, User's Guide, by The MathWorks, Inc., 2010.

[36] A. A. Aburas and S. M. Rehiel, Off-line Omni-style Handwriting Arabic Character Recognition System Based on Wavelet Compression, Arab Research Institute in Sciences & Engineering ARISER, 2007, 123-35.

[37] M. Z. Khedher, G. A. Abandah, and A. M. Al-Khawaldeh, Optimizing Feature Selection for Recognizing Handwritten Arabic Characters, World Academy of Science, Engineering and Technology 4, 2005, 81-84.

[38] A. T. Al-Taani and S. Al-Haj, Recognition of On-line Arabic Handwritten Characters Using Structural Features, Journal of Pattern Recognition Research, 2010, 23-37.

[39] G. A. Abandah, K. S. Younis and M. Z. Khedher, Handwritten Arabic Character Recognition Using Multiple Classifiers Based On Letter Form, Conf. on Signal Processing, Pattern Recognition, & Applications Austria, 2008, 128-133.

[40] H. Alamri, J.Sadri, C.Y.Suen, N.Nobile, "A novel comprehensive database for Arabic off line handwriting recognition," The11thInternational Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 664–669. 2008.

## AUTHORS

**Ahmed T. Sahlol** Obtained Bachelor degree from Mansoura University, Egypt in 2004, 2 years Diploma from Mansoura University in 2006, Master degree from Mansoura University, Egypt in 2010. He is currently visiting researcher at Concordia University in Canada and he got a governmental scholarship to study his PhD. His research interests include pattern recognition, optical character recognition and image processing.

**Ching Y. Suen** is the Director of CENPARMI and the Concordia Chair on AI and Pattern Recognition. He has served as the Chairman of the Department of Computer Science and as the Associate Dean (Research) of the Faculty of Engineering and Computer Science of Concordia University. His research projects have been funded by the ENCS Faculty and the Distinguished Chair Programs at Concordia University, FCAR (Quebec), NSERC (Canada), the National Networks of Centres of Excellence (Canada) and the industrial sectors in various countries, including Canada, France, Japan, Italy, and the United States.