Automatic Recognition Of Handwritten Arabic Characters Using Their Geometrical Features

Maged Mohamed Mahmoud Fahmy

Computer Science Department

College of Science

University of Bahrain

Isa Town

STATE OF BAHRAIN

E-mail: mfahmy@sci.uob.bh

Somaya Al Ali

Computer Science Department

Faculty of Science

University of Qatar

QATAR

E-mail: s.alali@usa.net

Abstract:This research aims to use geometrical features and neural networks to automatically recognize (read) off-line handwritten Arabic words. The nature of handwritten Arabic characters and hence the problems that could be faced when automatically (optically) recognizing them are discussed. This research concentrates on the feature extraction process, i.e. extraction of the main geometrical features of each of the extracted handwritten Arabic characters. A complete system able to recognize Arabic-handwritten characters of only a single writer is proposed and discussed. A review of some of the previous trials in the field of off-line handwritten Arabic character recognition is included. The system first attempts to remove some of the variations found in the images that do not affect the identity of the handwritten word (slant correction, slope correction, and baseline estimation). Next, the system codes the skeleton of the word so that feature information about the lines in the skeleton is extracted (segmentation and feature extraction). The features include locating endpoints, junctions, turning points, loops, generating frames (segmentation step), and detecting strokes. These features are then passed on to the recognition system for recognition. The character classification is achieved in this research using a feedforward error back propagation neural network. An 69.7 percent recognition rate has been achieved for the character frames of data.

Keywords: Arabic characters, scaling, segmentation, handwritten characters, feature extraction, and back propagation neural networks.

Dr. **Maged Mohamed Mahmoud Fahmy** is currently working as Assistant Professor, Computer Science Department, College of Science. Previously he worked as researcher at Informatics Research Institute, and Mubarak City for Scientific Research and Technological Application, Alexandria, Egypt. He got BSc. in Electronic Engineering, MSc. in Computer Security from the Alexandria University in 1986. He got Ph.D Engineering in Computer Vision from University of Newcastle-upon-Tyne, the UK, in 1994. He supervised many Master degrees. His fields of interest are in artificial intelligence, computer vision (image processing & neural network), computer networks.

Mr **Somaya Al Ali** took his BSc. in Computer Science from the Qatar University in 1994. Since March 1995 he has worked as assistant at the Qatar University. In 1999 he received the MSc. in Mathematics and Computer Science from the Alexandria University. Since February 2000 he has been preparing a Ph.D at the Department of Computer Science, the University of Nottingham, the UK.

1. Introduction

Off-line handwritten character recognition is the automatic transcription by computer, where only the image of that handwriting is available. Much work has been done on the recognition of English characters, both of separate and cursive scripts (joined). The work of recognizing Arabic characters is still limited. The importance of recognizing Arabic characters is also to be considered by Arabic non-speaking people such as Farisi, Curds, Persians, and Urdu-speaking who use the Arabic characters in writing although the pronunciation is different. Among those are the following. Abuhaiba et al. dealt with some problems in the processing of binary images of handwritten text document [1]. Almuallim and Yamaguchi proposed a structural recognition technique for handwritten Arabic words [2]. A look-up table is used for the recognition of isolated handwritten Arabic characters [3]. Amin et al proposed a new technique for the recognition of handprinted Arabic characters using neural network [4]. Obaid introduced handwritten Arabic characters recognition by neural network [5]. Saleh et al describe an efficient algorithm for coding handwritten Arabic characters [6]. It is worth noting that most of these works are done assuming that a handwritten Arabic word segmenting into separate characters has been done before recognition. This research deals with preprocessing steps (before classification) of off-line handwritten Arabic words. In this system, some of the methods applied to handwritten Arabic writing, such as slant correction, slope correction, thinning, segmentation, and feature extraction have not been

used ever, as seen from the literature survey. The feature extraction process includes locating endpoints, junctions, turning points, loops, generating frames, and detecting strokes. The accuracy could be improved if postprocessing were applied. Further work, as well as suggestions to improve the overall accuracy of the system, are discussed at the end of the context. Before discussing the proposed system, it is necessary to make a quick revision of the nature of handwritten Arabic characters and hence of the problems that could be faced when planning to automatically recognize them.

2. The Nature of Handwritten Arabic Characters and Its Difference from Latin

The main characteristics of Arabic Writing can be summarized as follows:

1) Arabic text (printed or handwritten) is in cursive and in general, from right to left. Arabic letters are normally connected to each other on the baseline.

2) Arabic writing uses letters (which consist of 28 basic letters), ten Hindi numerals, punctuation marks, as well as spaces and special symbols.

3) Some of the Arabic characters are located under the baseline (for example, ر,ز).

4) Some of the Arabic characters consist of two parts (for example ظ,ط,لا).

5) In the representation of vowels, Arabic uses diacritical markings. The presence or the absence of vowel diacritics indicates different meanings in what would otherwise be the same word. If the word is isolated, diacritics are essential to distinguish between the two possible meanings, i.e. (عَلَم ، عَلَّم). If it occurs in a sentence, contextual information inherent in the sentence can be used to infer the appropriate meaning. In this research, the issue of vowel diacritics is not treated, since it is more common to Arabic writing not to employ these diacritics.

(A) (B) (C) (D)

FIGURE 1: DIFFERENT SHAPES OF THE ARABIC LETTER (ع A'IN) IN: (A) BEGINNING, (B) MIDDLE, (C) END, AND (D) ISOLATED POSITIONS.

6) Arabic letter might have up to four different shapes (Figure 1), depending on its relative position in the word that increases the number of classes from 28 to 100 (Figure 2 (a)). For example, the letter (ع A'in) has four different shapes: at the beginning of the word, in the middle of the word, at the end of the word, and in isolation (stand-alone). Furthermore, there are two supplementary characters that operate on vowels to create a kind of stress (Hamza ء) and elongation (Madda ~); the latter operates only on the character Alif (Figure 2(b)). The character Lam-Alif (لا) is created as a combination of two characters, Lam (ل) and Alif (ا), when the character Alif is written immediately after the character Lam. This new character, together with the combination of Hamza (ء) and Madda(~), increases the number of classes to 120 (Figure 2 (a), (b)) [7].

7) Different Arabic characters may have exactly the same shape, and they are to be distinguished from one another only by the addition of a complementary character (position and number of dots associated with it). Hence, any thinning algorithm needs to efficiently deal with these dots without changing the identity of the character (Figure 3). In the segmentation process of a handwritten Arabic word, the characters are more difficult to segment if the dots are not allocated exactly under or above the character body (Figure 4).

8) There is a small number of characters which have the same shape in any position (except for the connecting part (—)), but the position of the character may also differ on the line (on the line, under the line, or up the line). There are some characters which can have different shapes in their position depending on the phonics of the word (e.g. , ئ , ؤ , أ ,ء,إ).

9) The widths and lengths of characters differ from one character to another and from one version to another. Many characters (17 out of 28) are composed of two parts, the body of the character and a number of complementary dots, either above or below the character body. There may be a dot or a group of two or three dots. There are only three characters that represent vowels (ى , و , ا) . Other vowels are represented by diacritics in the form of overscores or underscores. The use of diacritics is limited to cases where the word is from other language. No upper or lower cases exist in Arabic [8].

Name	Isolated	Start	Middle	End
Alif	ا			ـا
Ba	ب	بـ	ـبـ	ـب
Ta	ت	تـ	ـتـ	ـت
Tha	ث	ثـ	ـثـ	ـث
Jeem	ج	جـ	ـجـ	ـج
Hha	ح	حـ	ـحـ	ـح
Kha	خ	خـ	ـخـ	ـخ
Dal	د			ـد
Thal	ذ			ـذ
Ra	ر			ـر
Zay	ز			ـز
Seen	س	سـ	ـسـ	ـس
Sheen	ش	شـ	ـشـ	ـش
Sad	ص	صـ	ـصـ	ـص
Dhad	ض	ضـ	ـضـ	ـض
Tta	ط	طـ	ـطـ	ـط
Za	ظ	ظـ	ـظـ	ـظ
Ain	ع	عـ	ـعـ	ـع
Ghain	غ	غـ	ـغـ	ـغ
Fa	ف	فـ	ـفـ	ـف
Qaf	ق	قـ	ـقـ	ـق
Kaf	ك	كـ	ـكـ	ـك
Lam	ل	لـ	ـلـ	ـل
Meem	م	مـ	ـمـ	ـم
Noon	ن	نـ	ـنـ	ـن
Ha	ه	هـ	ـهـ	ـه
Waow	و			ـو
Ya	ي	يـ	ـيـ	ـي

(A)

Name	Isolated	Start	Middle	End
Alif	آ			ـآ
Alif	أ			ـأ
Alif	إ			ـإ
LamAlif	ﻻ			ـلا
LamAlif	ﻷ			ـلأ
LamAlif	ﻹ			ـلإ
LamAlif	ﻵ			ـلآ
Waow	ؤ			ـؤ
Ya	ئ	ئـ	ـئـ	ـئ

(B)

Figure 2. (A) Arabic Alin all its Forms; (B) Supplementary Characters (Hamza and Madda) and Their Position With Respect To the Alif, waow and ya

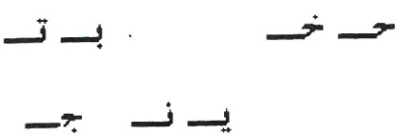
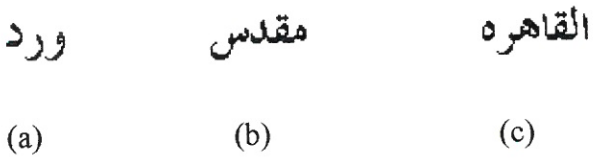


Figure.3. Some of the Arabic Characters That Differ by the Number of Associated Dots

Figure 4. A Handwritten Word That Can Have Segmentation Problems

10) Arabic writing is cursive and words are separated by spaces. Some Arabic characters are not connectable with the succeeding character. Therefore, if one of these characters exists in a word, it divides that word into two subwords. These characters appear only at the tail of a subword and the succeeding character forms the head of the next subword (Figure 5).



(a) (b) (c)

Figure 5. Arabic Word With Constituent Subwords

11) Arabic writing contains many fonts and writing styles. The letters overly in some of these fonts and styles. Furthermore, characters of the same font have different sizes. Hence, segmentation based on a fixed size width cannot be applied to Arabic [8].

3. Off-line Handwritten Arabic Character Preprocessing

This Section describes the operation of the complete preprocessing steps for the handwriting recognition system for a single Arabic word, from the handwriting on a graphics tablet to the output of a segmented word. Any word recognition system can be divided into sections, preprocessing, recognition, and postprocessing, as shown in Figure 6. In this research, the hand- written word is entered by the graphics tablet, and the word is normalized to be presented in a more informative manner at the stage of preprocessing. The recognition is then carried out to recognize the word, and this is done firstly by estimating the data likelihood for each frame of data in the representation, using a suitable classification technique (neural network in our research). Then the postprocessing can be carried out. The system built in this research concentrates on preprocessing operations which are of special importance for the process of Arabic word recognition. The steps involved in preprocessing are implemented in visual C++ code. The proposed system has two main advantages. The first advantage is its ability of dealing with the non-segmented words. The second advantage has to do with exploiting the position of features found in the character or in the sub-character.

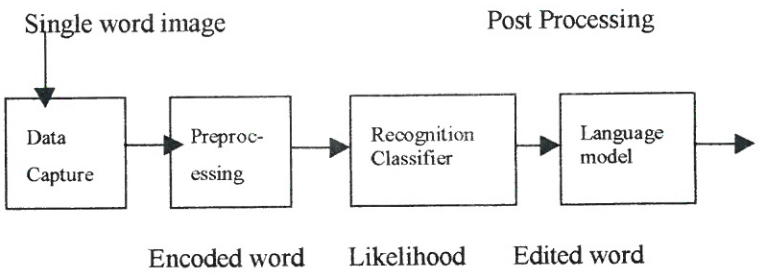


Figure 6. A Complete System for the Off-line Arabic-Handwriting Words Recognition

3.1 Data Capture

The word image acquisition in this system takes place as follows. A single author writes a cursive handwritten Arabic word. That word is written by the complete new 3- button ergonomic refillable inking pen on a Graphics tablet, which has been used essentially for natural free hand drowning, and for filling out forms, both on paper and electronically. The pen weights ultra light, allowing it to be effortlessly held in the palm of the hand. The Arabic words are displayed using a graphics program and then are saved in files, to be used as data for the system.

3.2 Preprocessing

The main advantage of preprocessing a handwritten word image is to organize the information so as to make the task of recognition simpler. The main part of the preprocessing stage is normalization, which attempts to remove some of those variations in the image which do not affect the identity of the word. This system incorporates normalization for each of the following factors:

1) **Stroke Width:** The stroke width depends on the writing instruments used, the pressure of the instrument, and the angle of it with respect to the tablet.

2) **Slant:** This is the deviation of strokes from the vertical axis. It varies with words and writers.

3) **Slope:** The slope is the angle of the base of a word if it is not written horizontally.

4) **Height of the letters:** They vary with authors for the same task, and for a given author for different tasks.

The normalization task will reduce each word image to one consisting of vertical letters of uniform height on a horizontal baseline, and made up of one-pixel-wide strokes. Figure 7 shows the processes involved in preprocessing. In this system, the word image is loaded and cropped. Then the slant and the slope of the word are corrected and thinned. The details of each of the processes are described in the following Sections. The preprocessing operation consists of the following steps:

1) Image Loading

2) Slope Correction

3) Slant Correction

4) Thinning

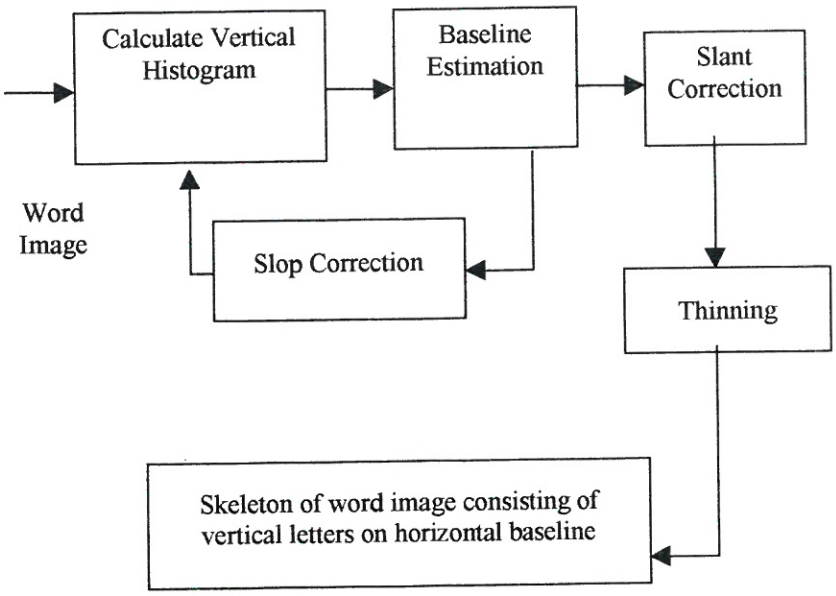


Figure 7. The Preprocessing Operations

In the following Subsections, the steps (processes) involved in the Arabic handwriting preprocessing used in this research are discussed in detail.

3.2.1 Image Loading

For loading the word image, the system uses a ready -made class library [9]. The image object class library is linked to this system and written in Visual C++ language. The image loading procedure consists of three main steps:

1) Load the Image
2) Get the dimension of the image in rectangle form
3) Crop the image to contain only the non-white rectangle

After being loaded the image will be shown on the screen as in Figure 8a, and the dimension of the image word is so calculated and cropped as to contain only the non-white rectangle (Figure 8b).

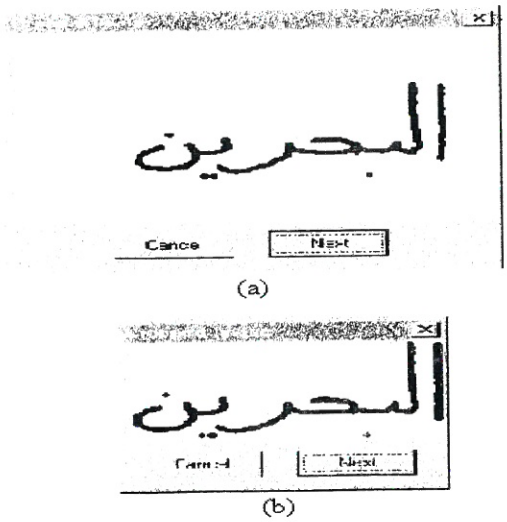


Figure 8. (a) Original Image of the Arabic Word After Loading From A File

(b) Same Image after Being Cropped To Contain Only the Nonwhite Rectangle

3.2.2 Slope Correction

The slope is defined as the angle of the baseline of a word that is not written horizontally. In Figure 10 (a and b), a word is seen before doing slope correction and after its slope being corrected. The character height is determined by intuitively finding the important lines. These lines are the upper baseline and the lower baseline, respectively, with a centerline between the two. With these lines, the ascenders and the descenders that are used by human readers in determining word shape [10], can also be identified. The heuristic, used for estimation of these lines, consists of three main steps [11]:

1) Calculation of the vertical density histogram for the word image
2) Baseline correction
3) Slope correction

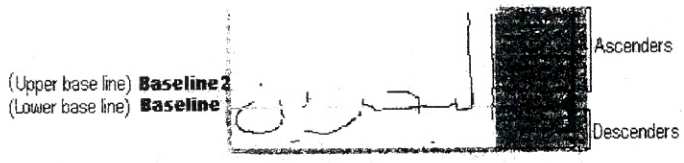


Figure 9. The Vertical Density Histogram of An Arabic Word

The first step is done by counting the number of black pixel in each horizontal line in the image (calculation of the vertical histogram). Then the base line estimation follows by rejecting the part of the image likely to be a hooked descender, as the letter (ﺝ ﺡ ﺥ) Such a descender is indicated by the maximum peak in the vertical density histogram, as illustrated in Figure 9. The second baseline is calculated in the same procedure by finding the second maximum peak in the vertical density histogram. Finally the slope correction procedure goes as follows:

First: *Calculate the slope.* Find the lowest remaining pixel on each vertical scan line.

(1) Retain only the points around the minimum of each chain of pixels, and discard the points that are too high.

(2) Find the line of best fit through these points.

Second: *Slope correction.* The image of the word is straightened to render its baseline horizontal by application of the *Shear transform* parallel to the y-axis.

(3) The baseline, the height, and the bounding rectangle of the cropped image are re-estimated, on the assumption that it is now horizontal.

3.2.3 Slant Correction

The slant is the deviation of strokes from the vertical axis, which varies with words and writers. In Figure 10 (c), a word is seen after correcting its slant.

The slant of a word is estimated by finding the average angle of a near-vertical stroke [11]. This is calculated by finding the edges of strokes, using an edge detection filter. This technique provides a chain of connected pixels representing the edge of strokes. A mode orientation of the edges close to the vertical is used as an overall slant estimate. The procedure of slant correction contains the following steps:

1) Thin the image and calculate all its end points.
2) Find all near vertical strokes by starting at each end point above the baseline until another end point on the baseline.
3) Calculate the average slant for all strokes.

4) By *shear transform* parallel to the axis, the slanted word can be corrected.

5) The bounding Rectangle and width of the image are re-estimated.

3.2.4 Thinning

Numerous algorithms have been proposed for thinning (also called skeletonizing) the plane region. This system uses an algorithm for thinning binary regions described in [12]. The algorithm presents the following aspects: (I) it does not remove end points, (II) does not break connectedness, and (III) does not cause excessive erosion of the region. Region points are assumed to have 1 and background points to have 0. The method applied to the contour points of the given region (a contour point is any pixel of value 1 and having at least one 8-neighbor value 0). The thinning algorithm is as illustrated below:

Input: A digitized image I.

Output: A thinned image I.

Let $N(p_1) = p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8$

{ $N(p_1)$ number of non-zero neighbors of p_1 }

Let $S(p_1)$ = { the number of 0-1 transitions in the ordered sequence of $p_3, p_4, ., p_9, p_2$ }

(1) For every border pixel in the binary region, flag a contour point p for

deletion if the following conditions are satisfied :

(a) $2 \leq Np_1 \leq 6$

(b) $S(p_1) = 1;$

(c) $p_2 \cdot p_4 \cdot p_6 = 0;$

(d) $p_4 \cdot p_6 \cdot p_8 = 0;$

(2) Delete the flagged points.

(3) For every border pixel in the binary region, flag a contour point p for deletion if the following conditions are satisfied:

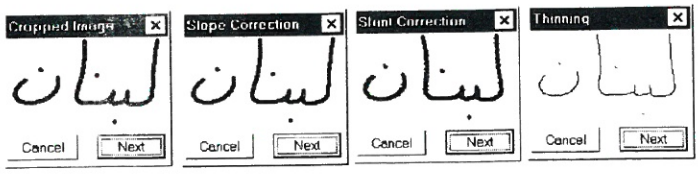
(a) $2 \leq Np_1 \leq 6$

(b) $S(p_1) = 1;$

(c) $p_2 \cdot p_4 \cdot p_6 = 0;$

(d) $p_4 \cdot p_6 \cdot p_8 = 0;$

(4) Delete the flagged points.



(A) (B) (C) (D)

Figure 10. (A) The Word Before the Slope Correction; (B) The Word After its Slope Corrected Horizontally; (C) The Same Word After Slant Correction; (D) The Operation of Thinning

3.3 Finding Handwriting Features

This Section discusses the way for representing the useful information contained in the image of the word. The choice of the feature extraction method limits or dictates the nature and output of the preprocessing step [13]. Since the word in this system is represented by a thinned pattern, or skeleton, the features which are the most suitable for this representation, and hence for the classifier, are used in the system.

Before finding handwriting features for a word, the original word image is normalized and encoded in a canonical form, and so the different images of the same word are encoded similarly. This system uses a skeleton coding scheme, similar to the one used in [11] with something distinctive in the segmentation process of frames due to the difference in Arabic handwriting style, far from that of English.

The word will be segmented into frames. These frames are calculated before thinning. The horizontal density histogram is calculated and smoothed. The maxima and minima of the smoothed density histogram are found, and frame boundaries are defined to be the midpoints between adjacent maximum / minimum pairs. To ensure that the frames do not exceed a certain width, more frames are added where the maxima and minima are far apart, and chosen according to the character height.

Every vertical frame will be segmented into some regions or rectangles. For each of these rectangles, four bins are allocated to represent different line segments, angles with vertical and horizontal, and the line 45 degrees from these.

The performance of the recognizer can be improved by passing to it more information about salient features in the word. A number of useful features can be easily discerned from the processing that has already been performed on the writing: endpoints, junctions, complementary characters, loops, and turning points.

The methods for the detection of intersection points, endpoints, and loops, are all operating on skeletonized bit maps. Thinner class, used in this system programming, has some functions for locating endpoints, junctions and loops. In the next Sections, the strategies for locating endpoints, junctions and loops are mentioned.

3.3.1 Locating Endpoints and Dots

Endpoint is the end/ the start of a line segment. Endpoints are points in the skeleton with only one neighbor and they mark ends of strokes, though some are artifacts of the skeletonization algorithm. Endpoints are found by examining all 1-pixel individuals in the skeletonized bit map image. As a consequence of skeletonization, an endpoint will have one and only one of its 8 contiguous neighbors as an 1-pixel. Therefore, sum up 8 neighbors and recognize an endpoint when the sum is one.

Dots above, or under, the letters (i.e. ب ت ث ج خ ذ ز ش ض غ ف ق ن) can be identified with a simple set of rules. Short, isolated strokes occurring on or above the half-line are marked as potential dots. The number of dots and their location relative to the main skeleton of the character have to be identified in every frame. The number of dots can be one, two, or three. Dots can be below or above the main skeleton of the character. Dots are calculated by tracing every bath from every end point. If tracing reaches another endpoint and the path length is only one pixel or three pixels, the procedure finds a dot. If the path is more than one pixel, the centre of the path is enrolled as dot feature. Then add it to the dots array and erase endpoint feature in that point.

3.3.2 Junctions

Junctions occur where two strokes meet or cross each other and are easily found in the skeleton as points with more than two neighbors. The system will use an algorithm as follows: each of the 1-pixels in the image is examined, and the number n of contiguous 1-pixel to the focus pixel is counted. If the count n of neighboring pixels exceeds 2 ($n \geq 3$), then the focus pixels is considered to be an intersection.

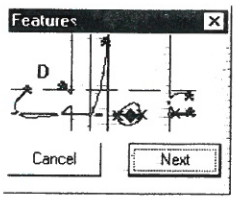


Figure 11. The Word With the Features on It, Endpoints Represented With ⋆, Intersections With X, Loops With ♦, and Dots With D

3.3.3 Turning Points

Points where a skeleton segment turns from upward downward are recorded as top turning points. Similarly, left, right, and bottom turning points can be found as illustrated in Figure 12. Turning points are detected by multiple fixed windows, to examine the variation in coordinate values of the start, the mid, and the endpoints of the curve.

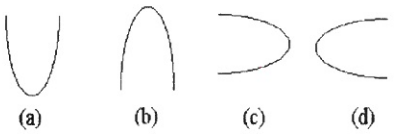


Figure 12. Four Turning Points in Different Directions, (a) Up, (b) Down, (c) Left, and (d) Right

3.3.4 Loops

Loops are found during a connected-component analysis of the smoothed image, to find areas of background colour not connected to the region surrounding the word. A loop is coded by a number that represents its area. Loops are detected by a comparison with previously drawn up membership lists. The process begins by cutting off all edges terminating in endpoints that form the character. If no 1-pixels remain, there are no loops and the process is complete. Otherwise, at least one loop is present. The program first builds a list, Q0, of all 0-pixels. Each pixel I Q0 is within some loop. We may remove an arbitrary pixel from Q0, and build a collection of all its 0-pixel neighbors that do not cross a boundary formed by 1-pixels. This collection is the membership for a single loop, L. All of the L member pixels are removed from Q0. The process continues until Q0 is empty and all loops within the character have been detected.

3.3.5 The Generation of Frames

The position of complementary characters, endpoints, turning points, loops, and junctions are useful, and they are recorded along with line-segment features for each horizontal strip. For dots, it is only useful to know whether they are above or below the baseline. Thus for each of the three segments in the frame, eleven features are encoded (four line-segment angles, four turning points, junction, endpoint and loop). And a two-dot feature for each frame.

The generation of frame procedure contains six main steps to be followed in the segmentation of an Arabic word:

1. Calculate the horizontal histogram for the word image (after completing the preprocessing stages) as shown in Figure 13
2. Find the maximum peak in the horizontal histogram
3. Find the minimum peak in the horizontal histogram

4. Make the frames by adding the minimum and maximum peaks in the same array (Frame Array)
5. Check for shortframes if any, and remove them.
6. Check for long frames if any, and segment them.

After this stage, the image of the word will be segmented into frames as shown in Figure 11. In this system, every frame in the image will be distributed into three segments. Each one can be identified as playing a definite but distinct role in the representation of handwriting. The region between the upper baseline and the lower one, both containing most of the horizontal movement in a word and capturing important data about short vertical strokes, makes up the majority of Arabic letters. Two more segments for the ascenders and descenders are found in the region above the upper baseline and below the lower baseline. In the end, the word image is segmented into frames and segments, so the previous features in these frames and segments could be distributed.

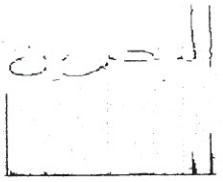


Figure 13. Horizontal Histogram and Segmentation of Words into Frames

3.3.6 Detect Strokes

The purpose of this process is to detect strokes in each segment. The strokes or line segments can be vertical, horizontal, and the lines 45 degrees from these. Within this framework, the lines of the skeleton image are *coarse coded* as follows. The one-pixel-wide lines of the skeleton are followed, and wherever a new grid is entered, the section in the previous box is coded according to its angle. Segments, which are not perfectly aligned to the discrete angles, are then coded according to the closest angle [14]. After the detection of all features in the word, the features should be drawn on the screen as illustrated in Figure 11.

4. Character Classification

The character classification is done in this research using feedforward error back propagation neural network. The network has a single hidden layer of standard perceptions with nonlinear activation functions. The mapping process is from input, represented by features extracted for the Arabic character, to the output, that represents an indication to that character. This mapping process is modeled in terms of some mathematical functions. The function contains a number of adjustable parameters whose values are determined with the help of the data. Such functions could be written as follows:

$$\mathbf{o_i} = \mathbf{o_i}(\mathrm{I}, \mathrm{w})$$

where w denotes the vector of parameters. The output $\mathbf{o_{k,j}}$ of a unit is a function of the inputs $\mathbf{p_{k,j}}$ and the network parameters (the weights of the links $\mathbf{w_{kj}}$ with a bias $\mathbf{b_k}$)

$$O_i = f_i(\{\sigma_j\}),$$
$$\sigma_i = b_i + \sum a_j w_{kj}$$

The network is fully connected— that is each input is connected to every output. The input units accept one frame of parameterized input. The output units estimate letter probabilities for each of the character classes. The feedback units have a standard sigmoid activation function

$$f(\sigma_i) = (1 + e^{-\sigma_i})^{-1},$$

The character outputs have a " softmax " function

$$f_i(\{\sigma_j\}) = \frac{e^{\sigma_i}}{\sum_j e^{\sigma_j}}.$$

4.1 Neural Network Architecture

The backpropagation neural network shown in Figure 14 has three layers; each input layer node is connected to each middle layer node and each middle node is connected to the output layer nodes. A formalization of the backpropagation neural network topology used in this research is shown in the following:

Topology $\mathbf{T = (F,L)}$

Interconnecting Linkage $\mathbf{L = \{ w_{i,j} \rightarrow k,l \}}$, where

back propagation neural network is fully connected.

Framework $\mathbf{F = \{ c_0, c_1, c_2 \}}$, where

c_0, c_1, and c_2 represent the input, the middle, and the output layers respectively.

Cluster (layer) $\mathbf{c_i = \{n_{i,j}\}}$

(Input Layer) $\mathbf{c_0 = \{n_{0,0}, n_{0,1}, n_{0,2}, \ldots, n_{0,33}\}}$

(Middle Layer) $\mathbf{c_1 = \{n_{1,0}, , n_{1,1}, n_{1,2}, \ldots, n_{1,60}\}}$

(Output Layer) $\mathbf{c_2 = \{n_{2,0}, , n_{2,1}, n_{2,2}, \ldots, n_{2,120}\}}$

At the beginning, the network had 33 neurons in the input layer and 60 neurons in the middle layer. In the output layer, there should be 120 neurons since it is required to classify 120 handwritten Arabic letters (in all their forms), and 120 output variables, each of which corresponds to one of the possible letters, might be considered. In general it will not be possible to determine a suitable form for the required mapping, unless there is a data set of examples. After analyzing the input data set and different possible outputs, it is found that 53 neurons at the output layer will do. The number of output neurons is 53, that is enough to represent different stroke versions resulted from different character frames (instead of having 120 outputs).

The recognition process has known two trials. The neural network has three layers. In both trials, the number of output neurons was 53 and the number of hidden neurons was 60. In the first trial, the number of inputs is 35, representing 11 features for each of the three segments of a character, plus two inputs representing dot(s). One of these two inputs represents dot(s) if they are above the baseline, while the other input represents dot(s) if they are below the baseline. The achieved accuracy is 53%. In the second trial, the number of inputs is 37, with two other features of each segment incorporated. Each of these two inputs has a value 0.3, if there is only one dot, 0.6 if there are two dots, and 0.9, if there are three dots. The recognition accuracy increases to 69.7%. Table 1 shows the target letters or sub-letters with the total number used for both training and testing the network.

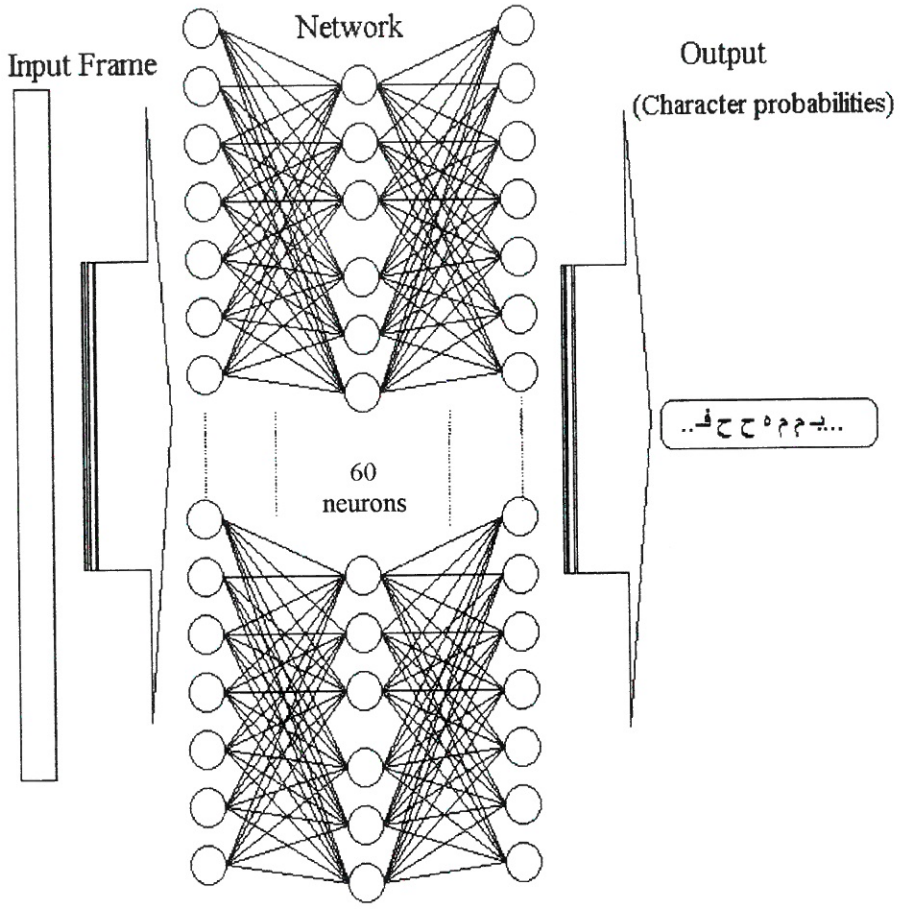


Figure 14. The Topology of Back Propagation Neural Network Used in this Research

4.2 Neural Network Training and Testing

The network is trained standard connectionist techniques, in particular the generalized delta rule [9]. This determines the way weights should be changed to improve the network approximation to a set of desired target outputs. For training, target values must be given, which the network output can be compared with. Target values are given for each frame of data manually, indicating the correct class, the class for which the network output should be one, all the others being zero. A single writer had written 300 different Arabic words and after that he was asked to write them again (which results in a data set of 600 words). The neural network was trained initially by 530 parts of characters (character frames). These frames of characters have been segmented from 300 handwritten words. The network was once again tested with another 530 parts of characters.

Table 1. The Data Set Used To Train and Test the Classifier in this System

Mis classifi	No. of testing patterns	Number of patterns	Character	Character number	Mis classif	No. testing patterns	Number of patterns	Character	Character number	Character
					5	92	185	ا	1	Alif
5	12	24	ب	4	4	24	48	ب	3	Ba
4	16	33	ت	8	1	6	13	ت	7	Ta
3	5	10	ث	12	1	9	19	ث	11	Tha
2	16	32	ج	16	2	12	24	ج	15	
3	11	23	ح	20	3	12	25	ح	19	Hha
3	6	12	خ	24	2	10	20	خ	23	Kha
5	37	75	س	36	1	18	37	س	35	Seen
0	4	8	ش	38	4	9	18	ش	37	Sheen
5	5	11	ص	40	4	5	11	ص	39	Sad
2	5	10	ض	43	0	4	8	ض	42	Dhad
5	10	20	ط	44				ط		Tta
3	6	11	ظ	46				ظ		Za
11	13	25	ع	49	3	15	30	ع	48	Ain
1	4	8	غ	53	2	4	8	غ	52	Ghain
3	7	14	ف	57	2	5	11	ف	56	Fa
2	8	16	ق	60	3	13	26	ق	59	Qaf
5	12	23	ك	62	4	10	19	ك	61	Kaf
0	15	29	ل	64	6	74	148	ل	63	Lam
2	30	60	م	68	3	31	61	م	67	Meem
27	38	75	ن	72	1	11	23	ن	71	Noon
0	6	12	ه	76	0	5	11	ه	75	Ha
20	49	98	ي	82	5	35	71	ي	81	Ya
10	86	141	ا	2						Alif
13	15	29	ب	6	10	22	44	ب	5	Ba
2	4	8	ت	10	2	4	8	ت	9	Ta
1	8	17	ث	14	3	4	9	ث	13	Tha
2	6	12	ج	18	2	6	11	ج	17	
1	5	10	ح	22	2	2	4	ح	21	Hha
1	4	8	خ	26	1	4	8	خ	25	Kha
1	8	15	د	28	5	15	31	د	27	Dal
1	7	14	ذ	30	4	8	16	ذ	29	Thal
1	19	38	ر	32	3	61	123	ر	31	Ra
1	6	12	ز	34	1	5	9	ز	33	Zay
1	31	63	U	41				ص		Sad
			ط		3	5	10	ط	45	Tta
			ظ		5	9	19	ظ	47	Za
1	4	8	ع	51	0	4	9	ح	50	Ain
0	5	10	غ	55	1	4	8	خ	54	Ghain
3	6	12	"ف	58				ف		Fa
			ق					ق		Qaf
			ك					ك		Kaf

Note that in this Table:

Column 1 (and 6) is the number of misclassified character patterns.

Column 2 (and 7) is the number of character patterns used for testing the network.

(The same number is used for training the network).

Column 3 (and 8) is the total number of character patterns.

Column 4 (and 9) represents the Arabic character.

Column 5 (and 10) represents the character number.

Column 11 represents the Arabic character pronunciation.

5. Experimental Results and Discussion

This research deals with Arabic character recognition by benefiting the position of the characters in a word. The trip to recognizing Arabic characters that have been extracted from words includes several stages. It starts with processing the slanted words and ends with getting the features of segmented characters. Some of the preprocessing techniques involved are being implemented in Arabic handwriting, to make it invariant to some of the distortions affecting handwriting. The features extracted for each character are entered as inputs to a back-propagation neural network for recognition.

A single writer wrote 300 different Arabic words and after that he was asked to write them again (results in a data set of 600 words). The slant and slop correction was done to each word. Some words are not corrected properly because the characters of the word are not written in the same direction, the baseline for those could not be calculated. That case might happen in particular words that contains character (*kaf* in their middle, ـكـ), like (هكذا). This problem can be overcome in calculating the baseline for lines of several words. Those words were deducted from the data set. Half of the data set was used (after preprocessing) for training neural network while the other half was used for testing it.

The neural network classifies the characters on the basis of the features. Feature classification must be optimal in order to extract distinct primitives and to correctly identify the character. However, the hand-printed characters tended to have *hairs,* that created problems in generalizing the primitives. Another problem encountered with during feature extraction was fixing the direction of curves.

The segmentation was the most problem generating part: the word is segmented into sub-characters (frames) by calculating the histogram, and then each frame is segmented again into more segments. By trial, it was found that if the frame contained a larger part of the target character, the result would have been more accurate. On initiating the experiment, the frames were segmented into five parts, that reflected the position of the features in the word based on baseline positions. But the result was not encouraging. The number of segments in each frame was decreased to three, and the neural network was trained again.

Some of the missclassified characters have wrong dots number or location in the character. The deduction of these frames from the data set used for training and testing the neural network would certainly improve the recognition process accuracy.

6. Conclusion

This research proposes a technique for the recognition of handwritten Arabic characters using geometrical features extraction and neural networks. There are four main advantages of this technique. First, the proposed technique combines rule-based (structural) and classification tests. Second, it is more efficient for large and complex sets, such as Arabic characters. Third, feature extraction is inexpensive. Fourth, the extraction time is independent of the character size. Preprocessing techniques have been described including segmentation of word images, to give invariance to scale, slant, slope, and stroke thickness. Representation of the image is discussed and the skeleton and stroke features used are described. A backpropagation neural network is used to estimate probabilities for the frames of characters represented

in the skeleton. This analysis exploits the position of a character in the word, and differs from previous ones in the field of off-line handwritten Arabic character recognition by dealing with recognition as a whole process. The process contains several steps that cannot be separated from each other (i.e. preprocessing, segmentation, feature extraction, and classification). An 69.7 percent recognition rate has been achieved for the character frames of data. Comparison of results, obtained in this research, with other researches' is difficult to make. There is no objective measure of good handwriting because of differences in experimental details, the actual handwriting used, the method of data collection, and of dealing with a hale off-line handwritten Arabic word. Single-author error rates for systems applied to handwriting in English (with post-processing) include 87 percent [11], 52 percent [16], 50 percent [17], and 30 percent [18]. Even in human recognition of an average handwriting, an estimated misclassification rate of 4% is unavoidable [5]. Moreover, humans usually use context as the main discriminator. This means that having an 100% recognition rate of handwritten characters without having the contextual information is near to impossible.

7. Future Work

Although "completely" automated off-line handwritten Arabic document recognition is not foreseeable in the near future, this research as well as other researches in the field of off-line handwritten character recognition, are only steps towards.

The future work that can be done, relative to this research, cannot be counted. At each step in this research, there can be a developing work. The following are suggestions to improving the performance of each step involved in handwritten Arabic character recognition.

At the Preprocessing step any contextual information that can be obtained before the segmentation stage is helpful. For example in characters (ذ, ـذ), (د, ـد) and (ﻗ, ـﻗ) the first character cannot be connected from the left side, and the second at the end of it. In (ـﻋ, ﻋ), the first character must be at the beginning of a word (or subword) and the second is at the end.

At the segmentation step, a step generating lots of problems, the segmentation can be more accurate if some rules (constraints) are established. Segmenting no loop, i.e. the line which represents the edges of frames, should not cut any loop. That will overcome the problem of concatenating characters like (ﻫ, ﻫـ, ﻣـ, ﻣﺼ, ﺿ, ﻗ, ﻇ, ﻇـ, ﻓـ, ﺣـ).

At the Feature Extraction step, adding different features topology like Fourier Descriptor (FD) may increase the recognition rate. Change the dot(s) features to three features indicating their relative position to the body of the characters. Stroke feature can be added for a more accurate recognition of characters like (ﻃ, ﻇ, ﻛ).

During the recognition process, more samples of frames can be added to increase the recognition rate. Finally, a post-processing operation can be done using a hidden Markov model (HMM), to combine the data likelihood of frames of characters, which makes the best choice of words for the observed data.

REFERENCES

1. ABUHAIBA, HOLT, M. and DATTA, S., **Processing of Binary Images of Handwritten Cursive Arabic Characters**, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (PAMI), Vol.16, No.6, June1994.
2. ALMUALLIM, H. and YAMAGUCHI, S., **A Method of Recognition of Arabic Cursive Handwriting**, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (PAMI), Vol. 9, 1987.

3. SAADALLAH, S. and YACU, S., **Design of An Arabic Character Reading Machine**, Proceedings of Computer Processing of Arabic Language, Kuwait, 1985.

4. AMIN, AL-SADOUN, H. and FISCHER, S., **Hand Printed Character Recognition System Using Artificial Network**, PATTERN RECOGNITION, Vol.29, No.4, 1996.

5. OBAID, A.M., **Arabic Handwritten Character Recognition by Neural Nets**, JOURNAL ON COMMUNICATIONS, Vol.45, August 1994.

6. SALEH, A., **A Method of Coding Arabic Characters and Its Application to Context Free Grammar**, PATTERN RECOGNITION LETTERS, Vol. 15, No. 12, 1994.

7. NAZIF, A., **Assistant for the Recognition of the Printed Arabic Characters**, Master Thesis, Faculty of Engineering, Cairo University, 1975.

8. BUNKE, H., WANG, P.S. and BIRD, H.S., **Handbook of Character Recognition and Document Image Analysis**, WORLD SCIENTIFIC, Singapore, 1994.

9. LEINECKER, R.C., **Visual C++ 5 Power ToolKit**, VENTANA, USA, 1997.

10. BOUNA, H., **Visual Recognition of Isolated Lower Case Letters**, VISION RESEARCH, Vol. 11, 1971, pp. 459-474.

11. ANDREW, W. SR, and ROBINSON, A.J., **An Off-line Cursive Handwriting Recognition System**, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, Vol. 20, No.3, March 1998.

12. RAFUEL, C. and WOODS, R., **Digital Image Processing,** ADDISON -WESLEY, USA, 1992.

13. ABHIJIT, S. P. and MACY, R. B., **Pattern Recognition With Neural Networks in C++**, CRC PRESS, 1995.

14. CASEY, R.G. and LECOLINET, E., **Strategies in Character Segmentation: A Survey,** International Conference on Document Analysis and Recognition, Vol. 2, August 1995.

15. RUMELHART, D.E., HINTON, G.E. and WILLIAMS, R .J., **Learning Representations By Back-Propagating Errors**, NATURE, Vol. 23, 1986, pp. 533-563.

16. BOZINOVIC, R.M. and SRIHARI, S.N., **Off-Line Cursive Word Recognition**, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (PAMI), Vol. 11, January 1989.

17. EDELMAN, S., ULLMAN, S. and FLASH, T., **Reading Cursive Script By Alignment of Letter Prototypes**, INTERNATIONAL JOURNAL ON COMPUTER VISION, Vol. 5, No. 3, 1990, pp. 303-331.

18. YANIKOGLU, B.A. and SANDON, P.A., **Off-Line Cursive Handwriting Recognition Using Style Parameter**, PCS-TR93-192, Dartmouth College,. 1993.