Pergamon

Pattern Recognition, Vol. 27, No. 6, pp. 815–824, 1994
Elsevier Science Ltd

0031-3203/94 $7.00 + .00

0031-3203(93)E0028-6

ARABIC CHARACTER RECOGNITION USING FOURIER DESCRIPTORS AND CHARACTER CONTOUR ENCODING

SABRI A. MAHMOUD
Computer Engineering Department, College of Computers and Information Sciences, King Saud University, P.O. Box 51405, Riyadh 11543, Saudi Arabia

(*Received* 25 *January* 1993; *in revised form* 24 *November* 1993; *received for publication* 2 *December* 1993)

Abstract—Normalized Fourier descriptors are known to be invariant to scale, translation, and rotation. This technique was used by researchers of Latin OCR yielding acceptable results. In addition, contour analysis was used in object recognition with success. Both techniques are adopted as they are necessary for the recognition of Arabic characters with acceptable recognition rates. This combination was deemed necessary due to the special characteristics of Arabic characters that have some very similar characters. The character images are smoothed by a statistically-based algorithm to eliminate noise. Then, the contours of the image (namely the character primary part, the dots, and hole contours) are extracted. Fourier descriptors and curvature features of the primary part of the character are computed. These features of the training set are used as the model features. 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. The dots' and holes' features are then used to specify the particular character. Experimental results have shown that the combination of the Fourier descriptors, the curvature features and the use of dots' and holes' features to be powerful in successfully classifying Arabic characters. Recognition rates of 100% were achieved for the model classes. However, this rate has come down to 98% in the post-recognition phase of identifying the specific characters. The major part of these errors come from corrupted data.

Arabic character recognition OCR Fourier descriptors Contour analysis
Curvature features Direction features

1. INTRODUCTION

Character recognition is a pattern recognition application with the ultimate aim of simulating the human reading capabilities of both machine-printed and handwritten cursive text. The systems currently available may read faster than humans, but cannot reliably read such a wide variety of text nor consider context. One can say that a great deal of intensive effort is needed to, at least, narrow the gap between humans' and machines' reading capabilities. Optical character recognition technology has many practical applications that are independent of the treated language. Entering data into commercial data processing applications, postal address reading and sorting, and as a reader for handwritten and printed postal codes, as a general data entry for the automation of the work of an ordinary office typist, are a few examples.

Most of the proposed work on optical character recognition has been on Latin or Chinese characters. An early attempt in the area of Latin character recognition was made in 1959[1] although some researchers trace the modern version of OCR back to the middle of the 1940s with the development of digital computers.[2] One of the early attempts of Chinese character recognition was made in 1966.[3] Arabic character recognition, in contrast, has started recently with a relatively slow advance. The first published work, that culd be traced, appeared in 1980.[4] The main cause for this lag is mainly due to the special characteristics of Arabic text and lack of adequate support.

Nouh *et al.*[4] suggested a standard Arabic character set, in order to facilitate the computer processing of Arabic characters. The disadvantage of the proposed system is the assumption that the incoming characters are generated according to a specified standard rules. Khaly and Sid-Ahmed[5] presented a recognition system capable of recognizing machine-printed Arabic text. Their segmentation algorithm requires the Arabic cursive words to be thinned in advance. The segmented characters are then recognized using moment invariant descriptors. This system is limited to specific fonts and involves thinning of the text which demands long processing time. In addition, the result of thinning may not be unique. Al-Yousefi and Udpa[6,7] used a statistical approach for recognizing handwritten Arabic characters. Secondary parts of a character are first isolated, and the moments of the horizontal and vertical projections of the primary part of the character are then computed. The system requires long processing time. Ramsis *et al.*[8] and El-Dabi *et al.*[9] used the accumulative invariant moments as feature descriptors. They used the width of the smallest character in the set as the minimum character width. Then the moments are calculated and checked against the feature vector. If the character is rejected, the width is adjusted by

adding another column, and the moments are recalculated and checked. The process is repeated until the character is recognized. The system is font-dependent and requires heavy computations due to computation of the moments for different character widths, and the sensitivity to the smallest variation in the input patterns. A system for the recognition of Hindu numerals is developed by Abdelazim and Hashish.[10] The system is based on modeling each numeral using hidden Markov chains. The recognition is accomplished by computing probability scores using the Viterbi algorithm which is borrowed from speech processing. A tree classifier based on geometrical features of the Hindu numerals is used. The system is problem-dependent and the extension of the technique to Arabic characters is expected to have higher error rates and be time consuming. Abdelazim *et al.*[11] used a partial observation of the character taking vertical columns of pixels one by one. The segmentation of Arabic cursive words into characters follows the recognition of character. Most of the Arabic characters are recognized seeing only 40% of the character. The algorithm is based on computing the accumulative probability distribution of the observed columns using a code book of 26 vectors representing all possible column vectors of the Arabic characters within some threshold. The technique is font-sensitive and the presence of noise, scaling or rotation results in column vectors that are not in the code book and hence increases the error of the system. Al Muallim and Yamaguchi[12] developed a method for the recognition of Arabic cursive handwritten text. Cursive words are first segmented into strokes. These strokes are then combined into a string of characters that represent the recognized word. The system failure was due to both wrong segmentation of words and wrong classification of strokes. The system of Goraine *et al.*[13] segments text into individual characters using principal and secondary strokes. The characters are then classified according to the stroke position and shape, and the existence of loops in the stroke. The system was tested on both printed and handwritten text and it involves thinning the text in advance. Consequently, a high processing time is required and the recognition accuracy is reduced due to the non-uniqueness of the thinning process. El-Gowely *et al.*[14] introduced a system for the recognition of a multi-font photo script Arabic text. The proposed system is composed of three phases. The text is first segmented according to a set of predefined rules. The output is then passed to a preliminary classification phase that labels the unknown characters into one of ten possible classes based on height, width, position of the character with respect to the baseline, and presence of dots. The last and third phase is the character recognition phase which is based on utilizing the geometrical features of the Arabic characters. Amin[15] used a technique based on Viterbi algorithm and hidden Markov models for Arabic printed text recognition. The system segments words into characters, recognizes the characters, and then uses a dictionary to recognize words. A recognition rate of 90% is reported which is too low for practical purposes and the system is slow. In addition, the system suffers from inherent ambiguity and deficiencies.

In this paper, an Automatic Arabic Character Recognition system is presented. The system recognizes segmented typewritten characters. The presented system is composed of several stages. The first stage is the smoothing of the input characters. The noise due to the writing or printing processes and quantization noise of the scanning device are eliminated or reduced considerably by using a statistically based algorithm. Then the system extracts features of the closed contours of the character primary part. A boundary line encoding technique which encodes the boundary of a character as a sequence of curved or line segments, and Fourier descriptors that describe the boundary by a set of numerical features are used. Ten Fourier descriptors, 16 direction and direction length features, 20 concave and convex curvature features, dots, and holes features are computed. The next stage is the training (or modeling) stage of the classes. Then comes the recognition (testing) stage of the new data. Theoretical and experimental evidence available in the literature indicate that a Fourier descriptors based technique is a powerful technique for classifying closed contours.[16-22] In character recognition the combination of Fourier descriptors and boundary line encoding features may reinforce the discriminating power of the recognition system[18] (especially for Arabic characters where variable shapes for the same character are involved).

The organization of the paper is as follows: Section 2 states the characteristics of Arabic language. Smoothing of character images is addressed in Section 3. Feature extraction including chain encoding, Fourier descriptors, boundary line encoding where concave and convex features are extracted, and dots and holes features are addressed in Section 4. The recognition phase including training and recognition stages is addressed in Section 5. Experimental results are analysed in Section 6. Finally, conclusions are presented in Section 7.

2. CHARACTERISTICS OF ARABIC TEXT

Arabic text has the following characteristics that are, in general, different from other languages like Latin or Chinese, but is similar to Persian and Urdu:

(1) Arabic text is written from right to left.

(2) Arabic text is written cursively in both machine printed or handwritten text.

(3) Arabic language contain 28 basic characters. Each character has different shapes (i.e. from two to four different shapes) depending on its position within a word. For example, the letter AIN (ع) has the following shapes: at the beginning (عـ), in the middle (ـعـ), at the end (ـع), or isolated (ع).

(4) In addition to the 28 main letters, Arabic language has additional characters such as TAA_MARBOUTAH (ة), LAM_ALF (لا).

(5) Arabic words may consist of one or more subwords. Every subword may have more than one

character, because some Arabic characters are not joinable to others from the left side. As an example, the word (كتاب) consists of two subwords: (كتا) which consists of three characters, and (ب) which is a single character.

(6) Sixteen of the Arabic characters have a single, a couple or triple dots, or a zigzag. The dots and zigzags are called secondaries and they are located above the character primary part like ALEF (ا), or below like BAA (ب), or in the middle like JEEM (ج).

(7) Several Arabic characters have exactly the same primary part. However, they are distinguished from each other by the addition of dots in different locations. For example, the three characters BAA (ب) TAA (ت), and THAA (ث) have a common primary part. However, character BAA (ب) has a single lower dot, TAA (ت) has two upper dots, and THA (ث) has three upper dots.

(8) The use of special stress marks called diacritics is another characteristic of the Arabic language. Diacritics such as Fat-hah (ـَ), Dhammah (ـُ), Shaddah (ـّ),Maddah (ـٓ), and Kasrah (ـِ) may change the meaning and/or the pronunciation of a given word. Moreover, Tanween may be formed by combining couples of Fat-hah (ـً), Dhammah (ـٌ), and Kasrah (ـٍ).

(9) Arabic characters do not have a fixed size (height and width). The character size varies according to its position in the word, for example, the character AIN (ـعـ, ـعـ, ـع, ع).

(10) An important feature of Arabic writing is the presence of a writing line (base line). The base line is a horizontal line where the characters' connection segments are located. The base line has the maximum number of object pixels.

Figure (1a) shows a list of the Arabic characters in their different writing forms according to their positions in the word. The marked boxes are those of the model classes. Figure (1b) shows a list of Arabic language diacritics. Other diacritics may be formed by combining two of these diacritics. For example, FAT-HAH, DAMMAH, and KASRAH may be combined with SHADDAH to give (ـّـَ), (ـّـُ), and (ـّـِ), respectively.

3. SMOOTHING

During the digitization process some spurious pixels may result in the character image. These pixels are noise pixels that add irregularities to the outer boundary of the characters and may have undesired effects on the recognition system. A statistically based smoothing algorithm for the smoothing of character images is used. This algorithm reduces the noise of a binary image by eliminating small areas and filling little holes, that results in the regularization of the character contour.[9] This simple and efficient technique is based on a statistical decision criterion. Given a binary image of an Arabic text, the algorithm modifies each pixel according to its initial value and to those of its neighborhood. The rules are stated as follows.

If $P_0 = 0$ then

$$P'_0 = \begin{cases} 0, & \text{if } \sum_{i=1}^{8} P_i < T \\ 1, & \text{otherwise} \end{cases}$$

Name	End	Middle	Beginning	Isolated
ALEF	* ـا			* ا
BAA	* ـب	* ـبـ	* بـ	* ب
TAA	ـت	ـتـ	تـ	ت
THAA	ـث	ـثـ	ثـ	ث
JEEM	ـج	ـجـ	جـ	ج
HAA	* ـح	* ـحـ	* حـ	* ح
KHAA	ـخ	ـخـ	خـ	خ
DAAL	* ـد			* د
THAAL	ـذ			ذ
RAA	* ـر			* ر
ZAIN	ـز			ز
SEEN	* ـس	* ـسـ	* سـ	* س
SHEEN	ـش	ـشـ	شـ	ش
SAAD	* ـص	* ـصـ	* صـ	* ص
DHAD	ـض	ـضـ	ضـ	ض
TTAA	* ـط	* ـطـ	* طـ	* ط
TTHAA	ـظ	ـظـ	ظـ	ظ
AIN	* ـع	* ـعـ	* عـ	* ع
GHAIN	ـغ	ـغـ	غـ	غ
FAA	* ـف	* ـفـ	* فـ	* ف
QAAF	* ـق	ـقـ	قـ	* ق
KAAF	* ـك	* ـكـ	* كـ	* ك
LAAM	* ـل	* ـلـ	* لـ	* ل
MEEM	* ـم	* ـمـ	* مـ	* م
NOON	* ـن	ـنـ	نـ	* ن
HHAA	* ـه	* ـهـ	* هـ	* ه
WAW	* ـو			* و
YAA	* ـي	ـيـ	يـ	* ي

Fig. 1(a). Arabic character set used. The cells with * are those of the models.

FAT-HAH	DAMMAH	KASRAH	SUKOON	SHADDA	MADDAH
ـَ	ـُ	ـِ	ـْ	ـّ	ـٓ

Fig. 1(b). Diacritics of Arabic text.

P4	P3	P2
P5	P0	P1
P6	P7	P8

Fig. 2(a). The current pixel P0 and its neighbors.

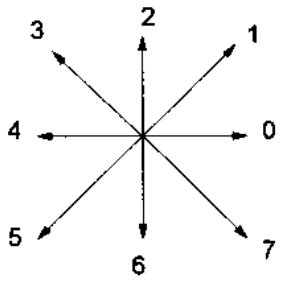

Fig. 2(b). Freeman chain code.

else

$$P'_0 = \begin{cases} 1, & \text{if } p_i + P_{i+1} = 2 \text{ for at least one } i = 1, \ldots, 8; \\ 0, & \text{otherwise.} \end{cases}$$

where P_0 is the current pixel value, P'_0 the new pixel value, and T the threshold.

A threshold of 5 was found, experimentally, to yield acceptable results. A lower threshold results in filling character holes and concave boundaries consequently changing the topology of the characters. Higher thresholds result in a very little or no smoothing. The labeling scheme of these pixels is shown in Fig. 2(a).

4. FEATURE EXTRACTION

4.1. *Chain encoding*

Both Fourier shape descriptors and boundary line encoding features require that the contour of the character be extracted. The contour encoding is a technique for expressing the digitized boundary of the character image by a sequence of Freeman chain codes specifying the direction in moving from one vertex to another on the contour. Figure 2(b) shows the Freeman chain codes for tracing a contour from a pixel to its neighboring pixels. For example, if the current pixel is P_0, as in Fig. 2(a), and P_2 is the next pixel on the contour then the chain code is 1. A contour tracing algorithm based on the algorithm of reference (23) is implemented. The algorithm employs the Left-Most-Looking (LML) rule, which may be described in terms of an observer walking along pixels belonging to the object (pixels equal to 1) and selecting the left-most pixel available relative to the direction of entry into the current pixel. Scanning of a character starts from top to bottom and right to left in accordance with the characteristics of Arabic language. At every pixel, the neighboring pixels are traced in a sequence dependent on the previous move (chain code). Figure 3 shows the sequence of direction movements in selecting the next neighboring pixel on the contour for a specific current direction. The algorithm produces two types of contours, namely, external boundary contours (the contours of the main character body, dots, and zigzag) and internal contours (the holes). The external contours' areas are compared to specify the contour of the character main body (which is the largest) and the contours of the dots and zigzag. The number of dots is equal to the number of external contours minus one and the number of holes equals the number of internal contours.

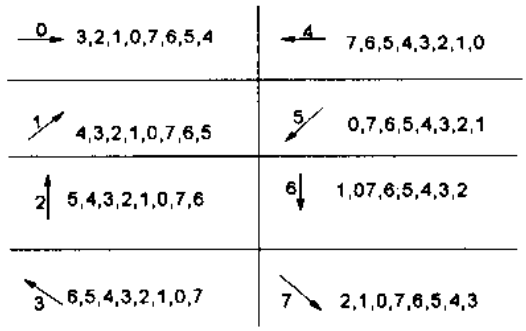

Fig. 3. The sequence of direction movements in tracing the contours.

The location of the dots and stress marks are found by comparing the minimum and maximum y-coordinates of the dot or zigzag contours with those of the contour of the character primary part. Comparing the centers of the dots to that of the primary part is not adequate for some characters (e.g. JEEM (ج), TTHA (ط), KAAF (ق), NOON (ن). For these characters the dots' centers are located in the middle of the character although it is considered as a bottom dot for the JEEM and top dots/zigzag for the KAAF and NOON.

4.2. *Fourier descriptors*

This method is based on using the coordinate values of the contour pixels of the character primary part to obtain the Fourier descriptors. The nature of the character contour data satisfies the mathematical conditions (namely periodicity and continuity). Thus, the character's closed boundary can be represented by a periodic function in a two-dimensional x–y plane tracing once around it. The character contour yields a complex function that denotes the parameteric representation of the boundary coordinates.[24]

The Fourier descriptors used in this work were first proposed by Granlund[25] and further developed by Persoon and Fu[17] and Nikravan *et al.*[20] The normalized Fourier descriptors are invariant with respect to translation, rotation, and scale of the characters. Hence, larger pitch characters may be recognized without the modification of the model features. The contour is represented by the discrete series $x(m)$ which represents the vertices x-coordinates and $y(m)$ which represents

the vertices y-coordinate (where $m = 0, 1, \ldots, L-1$, and L is the total number of pixels representing the closed contour). The discrete formulas for computing the Fourier descriptors of the character contour are defined as follows.

Let Δt_i be the length of link i and Δx_i, Δy_i represent the change in x, y coordinate values as the vertices are traversed. These are easily computed from the chain code of the current segment. Let T be the perimeter length of the closed contour of the character, and n the Fourier descriptor index, and define

$$t_q = \sum_{i=1}^{q} \Delta t_i$$

$$x_q = \sum_{i=1}^{q} \Delta X_i$$

$$y_q = \sum_{i=1}^{q} \Delta y_i.$$

Note that t_q is the cumulative length of the contour from the starting vertex to the current vertex and x_q, y_q are the total projections on the x- and y-axis of the first q links. Let

$$a_{x,n} = \frac{T}{2\pi^2 n^2} \sum_{q=1}^{k} \frac{x_q}{t_q}\left(\cos\frac{2\pi n}{T}t_q - \cos\frac{2\pi n}{T}t_q - 1\right)$$

$$b_{x,n} = \frac{T}{2\pi^2 n^2} \sum_{q-1}^{k} \frac{x_q}{t_q}\left(\sin\frac{2\pi n}{T}t_q - \sin\frac{2\pi n}{T}t_q - 1\right).$$

The Fourier descriptors are then

$$F_d[n] = \sqrt{(a_{x,n}^2 + a_{y,n}^2 + b_{x,n}^2 + b_{y,n}^2)}.$$

The first 10 Fourier descriptors are computed from the contour of the primary part of the character (i.e. n is taken from 1 to 10). This number of Fourier descriptors is adequate for this application. Researchers have shown that a small number of normalized FDs are adequate to recognize objects.[20] However, since some Arabic characters are very close in shape, the use of 10 FDs gives better results.

The Fourier descriptors of two closed curves which differ only in position, orientation, and size with analogous starting points are identical.[16] Hence, to obtain normalized Fourier descriptors the starting point of each character should be normalized. In this work, this normalization was done by adopting a fixed scanning sequence. Each character was scanned from right to left and from top to bottom. The first object pixel (i.e. the first black pixel) is taken as the starting point of the boundary. In addition, in order to have the system invariant to scale, normalization is done by dividing the Fourier descriptor coefficients by the largest one (i.e. by the first one). Hence, the obtained Fourier descriptor coefficient are invariant to rotation, shifting, and scale.

4.3. *Boundary line encoding features*

A printed character shape can vary according to its size, style, and font. Hence, other features are needed, in addition to the Fourier descriptors, to obtain high recognition rate. Boundary line encoding techniques are powerful features that lead to higher recognition rates. The algorithm is based on tracing the contour of the character to generate directions, direction lengths, and curvature features. The following features are obtained from the contour.

4.3.1. *Direction and direction length features.* The chain codes representing the boundary of the input character primary part are obtained (i.e. excluding the contours of dots, zigzag, and holes). Each contour code represents the direction followed in moving from one vertex, of the primary contour, to the next. The number of chain codes in each direction is estimated from the chain codes of the character primary contour and the total number of all the chain codes is found. In addition, the total length of the contour segments in each direction and the total contour length are estimated. The direction lengths are obtained from the repetition of the chain codes where even chain codes have a length of one and odd chain codes have a length of $\sqrt{2}$. The eight direction features are obtained by dividing the number of chain codes in each direction by the total number of chain codes (i.e. in all directions) and the eight direction length features are obtained by dividing the length of the contour segments in each direction by the total contour length. Hence, a total of 16 direction and direction length features are obtained by estimating the percentage of occurrence of each direction and direction lengths with respect to the total number of the generated directions and direction lengths of the character, respectively.

4.3.2. *Curvature features.* The external boundary of the primary part of the processed character is scanned in a clockwise fashion. The external angles between every two successive direction codes in the direction chain (i.e. the angles between the adjacent edges of the character primary part polygon) are used to obtain the concave and convex features. Concave features are generated if the external angle (i.e. the angle at the polygon vertex) is between 0 and 180 deg, whereas the convex features are generated if the outside angle is greater than 180 deg. There are two types of concave features, one type starts with odd chain codes and the other type starts with even chain codes. Figure 4(a) shows the two types of the concave features. For example, the chain code sequences 01, 02, 03, 23, 24, 25, 45, 46, 47, 60, 61, and 67 constitute one type and the other chain codes in the figure constitute the other type. Similarly, there are two types of convex features, one starting with odd chain codes and the other with even chain codes as shown in Fig. 4(b). The rectangle enclosing a character is divided into four quadrants as shown in Fig. 5, hence the character is divided into four parts. The number of concave and convex features in each quadrant of the character primary part are obtained. Hence, eight concave and eight convex features are obtained in the four quadrants of the character

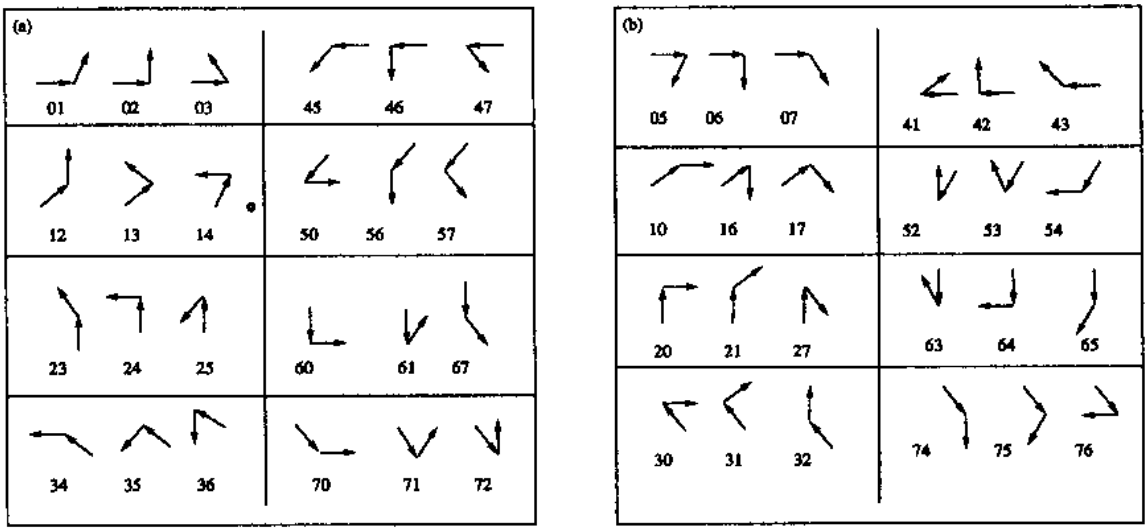

Fig. 4. (a) The concave features. (b) The convex features.

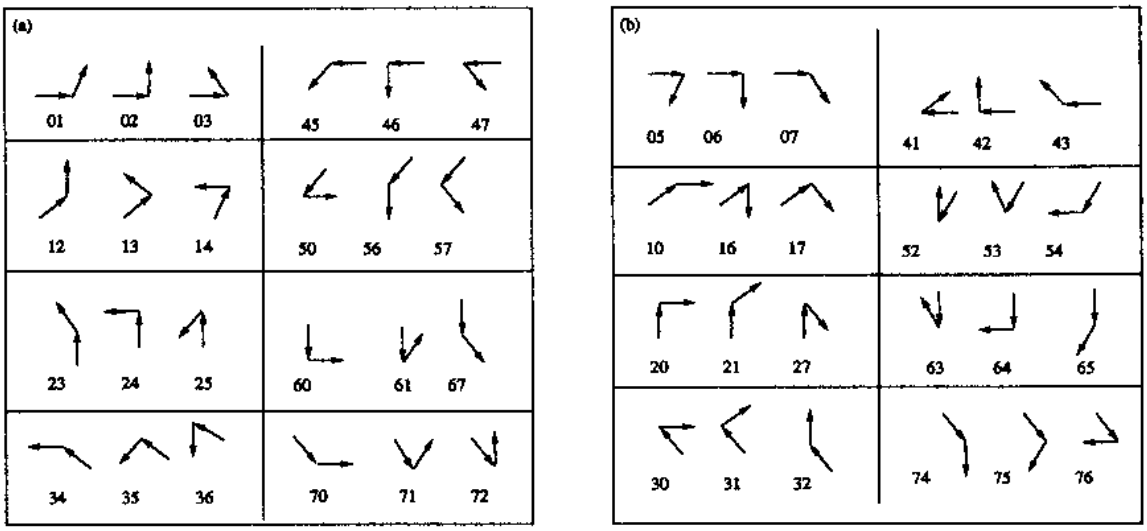

Fig. 5. (a) The eight concave features. (b) The eight convex features.

primary part. In addition, two concave and two convex features for the overall character are obtained. The number of concave and convex features in each quadrant and in the overall character are divided by the total number of the successive pairs of direction codes in the direction chain to obtain 20 concave and convex features. Hence, these features give the percentage of occurrence of each concave and convex feature with respect to the total number of generated curvature features.

4.4. *Holes and dots features*

All the previously described features have only described the external boundary of the character primary part. However, for an enhanced recognition system these features are not enough to classify Arabic characters correctly. Some ambiguous results often occur in an Arabic recognition system due to the characteristic of the Arabic characters where dots are used to differentiate between characters having similar primary parts such as BAA (ب), TAA (ت), and THAA (ث). The dots specifications (namely location and number) are used to discriminate between these similar characters.

The number of dots are estimated by two methods: (i) the number is found by estimating the number of external contours minus one (i.e. discarding the external contour of the character primary part) (ii) the number is found by estimating the total dots area of a character and dividing it by the average dot area (which is estimated in the training phase by averaging all the dot areas of all the characters used in the training phase). The number of dots is taken as the largest of the two estimates. This way of handling the

Fig. 6(a). TAA |( ) with isolated dots.

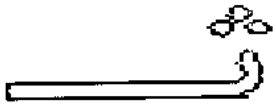

Fig. 6(b). THAA_B( ) with joined dots.

```
 22
2112
21112
211112
2112
 222 22  33
 21112   3113
211112  311113
 21112   31113
 2212    3333
  2
```

Fig. 6(c). The contours of the dots of Fig. 6(b) specified by 2s and 3s.

dots has proven to be very useful as will be shown in Section 6. This step was necessary at this stage, as some characters have their dots connected either due to noise or are very close and the smoothing operation joined the dots, hence giving otherwise a wrong number of dots. In addition, some Arabic fonts use a line on top of a character to indicate two dots and an inverted v shape to indicate a triple of dots. When the number of dots are isolated, as is normally the case as in Fig. 6(a), the number of dots are correctly estimated from the number of external contours. When the dots are joining as in Fig. 6(b) and (c), the estimate of the dots based on the number of external contours is wrong. The number of dots in the figure is estimated as two, where in fact there are three. However, using the second method of estimation (i.e. dividing the total area of the dots by the average dot area) a number of three dots is obtained. Hence, taking the maximum of both techniques gives the best estimate.

The inner details of characters such as the presence and number of holes is obtained to produce some unique features for some Arabic characters. The holes feature is used to identify some characters or resolve any ambiguity in the recognition phase. The presence or absence of the hole, the number of holes, and the relation between the height and width of a hole are used. For example, HHAA_B (ھ) is the only Arabic character with two holes in the character font used in this work.

Holes and dots are powerful features for enhancing the implemented Arabic recognition system. The use of the dot features reduces the number of character classes from 100 to 58. Hence, better recognition rates and speed are achieved using these features. These features are used in the post-recognition phase (i.e. for identifying a specific character after specifying the character model class). Table 1 gives a summary of the used features in the proposed system.

Table 1. Summary of the features used

Feature	No.
A. *Fourier shape descriptors*	10
B. *Boundary line encoding features*	
(1) Direction features	8
(2) Direction length features	8
(3) Concave curvature features in	
(a) The overall character	2
(b) Character quadrants	8
(4) Convex curvature features in	
(a) The overall character	2
(b) Character quadrants	8
(5) Dots feature	1
(6) Holes feature	1
Total features	48

5. RECOGNITION

The recognition phase consists of two parts, namely training (modeling) and testing.

5.1. *Training*

In order to avoid the redundancy and increase the recognition speed and accuracy, models are only generated for the primary parts of the characters (i.e. characters with similar writing forms but different number of dots are assigned to a single class having a common features vector). For example, since the following characters: BAA (ب), TAA (ت), and THAA (ث) have similar primary parts (which implies a common features vector), they are grouped in a single class. These different characters are easily identified based on the number and location of dots. Hence, the 100 different Arabic characters are classified into 58 character classes. This was made possible by isolating the dots feature and using it in the post-recognition phase. Character classes vary from one character in a class, in the case of WAW (ﻮ), to five characters in a class, in the case of the primary part of BAA (ﺒ), TAA (ﺘ), THAA (ﺜ), NOON (ﻨ), and YAA (ﻴ). A feature vector (V) of 46 features is generated for each class (this includes all the features of Table 1 excluding the dots and holes features).

To obtain a smoothed model features, five samples of each character are used in the training phase. In addition, the features of each class are finalized by averaging the features of all characters belonging to the class. In order to estimate the average dot area, the number of dots for each character is given to the system in the training phase. At the end of training, the system computes the average dot area which is used in the recognition phase.

5.2. *Recognition*

The feature vector V for the unknown character is computed and compared to the vectors of the models. Let there be k features (i.e. 46), M_{ij} the jth feature of model i, and V_j the jth feature of V. Nearest neighbor classification based on formula (1)

$$E_i = \sum_{j=1}^{k} |M_{ij} - V_j| \quad (1)$$

is performed to recognize the character. Hence, the distance (F_i) between the new character and all reference vectors are found. The minimum value found (i.e. Min (E_i) yields the recognized model i (it is considered as the class that matches most closely the obtained features vector of the unknown character). Hence, the class of the character is found. In a post-recognition phase the number and location of dots of the input character are used to specify the corresponding unique character.

In the case of ambiguity, as is the case of classes FAA_M (ـفـ) and GHAIN_M (ـغـ) which are very similar (in their primary parts), the hole feature is used. The FAA_M character has a hole while the GHAIN_M character does not have a hole (in this character set). However, in other Arabic character fonts both of these characters may have a hole as shown in Fig. 7. In such cases the relation between the height and width of the hole of the primary parts is used to resolve the ambiguity. FAA_M has the hole height greater than the width while GHAIN_M has the hole width greater than the height as shown in Fig. 7 (i.e. $h_1/w_1 < h_2/w_2$).

6. EXPERIMENTAL RESULTS

The input document image is captured using a scanner with a resolution of 300 pixels per inch. This resolution is sufficient for optical recognition of characters. The scanned document image is transformed into a binary image having two gray levels: black and white. The black pixels represent the text lines and are given a value of 1, while the white pixels represent the background and are given a value of zero. The test data consist of 50 samples of each character.

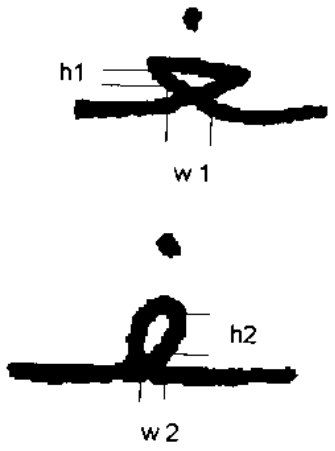

Fig. 7. Another form of writing a GHAIN_M and FAA_M in some fonts.

The system was trained using five samples of each character. The characters were initially smoothed and then the extraction of features was conduced. The system averaged the features of five samples for each character. In addition, since the number of the model classes are less than the number of different Arabic characters, the system averaged the features of each class with more than one character. Moreover, the average dot area of all these characters was computed. This was found necessary for the correct estimation of the number of dots in the case of corrupted data where more than one dot are connected, hence giving an otherwise wrong number of dots.

In the testing phase, the dots and holes features, and then the input character primary part, are extracted. Ten Fourier descriptors and 36 direction, direction-length, and curvature features are extracted. Hence, a total of 48 features of each input character are extracted. The first 46 features are compared with the corresponding features of the models. The model closest to the input character (in terms of the distance measure in equation (1)) is considered to identify the input character. In this phase, all the classes were correctly recognized except the FAA_M (ـفـ) which was confused with the GHAIN_M (ـغـ) class. To resolve this ambiguity, the hole feature was used. FAA_M has a hole while GHAIN_M does not have a hole (in this test character set). In the post-recognition phase, specific characters are assigned to the input character using the dots feature and, in rare occasions, the hole feature. The number and location of the dots are used to identify characters. A recognition rate of 98% was achieved in the post-recognition phase. Most of the errors and rejections came from corrupted data. The following are some examples of the tested characters.

The only error in the recognition of model classes was the assignment of the class representing the primary parts of GHAIN_M (ـغـ) to that of FAA_M (ـفـ). The contours of these classes are shown in Fig. 8. As is clear from the figure, the two main contours are very similar. However, using the hole feature, GHAIN_M has a hole while FAA_M has a hole. Hence using the hole feature brings the recognition rate to 100% for the classes. In other Arabic character fonts the GHAIN_M may have a hole. In that case, the relation between the height and width of the hole is used as a feature. While GHAIN_M has a low height/width ratio, FAA_M has a larger height/width ratio as shown in Fig. 7. Hence, this ambiguity between GHAIN_M and FAA_M is resolved using the hole feature.

The character recognition rate of the post-recognition phase of this system is 98%. The major part of the errors in classifying characters was mainly due to wrong estimation of the number of dots. Figure 9 shows a three-dotted character which is correctly recognized as the dots were well isolated. Figure 10 shows a character with three dots but two of them are joined. Using the

Fig. 8. The contours of these characters are similar but one has a hole while the other does not.

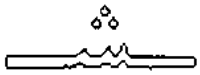

Fig. 9. A three-dotted character with isolated dots.

Fig. 10. Three-dotted character with two joined dots.

Fig. 11. Two characters with joined dots with a small total area.

Fig. 12. A character with two dots joining the primary contour.

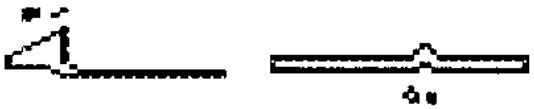

Fig. 13. Examples of characters with small dots that were correctly recognized.

estimate of dots from the number of external contours gives 2 which is not correct. However, using the estimate from dividing the total area of the dots by the average dot area (which is found in the training phase) gives 3 which is the correct result. Using the largest of the two, as detailed earlier, gives the correct number of dots. Hence, this character was correctly recognized. Figure 11 show examples where the number of dots could not be estimated correctly. There are two joined and small dots. Both of the dots estimate gave the number of dots as one which is wrong. Even humans seeing the dots alone may give the same estimate. However, humans recognize the character in context.

There are cases where the dots are joining the main body of the character as shown in Fig. 12. One external contour was extracted which indicates that there are no dots. This character was wrongly classified as a non-dotted YAA. This, however, is not a problem since in some Arabic dialects the dots are not used in writing this character. Figure 13 shows two characters with very small total dots area. These dots were correctly estimated from the number of external contours.

7. CONCLUSIONS

The paper presented a system for Arabic character recognition based on estimating the Fourier descriptors and curvature features of the contour of the primary part of Arabic characters. Ten Fourier descriptors, 16 direction and direction length features, and 20 concave and convex features for the quadrants of a character and for the whole character are extracted. In addition, the number and location of dots and the number of holes are also found. The number of holes is estimated by two methods, the first method estimates the number from that of the external contours and the other by dividing the total area of dots by the average dot area found in the training phase. The largest of the two estimates was taken as the correct one. This method of estimating the dots has proved to be more reliable than either of the two. A total of 48 features are used by the system to classify 100 Arabic characters. The characters are classified initially to 58 model classes using the first 46 features. Then, the particular character is found using the dot and hole features.

Averaging of the features' vectors of each character class is achieved by three levels. Initially the character image is smoothed using the statistically based algorithm of Section 3. Then the features of five samples of each character are averaged. Since each class represents from one to five characters, the features of the characters of each class are averaged.

All model classes were correctly recognized using the first 46 features except the GHAIN_M class which was confused with FAA_M class. Using the hole features, this confusion was resolved hence bringing the recognition rate of the model classes to 100%. This rate came down in the stage of recognizing the particular character to 98%. The major part of these errors are due to a wrong estimate of the dots which is a direct result of corrupted data as shown in the previous examples.

The use of large number of features was necessary to discriminate between Arabic characters that are very close in shape as is the case with GHAIN_M and FAA_M. The combination of the Fourier and curvature features has proven to be very useful in achieving higher recognition rates.

A new technique for the isolation of dots in the case of corrupted data and the segmentation of Arabic text will be the subject of future work. In addition, other techniques will be investigated. The use of several recognition techniques is necessary to achieve higher recognition rates of Arabic characters.

Acknowledgement—The author would like to express his gratitude to the referees for their constructive criticism. The incorportation of their suggestions and comments significantly improved the clarity of the final manuscript.

REFERENCES

1. R. L. Grimsdale, F. H. Summer, C. J. Tunis and T. Kilburn, A system for the automatic recognition of patterns, *IEEE Process.* **106**, 210–221 (1959).

2. V. K. Govindan and A. P. Shivaprasad, Character recognition—a review, *Pattern Recognition* **23**, 671–683 (1990).
3. R. Casey and G. Nagy, Recognition of printed Chinese characters, *IEEE Trans. Electron. Comput.* **15**, 91–101 (1966).
4. A. Nouh, A. Sultan and R. Tolba, An approach for Arabic characters recognition, *J. Engng Sci. Univ. Riyadh* **6**(2), 185–191 (1980).
5. F. Khaly and Sid-Ahmed, Machine recognition of optically captured machine printed Arabic text, *Pattern Recognition* **23**, 1207–1214 (1990).
6. H. Al-Yousefi and S. Udpa, Recognition of handwritten Arabic characters, *Proc. SPIE 22nd Annual Technical Symp. on Optical and Optoelectronics Applied Science and Engineering*, San Diego, California, Vol. 974, Applications of Digital Image Processing XI, pp. 330–336 (1988).
7. H. Al-Yousefi and S. Udpa, Recognition of handwritten Arabic characters via segmentation, *Arab Gulf J. Scient. Res.* **8**(2), 49–59 (1990).
8. R. Ramsis, S. S. El-Dabi and A. Kamel, Arabic character recognition system, IBM Kuwait Scientific Center, report No. KSC027, January (1988).
9. S. S. El-Dabi, R. Ramsis and A. Kamel, Arabic character recognition system: a statistical approach for recognizing cursive typewritten text, *Pattern Recognition* **23**, 485–495 (1990).
10. H. Abdelazim and M. Hashish, Automatic recognition of handwritten Hindi numerals, *Proc. CompuEURO'89: VLSI and Computer Peripherals*, Hamburg, pp. 287–298 (1989).
11. H. Y. Abdelazim, A. M. Mousa, Y. L. Saleh and M. A. Hashish, Arabic text recognition using a partial observation approach, *Proc. 12th National Computer Conf.*, Riyadh, Saudi Arabia, 21–24 October, pp. 427–437 (1990).
12. H. Almuallim and S. Yamaguchi, A method for recognition of Arabic cursive handwriting, *IEEE Trans. Pattern Analysis Mach. Intell.* **PAMI-9**(5), 715–722 (1987).
13. H. Goraine, M. Usher and S. Emami, Off-line Arabic character recognition, *IEEE Comput.* **27**(7), 71–74 (1992).
14. K. El-Gowely, O. El-Dessouki and A. Nazif, Multiphase recognition of multi-font photo script Arabic text, *Proc. 10th Conf. on Pattern Recognition*, pp. 700–702 (1990).
15. A. Amin, Machine recognition and correction of Arabic printed text, *IEEE Trans. Syst. Man Cybern.* **19**(5), 1300–1306 (1989).
16. C. T. Zahn and R. Z. Roskies, Fourier descriptors for plane closed curves, *IEEE Trans. Comput.* **C-21**(3), 269–281 (1972).
17. E. Persoon and King-Sun Fu, Shape discrimination using Fourier descriptors, *IEEE Trans. Syst. Man Cybern.* **SMC-7**(3), 170–179 (1977).
18. M. T. Lai and C. Y. Suen, Automatic recognition of characters by Fourier descriptors and boundary line encodings, *Pattern Recognition* **14**, 383–393 (1981).
19. P. Borghesi, V. Cappellini, R. Carla, A. Del Bimbo, A. Mecocci and M. T. Perseschi, Digital image processing techniques for object recognition and experimental results, *Digital Signal Process.* **84**, (1984).
20. B. Nikravan, R. M. Baul and K. F. Gill, An experimental evaluation of normalized Fourier descriptors in the identification of simple engineering objects, *Comput. Ind.* **13**, 37–47 (1989).
21. H. Itoh, T. Hirata and N. Ishii, Machine parts recognition and processing using Fourier descriptors, *Syst. Comput. Japan* **20**(8), 26–36 (1989).
22. K. Arbter, W. E. Snyder, H. Burchardt and G. Hirzinger, Application of affine-invariant Fourier descriptors to recognition of 3-D objects, *IEEE Trans. Pattern Analysis Mach. Intell.* **12**(7), 640–647 (1990).
23. T. Pavlidis, *Algorithms for Graphics and Image Processing*. Computer Science Press Rockville, Maryland (1982).
24. T. Pavlidis, *Structural Pattern Recognition*, Vol. 1. Springer, New York (1977).
25. G. H. Granlund, Fourier preprocessing for hand print character recognition, *IEEE Trans. Comput.* **C-21**, 195–201, Feb. (1972).

About the Author—SABRI A. MAHMOUD received his B.S. in electrical engineering from Sind University, Pakistan, in 1972. He was an Electrical Engineer for five years at Shuaiba North Power Station, Kuwait. He received his M.S. in computer sciences from Stevens Institute of Technology, U.S.A., in 1980. He has worked with Royal Saudi Naval Forces as senior systems engineer for five years. He received his Ph.D. in information systems engineering from the University of Bradford, U.K., in 1987. In September 1988 he joined the Computer Engineering Department, College of Computers and Information Systems, King Saud University, Saudi Arabia, as an Assistant Professor, where he is now an Associate Professor. His research interests include image processing, pattern recognition and the use of fuzzy concepts and neural networks in the above applications. Dr Mahmoud is a senior member of IEEE and a member of the Signal Processing Society.