ELSEVIER

Signal Processing 56 (1997) 59–75

SIGNAL PROCESSING

Arabic character recognition using 1-D slices of the character spectrum

Saleh A. ALshebeili[a,*], Asim A.-F. Nabawi[b], Sabri A. Mahmoud[c]

[a]*Electrical Engineering Department, King Saud University, P.O. Box 800, Riyadh 11421, Saudi Arabia*
[b]*Computer Engineering Department, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia*
[c]*Al-Manarain Est. for Technical Applications, P.O. Box 53531, Riyadh 11593, Saudi Arabia*

Received 20 October 1995; revised 10 April 1996 and 5 August 1996

Abstract

An Arabic character recognition algorithm using 1-D slices of the character spectrum is presented. The Fourier spectrum of the character's projections on the X- and Y-axes is estimated. The features are extracted from this 2-D spectrum. The features of 10 sets of characters were used as model features. The features of an input character are compared with the models' features using a distance measure. The model with the minimum distance is taken as the class representing the input character. Experimental results have shown that the presented algorithm is capable of recognizing Arabic characters with a recognition rate of 99.06%, using 10 features of the X-projection. This rate rises to 99.94% when 10 features of the Y-projection are added. The proposed system was compared with another, based on the Fourier descriptors, which was capable of recognizing 97.5% of test characters using 10 Fourier descriptors. The presented technique is superior to that of the Fourier descriptors in terms of recognition rates and speed, as fast Fourier transform is used in the calculation of the spectrum while standard equations are used to compute the Fourier descriptors. Both techniques are invariant to shift. However, the Fourier descriptor is invariant also to rotation and scale. © 1997 Elsevier Science B.V.

Zusammenfassung

Es wird ein Algorithmus zur Erkennung von arabischen Schriftzeichen vorgestellt, der auf eindimensionalen Schnitten des Schriftzeichenspektrums beruht. Das Fourierspektrum der Projektionen des Schriftzeichens auf die X- und Y-Achse wird geschätzt. Die Merkmale werden aus diesem zweidimensionalen Spektrum gewonnen. Die Merkmale von 10 Zeichensätzen wurden als Modellmerkmale verwendet. Die Merkmale eines Eingangszeichens werden mit Hilfe eines Abstandsmaßes mit den Modellmerkmalen verglichen. Das Modell mit dem kleinsten Abstand wird als die Klasse herangezogen, die das Eingangszeichen repräsentiert. Experimentelle Ergebnisse haben gezeigt, daß der vorgestellte Algorithmus im Stande ist, arabische Schriftzeichen mit einer Erkennungsrate von 99.06% unter Verwendung von 10 Merkmalen der X-Projektion zu erkennen. Diese Rate steigt auf 99.94% an, wenn 10 Merkmale der Y-Projektion hinzugefügt werden. Das vorgeschlagene System wurde mit einem anderen auf der Basis von Fourierdeskriptoren verglichen, das im Stande war, 97.5% der Testzeichen mit Hilfe von 10 Fourierdeskriptoren zu erkennen. Das vorgestellte Verfahren ist demjenigen mit Fourierdeskriptoren hinsichtlich Erkennungsrate und Geschwindigkeit überlegen, da die

*Corresponding author. Fax: 966 1 467 6757; e-mail: F45E040@saksuoo.bitnet.

schnelle Fouriertransformation zur Berechnung des Spektrums eingesetzt wird, wogegen Standardgleichungen zur Berechnung der Fourierdeskriptoren eingesetzt werden. Beide Verfahren sind invariant gegen Verschiebung. Jedoch ist dasjenige mit Fourierdeskriptoren auch gegen Drehung und Skalierung invariant. © 1997 Elsevier Science B.V.

Résumé

On présente ici un algorithme de reconnaissance de caractères arabes utilisant des coupes 1-D du spectre du caractère. On estime le spectre de Fourier des projections du caractère sur les axes X et Y. Les caractéristiques sont ensuite extraites de ce spectre 2-D. Celles de 10 ensembles de caractères ont été utilisées comme références du modèle. Les caractéristiques d'un caractère entré sont comparées avec celles du modèle à l'aide d'une mesure de distance. Le modèle présentant la distance minimale est considéré comme étant la classe représentant le caractère entré. Les résultats expérimentaux ont montré que l'algorithme présenté est capable de reconnaître les caractères arabes avec un taux de reconnaissance de 99.06%, en utilisant 10 caractéristiques de la projection sur l'axe X. Ce taux monte à 99.94% lorsqu'on y ajoute les 10 caractéristiques de la projection sur l'autre axe. Ce système a été comparé à un autre, basé sur les descripteurs de Fourier, qui pouvait reconnaître 97.5% des caractères en utilisant 10 descripteurs de Fourier. La technique présentée est supérieure à celle des descripteurs de Fourier en termes de taux et de vitesse de reconnaissance, car on utilise la transformée de Fourier rapide dans le calcul du spectre, alors que les équations standards sont utilisées pour calculer les descripteurs. Les deux techniques sont insensibles au déplacement, toutefois les descripteurs de Fourier sont également insensibles à la rotation et à l'échelle. © 1997 Elsevier Science B.V.

Keywords: Arabic character recognition; OCR; Fourier descriptors; Contour analysis; Character spectrum

1. Introduction

It has been centuries since writing in general, and Arabic writing in specific, has been in use as a means of formalizing, preserving and transmitting human thoughts, ideas and achievements, both scientific and literary. It has been almost the same distance in time that separated the advent of electronic data processing machines, or simply computers, from the advent of writing. The huge repertoire of data in written form, available today in every field of human knowledge, cries for redemption from its usual paper form, to be transferred into the spacious and flexible storage of the computing machines, for more prompt and fruitful manipulations. The scope of this emerging technological advance is very wide, and the applications are numerous and yielding. For example, helping the blind to read; sorting mail using addresses off the envelopes of the sent letters; processing checks and other financial forms in banks and financial firms; verification of personal signatures on checks and financial transactions; furthering the cause of, and enhancing the databases needed for, computer-aided instruction (CAI) programs and activities; improving man–machine interaction; office automation; and, last but not the least, transferring the huge amounts of data and information, available now in text form, to digital text form.

The continued efforts of researchers in the field have resulted in the too much cherished success in the field of optical character recognition (OCR) for several languages. These languages are mainly using Latin-based character sets, and some others including Chinese, Cyrillic (Russian) and Japanese. To have an overview of the achievements on this front in the marketplace, it is opportune to mention that early OCR machines, in the 1950s, were hardware based and could mainly read numbers typed using special fonts [12]. This state of affairs has gradually changed, and in the 1970s automatic text readers based on software logic emerged to read texts with special fonts [28]. Till that point in time, the reading algorithms were implemented on mainframes and mini-computers. The scope of those pioneer attempts at the ultimate automatic reading machine has widened in the 1980s, and the early primitive reading machines gave way to new ones capable of reading multiple fonts on the same typewritten page. As a result of the revolution in the hardware market that coincided with the launching of the personal computer (PC), several commercial systems are now available for PCs [18, 33], to read a variety of writing styles, including handwritten, and dealing with several character sets, including Korean and Chinese. The accuracy of some of these systems approach a figure of 99.9% [16], with speeds up to five characters per second. These figures, which are achieved under certain circumstances, slip down to only 99.0% for on-line recognition [22, 30, 32]. However, on-line character recognition applications are limited, compared to those of off-line applications.

Similar research efforts in the field of Arabic OCR (AOCR) are still suffering from lack of adequate support and enjoying a less opportune environment, for several reasons. It has almost only started in 1975 by Nazif [20], as compared to earlier research efforts in Latin, which may be traced back to the middle of the 1940s [12]. Several researchers are now engaged in studying AOCR, and they are trying to make for the losses in many corners, but there exists still an obvious lack of some important factors. Foremost of these factors are coordination and standardization. By standardization we mean the availability, among other elements, of adequate Arabic databases, on-line dictionaries, and programming tools, as well as the emergence of well-established benchmark test procedures. The first group of elements is direly needed for conducting serious AOCR work, while the latter is deemed necessary before evaluation of the relative performance of different algorithms and techniques may be reliably carried out. Researchers involved in Latin OCR enjoy the privilege of using standard databases for the evaluation of their algorithms and products [26]. Lack of coordination between researchers in the AOCR field is still a reality which has its bearing, albeit the emerging efforts in several directions, hence it needs no further elaboration.

As a rough guide to the state of affairs prevailing in the AOCR field, it can be stated that the recognition rates reported in the literature [1] do not approach the 99.9% rate reported for Latin OCR systems. This statement should be considered in the light of the nonstandard nature of those limited tests undertaken by either the academic researchers or the producers of AOCR systems. A major hindrance in this field is the ensuing difficulties due the cursive nature of Arabic texts, which will be addressed, in some detail, in the next section. Some examples of the research work on Arabic character recognition is stated below. For a more comprehensive study, reference may be made to the comprehensive survey of Badr and Mahmoud [1].

Al-Yousefi and Udpa [2, 3] 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. El-Dabi et al. [6, 27] used the accumulative invariant moments as feature descriptors. They used the width of the smallest character in the set as the minimum character width. The moments are then 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. This process is repeated until the character is recognized. The system is font-dependent and requires heavy computations due to computing the moments for different character widths, and the sensitivity to the smallest variation in the input patterns.

Fakir and Sodeyama [10] used Hough transformation for feature extraction, dynamic programming matching technique to classify character primary part, and topological classifier to recognize the characters using the character stress mark (viz. single, double and triple dots and the zigzag character (ء)). A limited test database consisting of two hundred words was used, where each word was written by three different persons. A total of 400 characters were used in the testing. Recognition rates of 97% and 80% were obtained for printed and handwritten characters, respectively. The recognition rate is low and the technique is computationally expensive. The technique of Goraine and Usher [11] is based on thinning the Arabic word and extracting the strokes (which may be sub-characters). Strokes are classified into eleven primitives in the first classification stage. In the second stage, the primitives are combined into characters. In a post-processing stage, words are spell-checked and the most-likely interpretation of the word is chosen. Thinning is an expensive operation which, sometimes, produces skeletons with extra spur tails, line segments may come as stair-case or zigzag-like segments. Similar to the previous technique, the used database is limited (60 words for training and 60 words for testing in the first experiment and 400 characters from an Arabic magazine in the second experiment). The obtained recognition rates were 95.87% and 90% for the printed words and the magazine words, respectively. The obtained rates are far inferior compared with those of other techniques without a spell checker.

Hassibi [14] used neural networks in the recognition phase of a machine-printed Arabic OCR. The segmentation stage produces meta character glyphs which may be a single character, a valid ligature, or a character piece. The easy-to-recognize meta character glyphs are recognized using classical classification techniques and a neural network is used to recognize the more difficult meta characters. Contextual analysis is used to join meta characters into words. Lexicons are used in the post-processing phase to improve the system recognition rate. The system used the largest Arabic text database reported in the literature of Arabic OCR. The recognition speed was not reported and is expected to be low due to the use of several techniques in the OCR system. In addition, the segmentation stage required manual intervention which usually slows the process, and the segmentation stage is restricted to 99% correct segmentation for the neural network to be able to learn the correct segments.

In this paper, an automatic Arabic character recognition system is presented. The system recognizes segmented typewritten characters. The X- and Y-projections of the segmented characters are obtained, and the Fourier transforms of the projections are then computed. The system is compared with another recognition system based on computing the Fourier descriptors of the closed contour of the character's primary part. Ten of the Fourier descriptors, that describe the character's boundary by a set of numerical features, are used. Results of using the presented technique are compared with new runs of the other technique using the same sets of data.

The organization of the paper is as follows. Section 2 states the characteristics of Arabic texts. Feature extraction using the X- and Y-projections of the character's primary part are addressed in Section 3. The classification phase, including training and recognition stages, is addressed in Section 4. Performance evaluation of the results is conducted in Section 5. The Fourier descriptor technique, used for comparison purposes with the presented technique, is concisely presented in Appendix A.

2. Characteristics of Arabic texts

The aforementioned lack of AOCR research activity may be attributed, in part, to the special characteristics of Arabic texts. Unlike Latin text, Arabic text runs from right to left. Texts in some other languages (e.g. Persian and Urdu) run also in the same direction (most of them essentially share the same Arabic alphabet, but with some modifications and/or additions).

The Arabic character set contains basically 28 characters. Due to the cursiveness of the Arabic text, whether machine-printed or hand-written, this set is actually augmented with several variations of each character. This is a direct result of almost every character assuming different shapes when joined from its right- or left-side, as well as being in the beginning or isolated. Three more characters (viz. Hamzah (ء), Taa-Marbootah (ة) and Alif-Maqsurah (ى)) augment the basic Arabic character set. Connecting Arabic characters results in a word having one or more connected components, or sub-words. As an example, the word (الرسلات) consists of four sub-words: the first one is (ا) which consists of one character, the second and third which consist of three characters each (سلا ، لر), and the fourth sub-word is (ت), which is a single isolated character. Moreover, each Arabic character does not have a fixed size (height and width) even when a set of characters with a fixed font size is used. The character size varies according to its shape which is a function of its position in the word.

Out of the Arabic basic character set of 28 characters, 15 characters have from one to three dots, as secondary parts. These dots are normally used to differentiate between the mostly otherwise similar characters. Additionally, four characters can have, associated with them, a secondary part called Hamzah (ء). These characters are Alif (ا), Kaaf (ك), Waaw (و) and Alif-Maqsurah (ى). The dots and Hamzahs may be found in different locations with respect to the character's main body: above it as in Alif (أ) and Noon (ن), below it like Baa (ب) and Yaa (ي), or inside it like Jeem (ج) and Kaaf (ك). The different shapes of Arabic characters are listed for convenience in Table 1. Note that the character Yaa (ي) has another form with the same main body but without the two dots below it. This is called Alif-Maqsurah (ى), and is accordingly pronounced as an Alif (ا). Although Hamzah (ء) is used as a secondary part in Arabic texts, it is frequently used as a character on its own. The word (سماء) is a common example of a word with Hamzah as a main character.

Another distinguishing feature of an Arabic text is the presence of a writing baseline. The baseline is a horizontal line that runs through the connected portions of the text. The baseline has the maximum number of text pixels. Nevertheless, in some Arabic (especially calligraphic) fonts, and in hand-written texts, characters connected to form a word may overlap vertically (sometimes without touching), and hence two or more characters may be connected above the baseline. The direction of the baseline in an Arabic text is a helpful piece of information since it is the easiest way of deducing the orientation of the text. On its own, knowledge of the exact location of the baseline, among other things, enhances the ability of the recognition system to deskew pages, as well as to both separate the text into lines and segment the characters [5, 8, 15, 31].

Table 1
The different shapes of the basic Arabic characters. The models are labeled using (*)

Character	Isolated	Beginning	Middle	End Form
Alif	* ا	ا	ـا	* ـا
Baa	ب	* بـ	* ـبـ	* ـب
Taa	ت	تـ	ـتـ	ـت
Thaa	ث	ثـ	ـثـ	ـث
Jeem	ج	جـ	ـجـ	ـج
Hhaa	* ح	* حـ	ـحـ	ـح
Khaa	خ	خـ	ـخـ	ـخ
Daal	* د	د	ـد	* ـد
Thaal	ذ	ذ	ـذ	ـذ
Raa	* ر	ر	ـر	* ـر
Zay	ز	ز	ـز	ـز
Seen	* س	* سـ	ـسـ	ـس
Sheen	ش	شـ	ـشـ	ـش
Ssaad	* ص	* صـ	* ـصـ	ـص
Dhaad	ض	ضـ	ـضـ	ـض
Ttaa	* ط	طـ	ـطـ	* ـط
Tthaa	ظ	ظـ	ـظـ	ـظ
Ain	* ع	* عـ	* ـعـ	* ـع
Ghain	غ	غـ	ـغـ	ـغ
Faa	* ف	* فـ	* ـفـ	ـف
Qaaf	ق	قـ	ـقـ	ـق
Kaaf	* ك	* كـ	* ـكـ	ـك
Laam	* ل	* لـ	* ـلـ	ـل
Meem	* م	* مـ	* ـمـ	ـم
Noon	* ن	نـ	ـنـ	ـن
Haa	ه	* هـ	* ـهـ	* ـه
Waaw	* و	و	ـو	* ـو
Yaa	* ي	يـ	ـيـ	* ـي
LaamAlif	* لا	لا	ـلا	ـلا

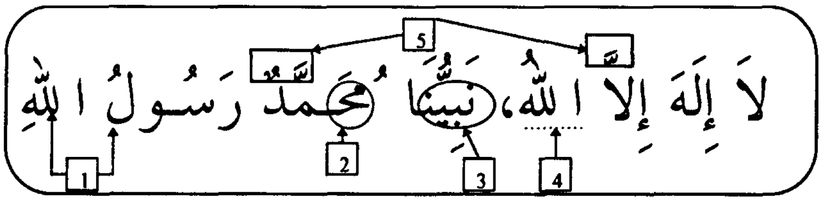

لَا إِلَهَ إِلَّا اللَّهُ، نَبِيُّنَا مُحَمَّدٌ رَسُولُ اللَّهِ
1
2
3
4
5

Fig. 1. An Arabic sentence demonstrating some of the characteristics of Arabic text. (1) Three different shapes for the LAAM character, (2) two overlapping (ligature) characters, MEEM and HHAA, (3) three characters with similar shapes, BAA, YAA and NOON, and NOON character with two different shapes, (4) one Arabic word with two sub-words, and (5) gives examples of vowel points.

The use of special stress marks, called vowel points, is another characteristic of Arabic texts. Characters in a written word can have different vowel points such as Fat-hah (ـَ), Kasrah (ـِ), Dhammah (ـُ), Sukoon (ـْ), Shaddah (ـّ) and Maddah (ـٓ). The first four of these may be considered as basic vowel points. Some of these vowel points may be used in conjunction with some others. For example, Shaddah (ـّ) may be used with any of the basic vowel points on the same single character. Moreover, Tanween may be formed by having double Fat-hahs (ـً), Dhammahs (ـٌ) or Kasrahs (ـٍ). These vowel points are written as strokes, placed either on top of, or below, the character. A different vowel point on one or more character of the same word may radically change the meaning of the word. As an example, the word (أسدي) has the following meanings: *my lions*, which is a noun in the plural form, if written as (أُسْدِي), *my lion*, which is also a noun in the singular form, if written as (أَسَدِي), while if the same word is written as (أَسْدَى) it means: *gave*, which is a verb in the past tense. Note, for example, that the difference between the first two otherwise identical words, is the vowel points on the first two characters, while the difference between the last two words is the vowel points on the two middle characters, as well as removing the two dots from under the last character. Nevertheless, readers of Arabic texts are capable of deducing the meaning from the context, when the text has no vowel points at all.

Last, but not the least, several characters can combine to form a ligature, especially in typeset and hand-written texts. For example, Laam (ل) and any of the characters Alif (ا) or Meem (م) may be connected to form two different types of ligatures: horizontal (لا) or vertical (لم), respectively. Although LaamAlif (لا) is considered as a ligature, it is the only ligature available as one character stroke on computer keyboards and is the only ligature defined in the ASCII code of Arabic character set. All other ligatures are formed by two keystokes. These ligatures pose as a problem in AOCR. It is difficult to segment, and, hence, need to be, a priori, added to the character set in order to facilitate recognition. Otherwise, it may be recognized without segmentation, but it will need manual intervention to correct it as rejections, using a proper dictionary to edit the mistakes of the OCR system. It is suggested in this respect that a considerable amount of the total recognition cost is allocated to the manual intervention necessary to correct such mistakes of the OCR system [19]. Fig. 1 demonstrates some of these characteristics using an Arabic sentence consisting of eight words as a vehicle.

The above-mentioned characteristics of Arabic text heavily influence AOCR. One of the crucial and time-consuming steps in any AOCR system is the segmentation hase. Many recognition errors are attributed to this phase. The effort and techniques deemed necessary for recognizing isolated Arabic characters are not fundamentally different from those used for Latin OCR. However, the wider variability of the Arabic character shapes needs more attention. It is generally accepted that segmentation of a typeset Arabic text is harder than that of a typewritten text. This is mainly due to ligatures and character overlapping. However, recognizing hand-written Arabic text is even harder.

3. Feature extraction

A typical system used for character recognition includes two main stages: the feature extraction and classification stages. In the first stage, each character is represented by some features that distinguish it from other characters. In the second stage, the feature vector of the incoming unknown character is assigned to the closest class using a suitable classification criterion.

Models of the Arabic characters are generated for the primary parts of the characters (i.e. characters with similar writing forms, but with different number of dots, are assigned to a single class having a common features vector). For example, since the following characters: Ttaa (ط) and Tthaa (ظ), have similar primary parts, they are grouped into a single class. These different characters are easily identified based on the number and location of dots. Hence, the different Arabic characters are classified into 43 character models. This allows us to avoid redundancy, save memory space, and enhance the recognition speed and accuracy. The algorithm of [17] may be used for isolating the dots' and holes' features and using it in the post-recognition phase for classifying the characters after classifying the models. Table 1 shows all the Arabic character shapes, and the 43 models are marked by an asterisk (*) beside the respective shape of the character.

It is assumed, in our development below, that Arabic characters that appear at the input of the recognition algorithm are segmented. This is the case in some applications such as automatic mail sorting, when the zip code is used, as well as the recognition of Arabic variables used in mathematical equations. For those applications where Arabic characters may appear cursive in text, the recognition algorithm must be preceded by an independent segmentation algorithm. In such cases, the algorithms reported in [4, 7, 9] may be applied before the feature-extraction stage.

The binarized image obtained from an A4 flatbed scanner is read into a 1-D buffer. Each page has a number of characters, each character is isolated using the X- and Y-projections of the whole page in order to find the coordinates of the character. The image area of each character is then read into a 2-D array buffer (using the computed coordinates) for future use in further processing. This yields the advantage of simple addressing of character pixels. Each pixel is addressed using its X- and Y-indices. Some binarized Arabic characters are shown in Fig. 2, where only the black pixels inside the character contour are shown as 1's, while the pixels outside the contour of each character, as well as those of the holes in its body are digitized as 0's. These 0-value pixels are displayed in the characters of Fig. 2 as blank pixels for clarity of presentation. The steps taken during the process of extracting Arabic characters are summarized in Fig. 3(a).

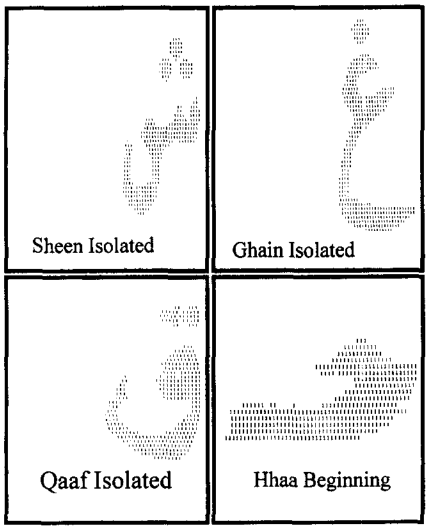


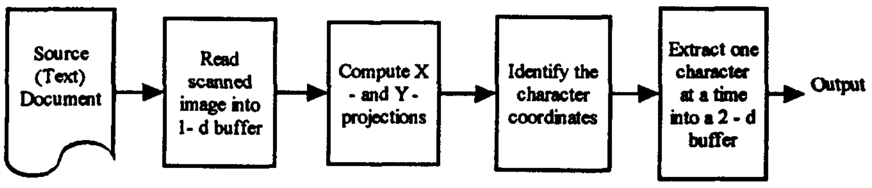
Fig. 2. Samples of some binarized Arabic characters.

In this section, we present a new method for the recognition of Arabic characters using 1-D slices of the 2-D Fourier transform, $C(\omega_1, \omega_2)$, of the character image. The proposed method utilizes the 1-D slices corresponding to $\omega_1 = 0$ and $\omega_2 = 0$. The main motivation behind the use of 1-D slices of the spectrum for recognition is twofold:

(i) to reduce the dimensionality of the problem, hence improve its computational complexity;
(ii) to allow the use of the well known 1-D signal processing techniques.

Let $C(\omega_1, \omega_2)$ be the 2-D Fourier transform of the character $c(m, n)$; i.e.,

$$C(\omega_1, \omega_2) = \sum_m \sum_n c(m, n) \exp(-\mathrm{j}\omega_1 m - \mathrm{j}\omega_2 n). \quad (1)$$


Source (Text) Document
Read scanned image into 1- d buffer
Compute X - and Y - projections
Identify the character coordinates
Extract one character at a time into a 2 - d buffer
Output
(a)

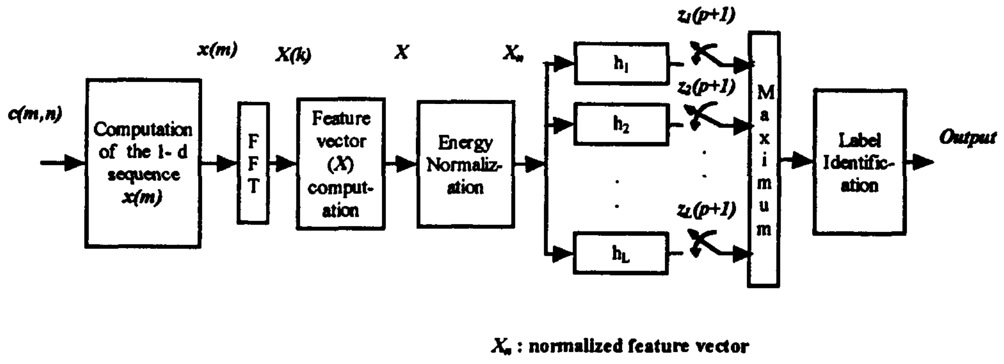


Fig. 3. (a) Schematic diagram of extracting characters from input documents. (b) Schematic diagram of the recognition algorithm.

By setting $\omega_1 = \omega$ and $\omega_2 = 0$ in Eq. (1), we obtain

$$C(\omega, 0) = \sum_m \sum_n c(m, n) \exp(-j\omega m)$$
$$= \sum_m \sum_n c(m, n) \exp(-j\omega m)$$
$$= \sum_m x(m) \exp(-j\omega m),$$

where $x(m) = \sum_n c(m, n)$. Obviously, the 1-D slice $C(\omega, 0)$ is the Fourier transform of the 1-D sequence $x(m)$. A similar expression can be developed for the 1-D slice $C(0, \omega)$. Specifically, by defining $y(n) = \sum_m c(m, n)$, it is straightforward to show that

$$C(0, \omega) = \sum_n y(n) \exp(-j\omega n).$$

For convenience, $C(\omega, 0)$ and $C(0, \omega)$ will be denoted by $X(\omega)$ and $Y(\omega)$, respectively.

Given an input character $c(m, n)$, the data feature vectors X and Y are computed as follows:

$$X = (|X(2\pi/N)|, |X(4\pi/N)|, \ldots, |X(2\pi p/N)|),$$
$$Y = (|Y(2\pi/N)|, |Y(4\pi/N)|, \ldots, |Y(2\pi p/N)|), \qquad (2)$$

where N is an integer which is greater than or equal to the lengths of $x(n)$ and $y(n)$. The feature vector contains the absolute values of p samples, equally spaced in frequency, of the Fourier transform of the signal. These samples can be computed in an efficient manner using the fast Fourier transform (FFT) algorithm or one of its modified versions which dispenses with any unnecessary computations from the FFT algorithm when a subset of the discrete Fourier transform (DFT) points are needed. In particular, the FFT algorithm provides a computationally efficient method for calculating a length-N DFT given N input data points. However, when a number of the input data or the output data points can be neglected, increased computational efficiency may be achieved by modifying the FFT algorithm. Several approaches have been proposed in the literature for achieving such a goal; for details, the reader may be referred to [29] and the references therein.

Once computed, the feature vectors are compared to the model feature vectors representing each Arabic character. In the following section we present the case where only X-spectrum is being used. In Section 5, we test the performance of the algorithm when the feature vector is defined in terms of the elements of (i) $\boldsymbol{X}$, (ii) $\boldsymbol{Y}$ and (iii) both $\boldsymbol{X}$ and $\boldsymbol{Y}$. In the sequel, we will denote the feature vector consisting of both $\boldsymbol{X}$ and $\boldsymbol{Y}$ by $\boldsymbol{XY}$.

4. Classification

In order to ascribe the input character to one of the model characters, a minimum distance classifier is used. Specifically, we propose to classify $c(m, n)$ by minimizing the following cost function:

$$f_i = \sum_{k=1}^{p} [\,|X_i(2\pi k/N)| - |X(2\pi k/N)|\,]^2, \tag{3}$$

which is a minimum distance in the spectrum domain. $X_i(2\pi k/N)$ is the kth element of the model feature vector of character i. The classification criterion may be formulated as follows:

$$\text{declare } c(m,n) \text{ as model } i \text{ iff } f_i \leqslant f_j,\ i \neq j \text{ and}$$
$$j = 1, 2, \dots, L, \tag{4}$$

where L is the number of models of the used character set.

The cost function f_i as defined above is dc-invariant, since a constant added to the sequence $x(m)$ changes only the discrete-time Fourier coefficient $X(0)$ which is excluded from $\boldsymbol{X}$. Furthermore, the function f_i is invariant to a time shift in $x(m)$, since only the magnitude of $X(\omega)$ is being used; a signal which is shifted in time does not have the magnitude of its Fourier transform altered. The effect of a time shift on a signal is to introduce a phase shift in its transform which is a linear function of ω.

Let $X(k)$ and $X_i(k)$ be sequences that represent the elements of the unknown (input) data and the model feature vectors $\boldsymbol{X}$ and $\boldsymbol{X}_i$, respectively. Under the constraint that

$$\sum_{k=1}^{p} X^2(k) = \sum_{k=1}^{p} X_i^2(k)$$
$$= E, \text{ a fixed positive value,}$$

for all values of $i = 1, 2, \dots, L$, the function f_i can be rewritten as

$$f_i = 2E - 2\sum_{k=1}^{p} X(k)X_i(k). \tag{5}$$

It is evident from Eq. (5) that minimizing f_i corresponds to maximizing

$$E_i = \sum_{k=1}^{p} X(k)X_i(k). \tag{6}$$

Eq. (6) can be easily implemented using a filter with real impulse response $h_i(k)$ which is a time-reversed and delayed version of X_i, as shown by

$$h_i(k) = X_i(p + 1 - k). \tag{7}$$

With the received signal $X(k)$ used as the filter input, the resulting filter output $z_i(k)$ is defined by the convolution sum:

$$z_i(k) = \sum_l X(l)h_i(k - l)$$
$$= \sum_l X(l)X_i(p + 1 - k + l).$$

Sampling this output at $k = p + 1$, we get

$$z_i(p + 1) = \sum_{l=1}^{p} X(l)X_i(l) = E_i. \tag{8}$$

Fig. 3(b) shows a schematic diagram of the recognition algorithm. First, the 1-D sequence $x(m)$ is computed from the received character $c(m, n)$. Then, the FFT algorithm is applied to $x(m)$ to obtain its discrete Fourier transform $X(k)$. Only the magnitude of the elements corresponding to $k = 1, 2, \dots, p$ are retained and used for recognition. The received sequence $X(k)$ is then normalized to an energy level E and applied to a bank of filters $\{h_i(k)\}$, each matched to a template as defined in Eq. (4). The label of the template that maximizes the output at $k = p + 1$ is declared as the label of the recognized input character.

5. Performance evaluation

In order to demonstrate the effectiveness of the approach presented in this paper in recognizing Arabic characters, a series of tests was performed on a data set consisting of 49 samples of each character. The test data consist of laser output of IBM Naskh font size 12 printed on white paper. Each character sample is printed on one page with five rows each consisting of ten characters. These data were scanned using a scanner with a resolution of 300 pixels per inch. The scanned document was then transformed into a binary image having only two levels: 1 and 0, where 1 represents black pixels, and 0 represents white pixels. The individual characters are extracted from the binary image using the X- and Y-projections of the whole image, after identifying their coordinates in the image. Each character is then extracted into a 2-D array for further processing.

Model features were computed by averaging the feature vectors of 10 samples of each character. Table 2 shows the elements of X- and Y-features for the Seen-I (س) and Kaaf-I (ك) characters (I stands for Isolated). The parameter N was selected, in all experiments, such that $N = 128$.

Thirty-nine new samples were then used for testing. Table 3 shows the percentage of successful recognition rates when X, Y and XY features were used with different values of p. The confusion matrix of Table 4 shows the results of testing 1677 characters using the XY-features.

Based on the results obtained, the following conclusions can be drawn:

(1) There is not much improvement in the recognition rates for values of $p > 10$. However, the approach described in this paper is capable of achieving high recognition rates with a relatively small number of parameters. For example, when the X-feature was used with $p = 10$, a recognition rate of 99.05% was achieved. The 0.95% error in this case was found to be due to a misclassification of five characters, namely, Baa-M (ـبـ), Haa-B (هـ), Haa-E (ـه), Waw-E (ـو) and Yaa-E (ـي). Note that B stands for Beginning and E for End.

(2) The 1-D slice $X(\omega)$ preserves more information than does $Y(\omega)$. This is evident from both Table 3 and Fig. 4, where we can observe that, at

Table 2
X- and Y-features for Seen-I and Kaaf-I characters

SEEN-I (س)		KAAF-I (ك)	
X-features	Y-features	X-features	Y-features
0.656	0.596	0.565	0.594
0.496	0.49	0.456	0.432
0.289	0.34	0.303	0.219
0.109	0.186	0.139	0.077
0.097	0.118	0.021	0.185
0.131	0.179	0.115	0.252
0.106	0.233	0.175	0.243
0.084	0.242	0.204	0.197
0.131	0.212	0.22	0.183
0.171	0.157	0.231	0.208
0.168	0.095	0.231	0.217
0.138	0.04	0.208	0.188
0.129	0.01	0.160	0.13
0.151	0.033	0.099	0.085
0.157	0.05	0.057	0.086
0.125	0.057	0.071	0.095
0.064	0.057	0.095	0.081
0.008	0.048	0.101	0.049
0.049	0.036	0.091	0.017
0.058	0.028	0.079	0.023

Table 3
Character recognition results using the X-, Y- and XY- features with different number of features (2–20)

No. of used coefficients	Recognition rate % using X-features	Recognition rate % using Y-features	Recognition rate % using XY-features
2	51.04	43.77	76.74
4	81.81	72.15	96.66
6	94.63	81.45	99.52
8	96.78	88.85	99.94
10	99.06	88.01	99.94
12	98.99	87.54	99.94
14	98.87	91.29	99.94
16	99.05	91.00	99.94
18	99.11	90.70	99.94
20	99.22	90.28	99.94

Table 4
The results of testing 39 sets, using 1-D slices of the character spectrum. Recognition rate 99.94%

Name	Code	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
ALIF-E	1	39																																										
ALIF-I	2		39																																									
BAA-B	3			39																																								
BAA-M	4	1			38																																							
BAA-I	5					39																																						
HHAA-B	6						39																																					
HHAA-I	7							39																																				
DAAL-E	8								39																																			
DAAL-I	9									39																																		
RAA-E	10										39																																	
RAA-I	11											39																																
SEEN-B	12												39																															
SEEN-I	13													39																														
SSAAD-B	14														39																													
SSADD-M	15															39																												
SSAAD-I	16																39																											
TTAA-E	17																	39																										
TTAA-I	18																		39																									
AIN-B	19																			39																								
AIN-M	20																				39																							
AIN-E	21																					39																						
AIN-I	22																						39																					
FAA-B	23																							39																				
FAA-M	24																								39																			
FAA-I	25																									39																		
KAAF-B	26																										39																	
KAAF-M	27																											39																
KAAF-I	28																												39															
LAAM-B	29																													39														
LAAM-M	30																														39													
LAAM-I	31																															39												
MEEM-B	32																																39											
MEEM-M	33																																	39										
MEEM-I	34																																		39									
NOON-I	35																																			39								
HAA-B	36																																				33							
HAA-M	37																																					39						
HAA-I	38																																						39					
WAAW-B	39																																							39				
WAAW-I	40																																								39			
YAA-I	41																																									39		
YAA-E	42																																										39	
LAAMALIF-I	43																																											39

a fixed value of p, the recognition rates obtained using $X(\omega)$ are much higher than those obtained using $Y(\omega)$. This, however, can be attributed to the fact that the distance measure as defined in Eq. (3) has smaller values between the feature vectors defined in terms of $Y(\omega)$ than it would be if the feature vectors were defined in terms of $X(\omega)$. Fig. 5 shows the average distance between the feature vector of the ith character and the feature vectors of the other characters, when $X(\omega)$ *and* $Y(\omega)$ are being used with $p = 20$. In Table 5 we list the characters whose average distances from the other characters are within 0.1 and 0.2.

(3) Simultaneous use of $X(\omega)$ and $Y(\omega)$ contributes a slight improvement to the capability of the recognition algorithm for values of $p > 10$. Therefore, in practice, $Y(\omega)$ can be neglected if the computational complexity and storage requirements of the recognition system are of concern.

It is relevant at this point to mention that the recognition algorithm of this paper and that of [17] both extract features from the frequency domain.

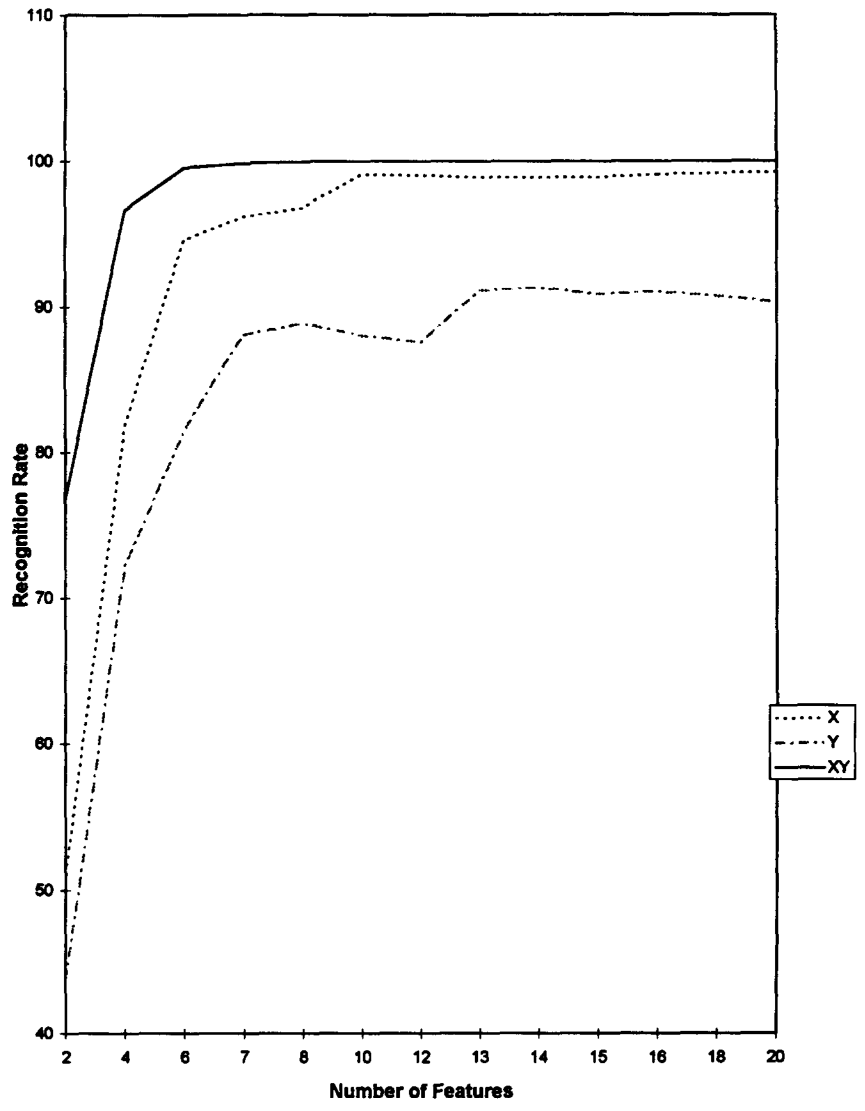


Fig. 4. The recognition rates using 2–20 features of X-, Y- and XY-feature vectors.

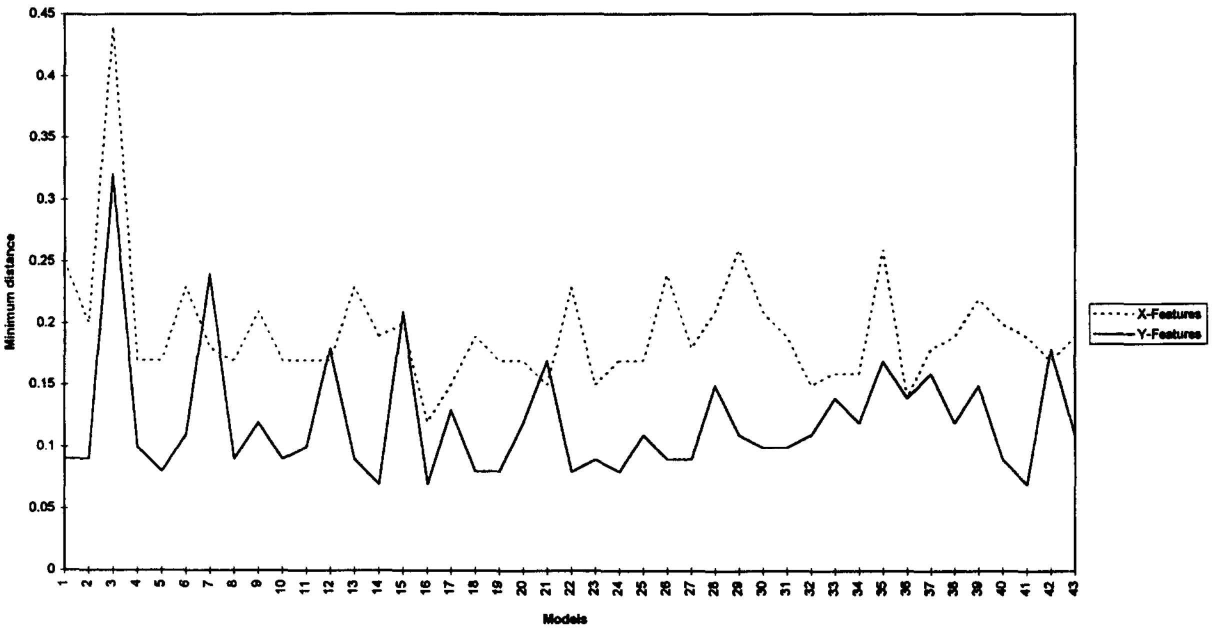


Fig. 5. Minimum distance between each model and other models using X- and Y-features.

In [17] a method based on Fourier descriptors was proposed and analyzed. Theoretical and experimental evidence available in the literature indicate that a Fourier descriptor based technique is a powerful tool for classifying closed contours [13, 21, 25].

For comparison purposes, the method of [17] which utilizes the coordinate values of the contour pixels of the character primary part to obtain the Fourier descriptors, are summarized in Appendix A and tested using our data. New tests of the method of [17] were carried out for comparison purposes with the proposed technique. The results are displayed in the confusion matrix in Table 6. Comparing the two confusion matrices of the techniques based on 1-D slices of the spectrum of the character projection on the X- and Y-axes and the one based on Fourier descriptor of the character primary contour indicates that the presented technique is superior to the Fourier descriptor based technique in terms of the recognition rate. A recognition rate of 99.94% for the 1-D slice technique using 10 features for the X- and Y-projections each was obtained compared to 97.5% for the Fourier descriptor technique using 10 Fourier descriptors. Both techniques are invariant to shifting. However, the Fourier descriptor technique is, in addition, invariant to scale and rotation.

The presented technique is proved to achieve very high recognition rate. Its recognition rate is higher than the majority of the techniques used for Arabic character recognition. The technique may be improved by using normalization of the input data to overcome the scaling limitation. Since the technique is used to recognize printed characters the possible range of rotation is small and may be handled by using one of the techniques for page correction and skew techniques to normalize the whole page, hence, producing characters with minial or no rotation.

Appendix A. Arabic character recognition using Fourier descriptors

This appendix summarizes the Fourier descriptor based technique for Arabic character recognition. For more details reference may be made to [17].

This method is based on using the character's closed boundary of the character primary part which is 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 parametric representation of the boundary coordinates [23]. A contour-tracing algorithm based on the algorithm of [24] is implemented. At every pixel, the neighboring pixels are traced in a sequence dependent on the previous move (chain code).

Table 5
Characters with an average distance of $\leqslant 0.1$ and $\leqslant 0.2$ from other characters (characters with average distances within the range are marked with *)

	Characters with average distance from other characters $\leqslant 0.1$		Characters with average distance from other characters $\leqslant 0.2$	
Character models	Using X-features	Using Y-features	Using X-features	Using Y-features
Alif-E		*	*	*
Alif-I				
Baa-M		*	*	*
Baa-B		*	*	*
Baa-E				*
Hhaa-B		*		*
Hhaa-I			*	
Daal-E		*	*	*
Daal-I				*
Raa-E		*	*	*
Raa-I		*	*	*
Seen-I			*	*
Seen-B		*		*
Ssaad-B		*	*	*
Ssaad-M		*		*
Ssaad-I			*	
Ttaa-E		*	*	*
Ttaa-I			*	*
Ain-M		*	*	*
Ain-B		*	*	*
Ain-E			*	*
Ain-I			*	*
Faa-M		*	*	*
Faa-B		*	*	*
Faa-I			*	*
Kaaf-M		*		*
Kaaf-B		*	*	*
Kaaf-I				*
Laam-B		*		*
Laam-M			*	*
Laam-I				*
Meem-M		*		*
Meem-B		*	*	*
Meem-I			*	*
Noon-I			*	*
Haa-B			*	*
Haa-M			*	*
Haa-E			*	*
Waaw-E			*	*
Waaw-I				*
Yaa-E			*	*
Yaa-I			*	*
LaamAlif				*

Table 6
The results of testing 39 sets, using 10 Fourier descriptors. Recognition rate 97.5%

Name	Code	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
ALIF-E	1	39																																										
ALIF-I	2		39																																									
BAA-B	3			39																																								
BAA-M	4		1		38																																							
BAA-I	5					39																																						
HHAA-B	6						39																																					
HHAA-I	7							39																																				
DAAL-E	8								39																																			
DAAL-I	9									39																																		
RAA-E	10										39																																	
RAA-I	11											39																																
SEEN-B	12												39																															
SEEN-I	13													39																														
SSAAD-B	14														39																													
SSADD-M	15															39																												
SSAAD-I	16																39																											
TTAA-E	17																	39																										
TTAA-I	18																		39																									
AIN-B	19																			39																								
AIN-M	20															4					35																							
AIN-E	21																					39																						
AIN-I	22																						39																					
FAA-B	23																3							36																				
FAA-M	24																				2				37																			
FAA-I	25																									39																		
KAAF-B	26																										39																	
KAAF-M	27																											39																
KAAF-I	28																												39															
LAAM-B	29																													39														
LAAM-M	30																														39													
LAAM-I	31																															39												
MEEM-B	32																																39											
MEEM-M	33																																	39										
MEEM-I	34																																		39									
NOON-I	35																																			39								
HAA-B	36																																				33		6					
HAA-M	37																																					39						
HAA-I	38																																				11		28					
WAAW-B	39																																							39				
WAAW-I	40																																								39			
YAA-I	41																																									24		15
YAA-E	42																																										39	
LAAMALIF-I	43																																											39

The Fourier descriptors used in this work were first proposed by Granlund [13] and further developed by Persoon et al. [25] and Nikravan et al. [21]. The normalized Fourier descriptors are invariant with respect to translation, rotation and scale of the characters. 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 $m = 0, 1, \dots, L-1$, 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. 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-axis 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 given by

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

The first 10 Fourier descriptors are computed from the contour of the primary part of the character. The Fourier descriptors of two closed curves which differ only in position, orientation and size with analogous starting points are identical [34]. In this work, a fixed scanning sequence of each character (i.e. from right to left and from top to bottom) is adopted. In addition, normalization is carried out by dividing the Fourier descriptor coefficients by the largest one (i.e. by the first one). Hence, the obtained Fourier descriptor coefficients are invariant to rotation, shifting and scale.

References

[1] B. Al-Badr and S. Mahmoud, "Survey and bibliography of Arabic optical text recognition", *Signal Processing*, Vol. 41, No. 1, January 1995, pp. 49–77.

[2] 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*, Vol. 974, San Diego, CA, 1988, pp. 330–336.

[3] H. AL-Yousefi and S. Udpa, "Recognition of handwritten arabic characters via segmentation", *Arab Gulf J. Scient. Res.*, Vol. 8, No. 2, 1990, pp. 49–59.

[4] A. Amin, "Machine recognition and correction of Arabic printed text", *IEEE Trans. Systems Man Cybernet.*, Vol. 19, No. 5, 1989, pp. 1300–1306.

[5] K. Bouhlila, M. K. Kamrouni and N. Ellouze, "Method of segmentation of arabic text image into characters", *First Kuwait Comput. Conf.*, Kuwait, 27–29 March 1989, pp. 442–446.

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

[7] F. El-Khaly and M. Sid-Ahmed, "Machine recognition of optically captured machine printed Arabic text", *Pattern Recognition*, Vol. 23, No. 11, 1990, pp. 1207–1214.

[8] S.H. El Ramly and M.A. El-Hamalawy, "A new font for Arabic character simplifies recognition procedure", *Proc. 14th Internat. Conf. on Statistics, Computer Science and Demographic Research*, Cairo, Egypt, March 1989, pp. 255–261.

[9] T.S. El-Sheikh and S.G. El-Taweel, "Segmentation of handwritten words", *Proc. 12th National Comput. Conf.*, Riyadh, Saudi Arabia, October 1990, pp. 389–402.

[10] M. Fakir and C. Sodeyama, "Recognition of Arabic printed scripts by dynamic programming matching method", *IEICE Trans. Inform. Systems*, Vol. E76-D, No. 2, February 1993.

[11] H. Goraine and M. Usher, "Printed Arabic text recognition", *Proc. 4th Internat. Conf. and Exhibition on Multilingual Computing* (Arabic and Roman Script), University of Cambridge, London, UK, April 1994, pp. 2.6.1–2.6.8.

[12] V.K. Govindan and A.P. Shivaprasad, "Character recognition – A review", *Pattern Recognition*, Vol. 23, No. 7, 1990, pp. 671–683.

[13] G.H. Granlund, "Fourier preprocessing for hand print character recognition", *IEEE Trans Comput.*, 1972, pp. 195–205.

[14] K.M. Hassibi, "Machine-printed Arabic OCR using neural networks", *Proc. 4th Internat. Conf. and Exhibition on Multilingual Computing* (Arabic and Roman Script), University of Cambridge, London, UK, April 1994, pp. 2.3.1–2.3.12.

[15] K. Jambi, "A system for recognizing handwritten arabic words", *Proc. 13th National Comput. Conf.*, Riyadh, November 1992, pp. 472–482.

[16] S. Kahan, T. Pavlidis and H.S. Baird, "On the recognition of printed characters of any font and size", *Pattern Recognition*, Vol. 9, No. 2, 1987, pp. 274–287.

[17] S.A. Mahmoud, "Arabic character recognition using Fourier descriptors and character contour encoding", *Pattern Recognition*, Vol. 27, No. 6, June 1994, pp. 815–824.

[18] D. McClelland, "OCR: Teaching your Mac to read", *Macworld*, November 1991, pp. 169–178.

[19] G. Nagy, "At the frontiers of OCR", *Proc. IEEE*, Vol. 80, No. 7, July 1992, pp. 1093–1100.

[20] A.M. Nazif, A system for the recognition of printed Arabic characters, M.Sc. Thesis, Faculty of Engineering, Cairo University, Cairo, Egypt, 1975.

[21] B. Nikravan et al., "An experimental evaluation of normalized Fourier descriptors in the identification of simple engineering objects", *Comput. Indust.*, Vol. 13, 1989, pp. 37–47.

[22] F. Nouboud and R. Plamondon, "On-line recognition of handprinted characters: Survey and beta tests", *Pattern Recognition*, Vol. 23, No. 9, 1990, pp. 1031–1044.

[23] T. Pavlidis, *Structural Pattern Recognition*, Vol. 1, Springer, New York, 1977.

[24] T. Pavlidis, *Algorithms for Graphics and Image Processing*, Computer Science Press, Rockville, MD, 1982.

[25] E. Persoon and K.-S. Fu, "Shape discrimination using Fourier descriptors", *IEEE Trans. SMC*, Vol. SMC-7, No. 3, March 1977, pp. 170–179.

[26] I.T. Phillips, Su Chen and Robert M. Haralick, "CD-ROM document database standard", *Proc. Internat. Conf. on Document Analysis and Recognition*, Nangano, Japan, October 1993, pp. 478–482.

[27] R. Ramsis, S.S. El-Dabi and A. Kamel, "Arabic character recognition system", *IBM Kuwait Scientific Center*, Report No. KSC027, January 1988.

[28] J.W. Smith and Z. Merali, *Optical Character Recognition*, The British Library, Wetherby, West Yorkshire LS23 7BQ, UK, 1985.

[29] H.V. Sorensen and C.S. Burrus, "Efficient computation of the DFT with only a subset of input or output points", *IEEE Trans. Signal Process.*, Vol. 41, No. 3, March 1993.

[30] C.C. Tappert, C.Y. Sen and T. Wakahara, "The state of the art in on-line handwriting recognition", *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 12, No. 8, August 1990, pp. 787–808.

[31] M.F. Tolba and E. Shaddad, "On the segmentation and recognition of printed Arabic characters", *Proc. 2nd Conf. on Arabic Computational linguistics*, Kuwait, November 1989, pp. 490–507.

[32] T. Wakahara, H. Murase and K. Odaka, "On-line handwriting recognition", *Proc. IEEE*, Vol. 80, No. 7, July 1992, pp. 1181–1194.

[33] E.M. Welch, "Can you read this?", *OCR Software. MacUser*, Vol. 9, No. 8, November 1993, pp. 169–178.

[34] C.T. Zahn and R.Z. Roskies, "Fourier descriptors for plane closed curves", *IEEE Trans. Comput.*, Vol. C-21, No. 3, March 1972, pp. 269–281.