Arabic Handwritten Alphanumeric Character Recognition using Fuzzy Attributed Turning Functions

Mohammad Tanvir Parvez and Sabri A. Mahmoud
Information and Computer Science Department
King Fahd University of Petroleum & Minerals (KFUPM), Dhahran 31261, Saudi Arabia.
Email: {tparvez, smasaad}@kfupm.edu.sa

Abstract

In this paper, we present a novel method for recognition of unconstrained handwritten Arabic alphanumeric characters. The algorithm binarizes the character image, smoothes it and extracts its contour. A novel approach for polygonal approximation of handwritten character contours is applied. The directions and length features are extracted from the polygonal approximation. These features are used to build character models in the training phase. For the recognition purpose, we introduce Fuzzy Attributed Turning Functions (FATF) and define a dissimilarity measure based on FATF for comparing polygonal shapes. Experimental results demonstrate the effectiveness of our algorithm for recognition of handwritten Arabic characters. We have obtained around 98% accuracy for Arabic handwritten characters and more than 97% accuracy for handwritten Arabic numerals.

1. Introduction

Commercial Optical Character Readers (OCRs) for Latin scripts emerged in the 1950s. Today, the character and document recognition technology has advanced significantly, providing commercial products for the recognition of texts [7]. However, there is no operationally accurate Arabic handwritten OCR commercial product available in the market [5].

The recognition of Arabic handwriting presents some unique challenges and benefits to the researchers [5]. Arabic letter shapes are context dependent and are written cursively both in print and handwriting. In addition, researchers have to consider different writing styles and issues related to the quality of scanned documents (like noise from scanning process, ink problem etc.). In Arabic handwritten text, the characters join at the writing line, unlike other scripts like Latin, and hence more simplified.

There are a number of works related to Arabic handwritten character recognition. Abuhaiba et al. proposed a set of character graph models (CGM) to recognize isolated handwritten letters [2]. Skeletons of characters are generated using fuzzy clustering algorithm. The skeleton segments and their fuzzy directions are used as features. Each model is represented as a state machine with transitions corresponding to fuzzy directions of segments in the character skeleton and with additional "fuzzy" constraints to distinguish some characters. Adnan Amin [3] used inductive learning to recognize hand printed characters. Stroke types and relationships between the strokes are extracted from the characters to generate by induction the first-order Horn clauses representing the characters. These Horn clauses are then used for classification.

Mozaffari et al. [12] proposed a method for Farsi/Arabic handwritten zip code recognition. The skeleton of the numeral is decomposed into primitives. Features are extracted from these primitives and a classifier based on nearest neighbor is used for recognition. Mezghani et al. [11] presented a method for Bayes classification of online Arabic characters. They used histograms of tangent differences and Gibbs modeling of the class-conditional probability density functions. Abandah et al. [1] used principal component analysis to select best subset of features out of a large number of extracted features. They utilized both parametric and non-parametric classifiers to determine the best set of features.

In this paper, we present a novel approach for the recognition of handwritten Arabic alphanumeric characters. The character image is binarized and smoothed. Then the character contour is extracted and represented by a polygonal approximation. The directions and length features of the polygonal approximation are extracted and character models are built. The edges of the polygonal approximation are modeled as fuzzy directional edges. A classifier based on fuzzy logic and turning angle functions is utilized in the recognition phase. Experimental results demonstrate the effectiveness of our algorithm for recognition of unconstrained Arabic handwritten alphanumeric characters.

The rest of the paper is organized as follows. Section 2 presents our algorithm for recognition of handwritten Arabic alphanumeric characters. Section

3 details the experimental results. The conclusions are presented in Section 4.

2. Recognition System

The general outline of our algorithm for the recognition of unconstrained handwritten Arabic characters is shown in Figure 1. The image of the character sample first goes through the preprocessing steps, like binarization, smoothing etc. Then the contour of the character image is extracted and polygonal approximation of the contour is constructed. The directions of the polygonal approximation are modeled as fuzzy directional edges. Directions and length features are used to build character models. In the recognition phase, a fuzzy similarity measure based on turning angle functions is introduced for classification. In the following subsections, we describe each of these steps in detail.

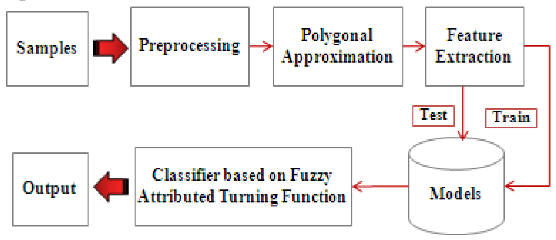


Figure 1: Overview of the Arabic handwritten character recognition system.

2.1. Preprocessing

The grayscale character image is first converted into binary (black and white) image. The algorithm used for this purpose is given by Otsu [13]. The binary image is then smoothed using statistical average based smoothing [10]. Smoothing the character image removes spurious strokes or noise from the image.

2.2. Polygonal Approximation

Once the character image is binarized and smoothed, the image is converted into a more concise representation. Skeletonization or thinning and contour of the image are the two common representations. Although many researchers have used thinning, it may incur some difficulties like mislocalization of features, ambiguities, 'hairs' (small spurious lines) etc [9]. The contour approach avoids these difficulties, preserves the shape information and is faster to generate. In the proposed method, we utilize the contour of the character for feature extraction. Contour extraction algorithm can be found in [14]. The extracted contour is then represented by a polygonal approximation.

Polygonal approximation or dominant points schemes for contour representation are useful in handwriting recognition for several reasons. Information on the text curve is concentrated at the corner (dominant) points; a polygonal approximation is a more compact representation of the character contour; polygonal approximations can avoid small spurious lines / noise in the handwritten text which in turn can improve the recognition rate of the system.

Figure 2 shows our novel approach for constructing the polygonal approximation of the character contour. The algorithm first selects an initial set of dominant points from the contour $C = P_i$ (x_i, y_i), $i = 1, 2, \ldots, n$. Let c_i, $i = 1, 2, \ldots, n$ be the Freeman chain code for n points of C. A point P_i is called a *break point* if c_i and c_{i+1} are different. Here, c_{n+1} is same as c_1. The set of all break points in C are selected as the initial set of dominant points. The approximation defined by the break points is affected by noise on the contour. Thus it is required to remove redundant points from the initial set of dominant points, which is done as follows.

Let P_i. P_j and P_k be three consecutive dominant points on C. The *left-support region* of P_j consists of the points $\{P_i, P_{i+1}, \ldots, P_{j-1}\}$. Similarly, the *right-support region* of P_j consists of the points $\{P_{j+1}, P_{j+2}, \ldots, P_k\}$. The total number of points in the left and right-support regions of P_j is called the *strength* of P_j. Dominant points are sorted based on their *strength*, then based on the distance from the centroid of the contour C. Then, for each dominant point P_j from the sorted list (weakest point first), P_j is suppressed if the perpendicular distance d_{per} from P_j to the line joining P_i and P_k is less than some predefined threshold d_{col} and the following two constraints on P_i. P_j and P_k are satisfied:

Constraint 1: The triangle formed by the three points P_i. P_j and P_k is an acute triangle, and

Constraint 2: For each dominant point P_l other than P_i. P_j and P_k, the minimum distance from P_l to the line segment joining P_i and P_k is greater than d_{col}.

```
Algorithm FindPolygonalApproximation
  - The Candidate Set (CS) of dominant points consists of all
    break points in contour C.
  - Initial value for $d_{col}$ = 0.5.
  - Repeat:
      o Suppress redundant dominant points from CS. Resulting
        set of dominant points is called Reduced Candidate
        Set (RCS) and denoted by $RCS_{dcol}$.
      o Output the approximation defined by $RCS_{dcol}$
      o CS $\leftarrow$ $RCS_{dcol}$
      o Increase the value for $d_{col}$.
  - Until a terminating condition.
```

Figure 2: Algorithm for polygonal approximation.

This operation of removing redundant points is called *constrained collinear-points suppression* (*CCS*). Our algorithm iteratively applies this suppression technique until some terminating condition is satisfied. This condition is defined as follows. Let $V^{(k)}$ be the set of dominant points retained at iteration k of *FindCutPoints*. We terminate `FindPolygonalApproximation` if $V^{(k)} = V^{(k+1)}$, for some $k \geq 1$.

Figure 3 illustrates the application of *FindPolygonalApproximation* for the Arabic character Twaa (ط). Note that, the number of dominant points is reduced gradually as the value for the threshold d_{col} is increased.

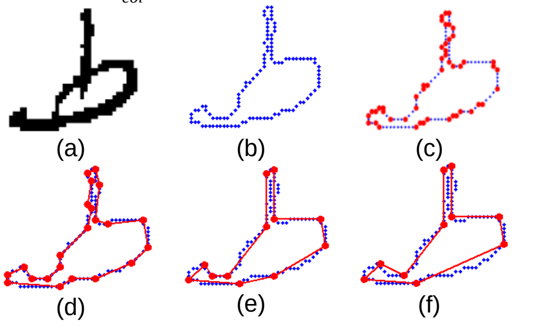


Figure 3: Illustration of the algorithm *FindPolygonalApproximation*: (a) Arabic ligature Twaa (ط), (b) contour after smoothing, (c) break points of (b) and (d-f) polygonal approximations of (b) defined by $RCS_{d_{col}}$ where d_{col} = 1.0, 1.5 and 2.0.

2.3. Modeling of Characters

Let $D_i = (x_i, y_i)$, i=1, 2, .., n_d be the sequence of dominant points of the polygonal approximation of the character contour C. These points define a polygon P of n_d edges, where each edge d_i of P is a vector D_iD_{i+1} with length $l_i = |d_i|$, where $d_i = d_{i+nd}$. A suitable representation of this polygon P *turning function* [4]. The turning function, or cumulative angle function, $\Theta_A(s)$ of a polygon or polyline A gives the angle between the counterclockwise tangent and the x-axis as a function of the arc length s. Figure 4 illustrates the turning function for the Arabic character Daal (د).

The dissimilarity measure for two turning functions Θ_A and Θ_B is defined as follows. The two polygons A and B are scaled so that they have unit length perimeters. A dissimilarity measure for A and B is the L_p metric applied to Θ_A and Θ_B: $d(A,B) = \left(\int_0^1 |\theta_A(s) - \theta_B(s)|^p \, ds \right)^{1/p}$. In practice, the measure $d(A,B)$ is evaluated as follows. For L_2 metric, the integral $\int_0^l |\theta_A(s) - \theta_B(s)|^2 \, ds$ is computed by adding up the value of the integral within each *strip* defined by a consecutive pair of discontinuities in $\theta_A(s)$ and $\theta_B(s)$ (Figure 5). The integral within a strip is computed as: $w_s \times d_s^2$, where w_s is the width of the strip and $d_s = |\theta_A(s) - \theta_B(s)|$. Note that d_s is constant within a strip.

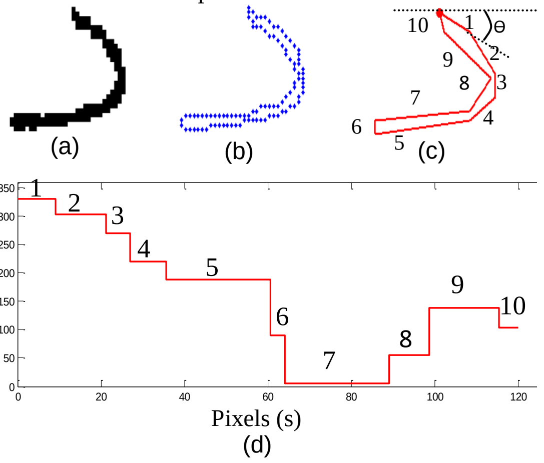


Figure 4: Illustration of turning function: (a) Arabic character Daal (د), (b) contour of (a) after smoothing, (c) polygon defined by the dominant points of (b) and (d) the turning function representation of (c).

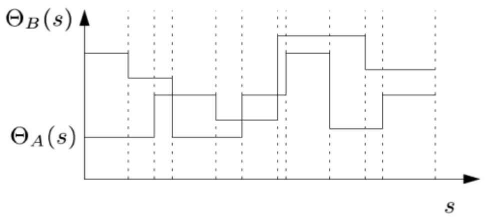


Figure 5: The rectangular strips formed by the functions $\theta_A(s)$ and $\theta_B(s)$.

However, we found that the above definition of the dissimilarity measure $d(A,B)$ perform poorly when used for recognition of unconstrained handwritten characters. The weakness arises from the way d_s is computed: it is simply the difference in two turning functions within a strip. In reality, d_s denotes the difference in writing directions within a strip. Due to the different writing styles, the computation of d_s should tolerate some level of deviations in writing directions. To alleviate these problems, we propose *Fuzzy Attributed Turning Functions*.

2.4. Fuzzy Modeling of Characters

In modeling the character contour C, we describe the vector d_i as a fuzzy direction. Fuzzy directions are fuzzy sets that have membership functions similar to those of fuzzy numbers that are characterized by possibility distributions. The fuzzy distributions [8] used in character modeling are a) *s-numbers*: (p/β), where p is the left peak point and β is the length of the transition interval (bandwidth) from π_x = 0 to 1; b) *z-numbers*: $(p\backslash\beta)$, where p is the right peak point and β is the bandwidth; and c) *s/z numbers*: $(p_1/\beta_1;\ p_2\backslash\beta_2)$, which is the intersection of the possibility distribution of an s-number and a z-number. Here the left peak point of s-number (p_1)

lies to the left of the right peak point the *z*-number (p_2).

In the modeling of Arabic handwritten characters, a restricted set of *s/z* numbers, called π-numbers, are used to model the directions d_i. These numbers are denoted by $\pi = (p_1/\beta_1;\Theta;\ p_2\backslash\beta_2)$, where $p_1 = \Theta - \gamma_1$, $\gamma_1 > 0$ and $p_2 = \Theta + \gamma_2$, $\gamma_2 > 0$. Figure 6 illustrates the concept of π-numbers. In a π-number, the membership value is 1 in $[p_1, p_2]$, and decreases linearly in $(p_1, p_1{-}\beta_1]$ and $(p_2, p_2{+}\beta_2]$. This simple modeling of directions can effectively handle variations in strokes.

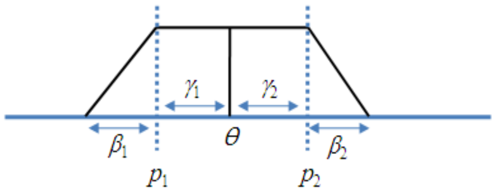


Figure 6: Illustration of a π-number.

A polygonal approximation of the Arabic character Daal (د) of Figure 7(a) is shown in Figure 7(c). This approximation contains 13 edges and each of these directions is modeled as a π-number. For example, the anticlockwise direction with respect to *x*-axis for the edge marked 1 is 315°. This angle can be mapped to a π-numbers as (300°/15°;315°;330°\15°). For this fuzzy number, any direction *d* that falls between 300° and 330° will have a membership value of 1.

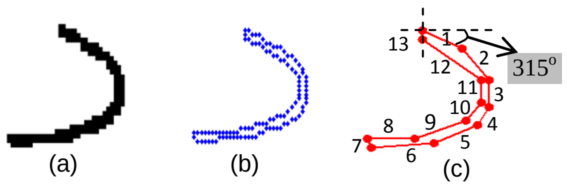


Figure 7: Illustration of directions in a polygonal approximation.

Sometimes it is useful to have Θ, in a π-number $(p_1/\beta_1;\Theta;\ p_2\backslash\beta_2)$, take value from a finite set of directions S_D, called *standard writing directions*. The set S_D can be regarded as quantization in stroke directions. This enables us to use limited number of models for each character with sufficient coverage for our application.

With this definition of a π-number, we now describe a novel fuzzy modeling of handwritten characters. We call $\Phi_A(s)$ as *Fuzzy Attributed Turning Function* (FATF), where the angles in $\Phi_A(s)$ are modeled as π-numbers $(p_1/\beta_1;\Theta;p_2\backslash\beta_2)$. The dissimilarity measure $d(A,B)$ for two FATF $\Phi_A(s)$ and $\Phi_B(s)$ is defined as follows: $d(A,B) = \left(\int_0^l |\Phi_A(s) - \Phi_B(s)|^p\, ds\right)^{1/p}$, where l is the length of the perimeter of the scaled polygons.

Now for L_2 metric, the integral $\int_0^l |\Phi_A(s) - \Phi_B(s)|^2 ds$ is computed as follows. Assume that, within a strip $\Phi_A(s) = \varphi_1$ and $\Phi_B(s) = \varphi_2$. Here, φ_1 is represented by a π-number $(p_1/\beta_1;\Theta;\ p_2\backslash\beta_2)$. Let m_{φ_2} be the membership value of φ_2 in the fuzzy direction for φ_1. Since d_s is used as the dissimilarity measure for writing directions, d_s is taken as $(1 - m_{\varphi_2})$. Therefore,

$$d_s = \begin{cases} 0, & \varphi_2 \in [\Theta - \gamma_1, \Theta + \gamma_2] \\ 1, & \varphi_2 > (\Theta + \gamma_2 + \beta_2) \text{ or } \varphi_2 < (\Theta - \gamma_1 - \beta_1) \\ 1 - \frac{\varphi_2 - p_2}{\beta_2}, & p_2 < \varphi_2 < p_2 + \beta_2 \\ 1 - \frac{p_1 - \varphi_2}{\beta_1}, & p_1 - \beta_1 < \varphi_2 < p_1 \end{cases}$$

In Figure 8, turning functions of two different samples of the Arabic character Daal (د) are shown in the same scale. The directions of the first Daal in Figure 8(a) are marked with a suffix *a* and for the second Daal in Figure 8(c) with suffix *b*. As can be seen in Figure 8(e), the measure $d(A,B)$ assumes that the differences in directions for Strip 1 and 2 are high because it blindly compute d_s as the absolute difference in direction angles. However, Figure 8 (b) and (c) clearly show that directions 1a and 1b-2b refer to the same segment of the character contour and should be treated as 'close'. Directions 1b and 2b differ from 1a because of the variations in writing styles. These variations must be tolerated in the dissimilarity measure. The proposed measure for FATF effectively achieves this goal by modeling the directions as fuzzy sets. This is illustrated pictorially by zooming Strip 1 and 2 in the upper-right box in Figure 8 (e). Here, the direction 1a is a 'band' of directions, effectively accommodating variations in strokes due to different writing styles. Due to these fuzzy descriptions of the directional angles, directions 1b and 2b are considered as 'close' to the direction 1a in FATF.

The dissimilarity measure $d(A,B)$ can be thought of utilizing local features on the character contour. There are also some global features which we need to consider at the recognition phase. These include length of the perimeter and the number of nodes in the polygonal approximations, presence of holes and dots in the character image etc. In addition, the nature of π-numbers incorporates a certain level of rotational invariance as rotation in Euclidean space is equivalent to vertical shifts in turning functions.

3. Experimentations

Two databases are used for experimentations: a database of handwritten Arabic characters and a database of handwritten Arabic numerals. In our experimentations, a nearest neighbor (NN) classifier based on fuzzy attributed turning function (FATF) is used. For simplicity, all the π-numbers in FATF have $\gamma_1 = \gamma_2 = 15°$, $\beta_1 = \beta_2 = 20°$ and standard directions $S_D = 20i$, $i = 0, 1, \ldots, 17$. Polygonal approximation of each character sample is obtained by restricting the highest value for d_{col} to 1.5.

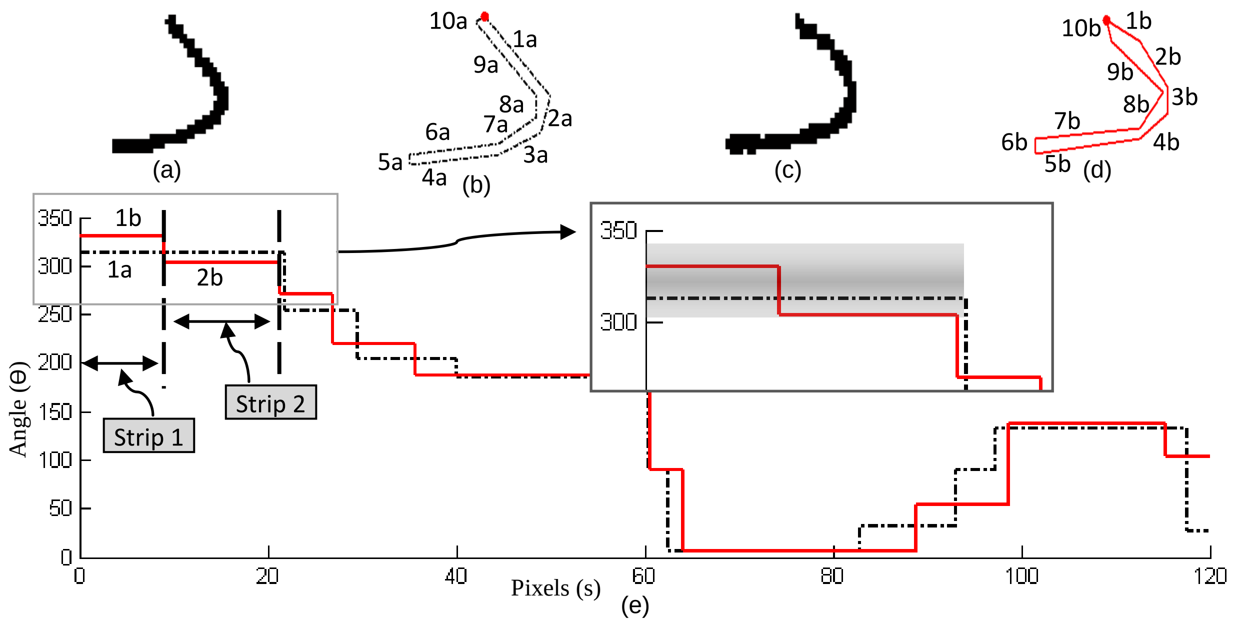


Figure 8: Illustration of turning functions for two samples of Arabic character Daal (د).

3.1 Character Recognition

The proposed algorithm is trained and tested using a database of 1948 samples of unconstrained handwritten Arabic characters written by 4 writers [2]. The database contains the basic character shapes without dots, comprising a total of 51 shapes (each character can have up to 4 shapes depending upon the context). The database is divided into 70% for training and 30% for testing (for each writer).

Table 1 shows the recognition rates obtained by our algorithm in top *n* cases, where *n* = 1,2,…,5. As can be seen in Table 1, more than 97% success rate is obtained for all four writers when the proposed FATF with 1-NN is used for classification. However, recognition based on turning angle function (TF) gives much lower accuracy, with a maximum of 74.83% recognition rate for Writer 4 using 1-NN. Thus the proposed FATF based classifier is much more successful than traditional turning functions in recognition of unconstrained handwritten Arabic alphabetic characters.

3.2 Arabic Numerals Recognition

The proposed algorithm is also tested with a database of Arabic numerals called ADBase [6]. The ADBase is composed of 70,000 digits written by 700 writers. Each writer wrote each digit (from ‘0’ to ‘9’) ten times. The database is partitioned into two sets: a training set (60,000 digits samples) and a test set (10,000 digits samples). An average accuracy of 97.18% is obtained for this database. The confusion matrix and the recognition accuracies for each numeral are shown in Table 2.

As can be seen from Table 2, the majority of the errors arise from the confusions between 0 and 5, 0 and 1, 3 and 2, 5 and 2, and 9 and 2. After analyzing the erroneous cases manually, we have found that around 41% of the error occurred due to the bad handwriting, which is difficult to be recognized even by humans. Further 22% of the error occurred due to broken digits. Figure 9 shows some examples of misclassified digits.

Table 1: Recognition accuracies in percentage for the four writers using FATF and TF.

Writer	FATF					TF				
	Top 1	**Top 2**	**Top 3**	**Top 4**	**Top 5**	**Top 1**	**Top 2**	**Top 3**	**Top 4**	**Top 5**
1	97.97	98.65	98.65	98.65	99.32	71.62	75.68	79.05	81.76	85.14
2	97.84	98.56	98.56	98.56	98.56	67.63	73.38	82.73	82.73	89.21
3	97.90	97.90	97.90	98.60	99.30	70.63	80.42	86.01	88.11	89.51
4	97.20	97.90	98.60	98.60	98.60	74.83	82.52	84.62	85.31	88.81

Table 2: Confusion matrix and recognition accuracies for Arabic handwritten numerals.

Digits	٠	١	٢	٣	٤	٥	٦	٧	٨	٩	Accuracy
٠	925	17	0	0	0	56	0	0	2	0	92.50%
١	1	984	1	3	0	1	0	5	3	2	98.40%
٢	1	4	988	2	1	2	0	0	2	0	98.80%
٣	0	6	14	964	0	0	1	7	6	2	96.40%
٤	0	4	12	0	978	1	3	0	0	2	97.80%
٥	8	3	15	1	2	955	0	4	6	6	95.50%
٦	0	6	1	0	1	0	981	0	0	11	98.10%
٧	0	7	0	0	0	2	0	991	0	0	99.10%
٨	1	3	1	0	0	0	0	0	990	5	99.00%
٩	0	9	14	2	1	1	5	0	6	962	96.20%

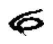
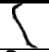
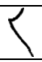
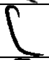
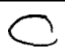
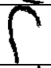
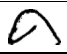
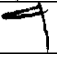
Image	Actual Digit	Digit Found	Image	Actual Digit	Digit Found
	٠	٥		٤	١
	٣	٢		٤	٢
	٥	٢		٢	٨
	٨	٩		٦	٩

Figure 9: Examples of misclassified digits.

4 Conclusions

We have proposed a novel method utilizing polygonal approximations and fuzzy directional edges for recognition of handwritten Arabic characters. We also introduce an effective dissimilarity measure for comparing polygonal shapes which can be utilized in shape analysis and retrieval systems. The authors are extending the proposed technique to handwritten text recognition. A possible drawback of the proposed method is the high computational complexity. However, the authors are exploring template reduction techniques to reduce the complexity.

Acknowledgement

We acknowledge King Fahd University of Petroleum & Minerals (KFUPM) for partially supporting this research work.

References

[1] Abandah, G.A., Younis, K. S. and Khedher, M. Z.: Handwritten Arabic character recognition using multiple classifiers based on letter form, In Proc. 5th IASTED Int'l Conf. on Signal Processing, Pattern Recognition, & Applications (SPPRA 2008), 2008.

[2] Abuhaiba, I.S.I., Mahmoud, S.A., and Green, R.J., Recognition of handwritten cursive Arabic characters, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 16, pp. 664–672, 1994.

[3] Amin, A.: Recognition of hand-printed characters based on structural description and inductive logic programming, Pattern Recognition Letters, vol. 24, no. 16, pp. 3187–3196, 2003.

[4] E. Arkin, P. Chew, D. Huttenlocher, K. Kedem, and J. Mitchel. An efficiently computable metric for comparing polygonal shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(3), pp. 209–215, 1991.

[5] Cheriet, M.: Visual recognition of Arabic handwriting: challenges and new directions, Arabic and Chinese Handwriting Recognition, LNCS 4768, pp. 1–21, 2008.

[6] El-Sherif, E., Abdelazeem, S.: A two-stage system for Arabic handwritten digit recognition tested on a new large database. International Conference on Artificial Intelligence and Pattern Recognition (AIPR-07), Orlando, FL, USA, pp. 237–242, 2007.

[7] Fujisawa, H.: Forty years of research in character and document recognition – an industrial perspective, Pattern Recognition 41, pp. 2435–2446, 2008.

[8] Kandal, A.: Fuzzy Mathematical Techniques with Applications, Reading MA, Addison-Wesley, pp. 38–40, 1986.

[9] Lorigo, L. and Govindaraju, V.: Offline Arabic handwriting recognition: a survey, Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 28, pp. 712–724, 2006.

[10] Mahmoud, S. A.: Arabic character recognition using Fourier descriptors and character contour encoding. Pattern Recognition 27, pp. 815–824. 1994.

[11] Mezghani, N., Mitiche, A. and Cheriet, M,: Bayes classification of online Arabic characters by Gibbs modeling of class conditional densities, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 7, pp. 1121–1131, 2008.

[12] Mozaffari, S., Faez, K., and Ziaratban, M.: Structural decomposition and statistical description of Farsi/Arabic handwritten numeric characters, Proc. Int'l Conf. Document Analysis and Recognition, pp. 237–241, 2005.

[13] Otsu, N.: A threshold selection method from gray-level histograms, IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC–9 , no. 1, pp. 62–66, 1979.

[14] Pavlidis, T.: Algorithms for Graphics and Image Processing, Computer Science Press, Rockville, Maryland, 1982.