Online Arabic Handwriting Digit Recognition

Maha El Meseery 1, Hany Ahmed 2, Sherif Abdel Azeem 3

Electrical Engineering Department

American University in Cairo

Cairo, Egypt

Email: 1melmeseery@aucegypt.edu, 2 eng\_hanyahmed@hotmail.com, 3shazeem@aucegypt.edu

**Abstract :** The two main contributions for this paper are presenting a large Arabic On-line Digits dataset and implementing an efficient on-line digit recognition system. In the dataset, we collected a new Arabic on-line digit from 300 writers. The paper also presents an on-line recognition system used to classify the user input strokes into digits. The system uses a combination of temporal and spatial features to recognize Arabic digits. The result shows the benefit of combining both the temporal and spatial features in the recognition rate. The system achieved 98.9 % which is higher than other comercial systems.

**Keywords:** On-line Handwriting Recognition; Handwritten Arabic digits; offline Handwriting Recognition; Two stage classifiers;

**1** **Introduction**

The field of on-line handwriting recognitions is one of the most developed research area that can immensely improve human computer interaction. Real-time on-line systems are currently under high demand in due to the hand held and PDA devices revolution. On-line handwriting system uses a tablet or an input device that traces the user movement. In those systems the recognizer stores a series of x and y coordinates that represents the user pen path.

Even though Arabic handwriting has recently gained a lot of attention [?], the problem of on-line Arabic digits has been neglected for quite some time now. Most systems working on Arabic handwriting either focus on off-line Arabic character and digit recognition problems or only on-line Arabic character recognition problem . Specialized Arabic on-line Arabic digits systems are extremely rare. In [?] developed a method based on fractal theory and test them on on-line and offline Farsi digits. Most handwriting recognition systems are divided into three main phases; pre-processing, feature extraction and finally classification. Pre-processing phase is mainly concerned with removing device noise from user’s strokes. The goal of the feature extraction phase is to produce features that possess discriminating power to differentiate between the different classes. Current on-line handwriting recognition systems depend on either on-line or off-line features. In off-line features, the digits or characters are recognized using spatial features produced from the image of the character [?, ?]. In on-line features,the temporal and order information are used. For example, various directional and pen trajectories features were used in on-line handwriting systems [?, ?]. Other on-line systems were based on stroke and dominant points features [?, ?]. Recently, some system tried to combine both spatial and temporal features to achieve better accuracy [?]. The final classification step depends on using a classifier to label the input feature vector into different characters or digits.

The first contribution of this paper is presenting an Arabic On-line Digits Dataset collected form 300 writers. The second main contribution of this paper is introducing an efficient system to recognize on-line Arabic digits. The system is divided into pre-processing stage where the users use a tablet or input device to write Arabic digits to the system. After pre-processing, the features extraction stage computes a set of both on-line and off-line features from the users' strokes. The final stage uses the computed feature vector to classify the input into one of the Arabic digit. Table 1 shows Arabic digits from 0 to 9.

Table 1: Arabic Printed and Handwritten Digits.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Latin Equivalent | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Printed | P%20Farsi%200 | P%20Farsi%201 | P%20Farsi%202 | P%20Farsi%203 | P%20Arabic%204 | P%20Arabic%205 | P%20Arabic%206 | P%20Farsi%207 | P%20Farsi%208 | P%20Farsi%209 |
| Typical Handwritten | H%20Farsi%200%20other | H%20Farsi%201 | H%20Arabic%202 | H%20Arabic%203 | H%20Arabic%204 | H%20Arabic%205 | H%20Arabic%206 | H%20Arabic%207 | H%20Arabic%208 | H%20Arabic%209 |
| Other Writing Style | -- | -- | -- | H%20Arabic%203%20other | -- | -- | -- | -- | -- | -- |

**2** **Arabic On-line Digit Dataset : AOD**

Even though there is more than one Arabic character dataset there has not been a large on-line Arabic Digit dataset. Most on-line recognition datasets does not include a large Arabic digit dataset. There are various large offline Arabic digits[?, ?] and Latin digits[?] datasets but there is only small on-line Farsi digits (similar to Arabic digits) [?]. Due to the lack of on-line Arabic digits dataset we worked to collect a Large On-line Arabic Digit Dataset (OAD). To ensure including different writing styles, the database was gathered from 300 writers varying over different age groups with more than  in the age group between 20 and 35. Our youngest writer is 11 years old and our oldest is 70 years old. More than  are right handed and around  of the writers are females. Each writer was asked to write an average of 10 sample per digits with no constrain on the number of stroke for each digits or the writing style in orientation or size. We collected average of  samples which means  sample per digit. The data set was collected using DigiMemo  Inch on-line Dataset. Then each digit is labeled and ground truth data is stored along with the strokes information. The AOD is intended to be a benchmark for Online Arabic handwritten digit recognition research and, henceforth, we have made it available for free at: http://code.google.com/p/auc-recognition

Due to Irregular movement of the hand, and the inaccuracy of the digitalization process, the pre-processing step is influential on the rest of the system. Pre-processing goes through two stages: Re-sampling and smoothing.

**3.1** **Resampling**

Linear interpolation[?] used to solve irregularity in the distribution of points and large distances between consecutive points, as shown in Fig 1.

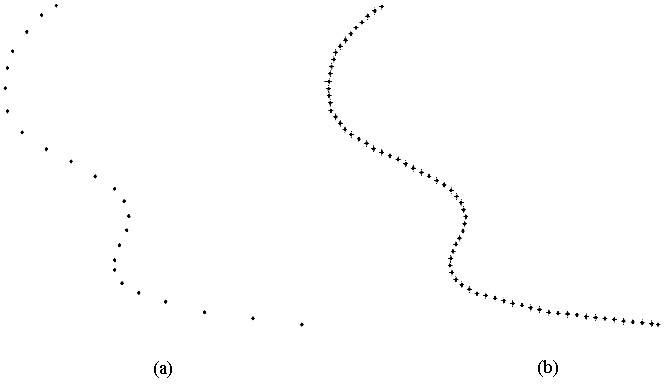


Figure 1: (a) Before (B) After.

We noticed that some writers move their hands up while writing the digit which mean more than one stroke in a digit, so we need to group these stokes together then concatenate the strokes into a continuous flow,as shown in Fig. 1.

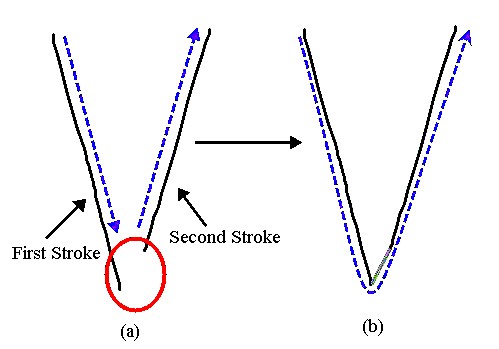


Figure 2:

As shown in figure 2 , the linear interpolation solved the problem of more than one stroke in a digit . Another problem is writing more than one stroke in a digit and change the direction of writing flow in the second stroke , as shown in Fig 3.

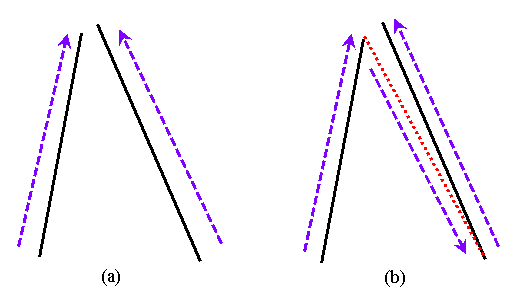


Figure 3: (a) Example of changing direction , (b) direct interpolation will cause in problem

The previous problem can't be solved using interpolation directly , the first step to solve this problem is to locate the pen up and pen down location, the end and the start of the second stroke is tested to determine the direction of stroke. If the user changed the writing direction of the second stroke; the points of the second stroke will be reversed before Interpolating the points,as shown in Fig 4.

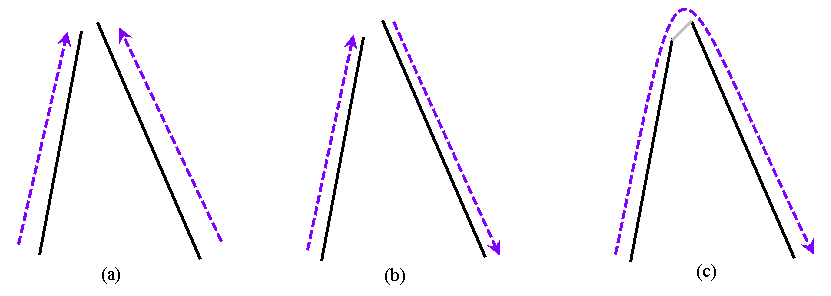


Figure 4: (a)Original (b) changing the direction , (c) interpolation

Finally , Adjusting the number of  points in each digit to a fixed number (70, empirically) for classification purposes.

**3.2** **Smoothing**

To eliminate the noise, we perform smoothing using 5-point moving average algorithm using MATLAB command "smooth".

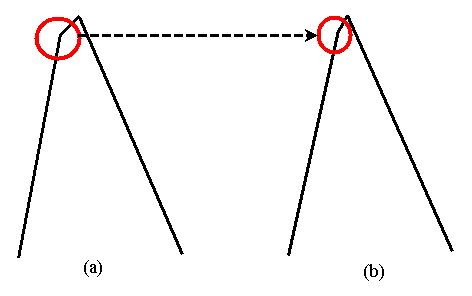


Figure 5: (a) Output from resampling , (b) after smoothing

As described , the pre-processing step is very important to permit the user to write the character with any number of strokes with any order .

**5** **Feature Extraction**

The feature extraction method used in a digit recognition system is the most important factor in achieving a good recognition rate. After studying various feature extraction methods, we selected a mixed feature vector containing various features. The following sections describe various features sets and algorithms we used to finally reach the final feature vector.

**5.1** **Temporal Features**

The temporal features are feature based on the online information of the users input stroke. They are computed for each point in the user pen path. Given the stroke is defined as , , where  is a point at instant  and = 70 number of points that represent the digit. The following features are computed:

**5.1.1**  **F1: Directional Feature:**

The local writing direction at a point at instant  is described by cosine and sine [?]:



where  ,  and  are defined as follows:



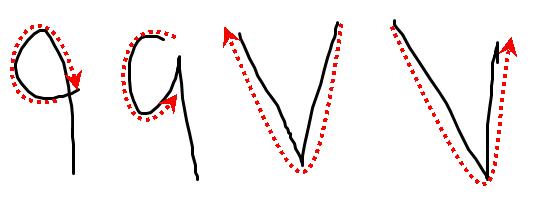


Figure 1: Various directions of Arabic digits 7 and 9.

The first set of features we used was the directional features only. Figure 1 shows the writing direction of Arabic digit '9' and '7'. It is clear that the writing direction is different and can be used to discriminate different digits. Unfortunately, the test found that some digits written in various directions can't be classified as shown in Fig 2. The figure shows that when a users change the writing direction more than once (or Over trace a digit), the system get confused and gives wrong results. This means that depending on directional features only is not enough and thus we tried to use other features which describe the global digit.

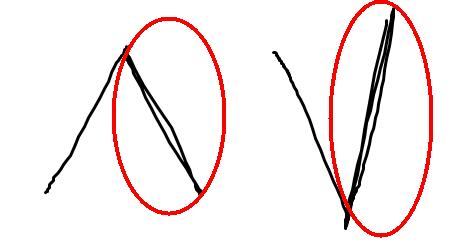


Figure 2: The images show the over tracing in writing a digit.

**5.2** **Spatial Features**

The following spatial features are used to solve the problem of depending on temporal features. But before extracting the spatial features we needed to convert the user’s strokes into bitmap image. The image is created for the on-line stroke by first connecting the stroke points by a straight line. At this stage some of the spatial features are extracted and other will require that the image is resized to a normalized size of . We can divide the spatial features into two sections. Global descriptive and local connectivity features where global features are the whole image features like area, aspect ratio. Unlike the local connectivity features which depends on neighborhood black pixels and connectivity.

**5.2.1** **F2:Global Features:**

• The aspect ratio characterizes the height-to-width ratio of the bounding box containing the preceding and succeeding points of .



where  and  are the height and width of image before resizing.

• The area of the bitmap image 

**5.2.2**  **F3: Number of white pixels**

The following connectivity features are computed on the bitmap image . The features describes the digit visually using connectivity of white and black pixels [?].

1. *'White 2'* White pixels between the black pixels at the top of the digit, the black pixels at the left of the digit, the black pixels at the bottom of the digit, and the right corner of the bounding box. Figure 3 shows the feature.

2. *'White 4'* White pixels surrounded by black pixels from above and below and having the left corner of the bounding box to their left. see fig 3.

3. *'White 5'* White pixels surrounded by black pixels in all directions. see fig 3.

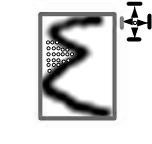
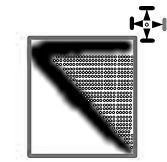
4. *'White 6'* White pixels surrounded by black pixels from the top and the right and by the left and bottom corners of the bounding box. see fig 3.

5. *'White 7'* White pixels surrounded by black pixels to their right and left and by the bottom corner of the bounding box from below.

6. *'White 8'* White pixels contained between the right corner of the bounding box until black pixels are encountered.

8. *'White 9'*The number of white pixels surrounded by black pixels in all directions in the upper left half of the bounding box

9. *'White 9 height'*the average height of the black pixels in the left half of the bounding box that is the distance between the first black pixel in one column and the last black pixel in the same column averaged over the columns of the left half of the bounding box.

fig6.TIFfig7.TIF

[ Feature 'White\_2'] [Feature 'White\_4'.] [Feature 'White\_5'.] [Feature 'White\_6'.]

fig8.TIFfig10.TIF

Figure 4: Examples of the Spatial Features

The spatial features improved the accuracy but another problem appeared while testing and it related to the Arabic digit 'zero'. The next section explains in details the problem and how we solved it.

**5.3** **Zero Problem**

The Arabic Digit zero '0' provided a different problem form other digits because it ambiguous and no specific way to write it, as it is only like a decimal point. Figure 5 shows different samples of the Arabic Digit '0'. The figure clearly shows that the digit does not have a specific way to write it normally written like other digits especially Arabic Digit '5' and '1'. Due to the ambiguity of the digit zero it is difficult to use the same set of feature to represent the zero as the other digits. The most discriminating feature of the zero than the other features is the size and number of points used to write it. Figure 6 shows the average size of Digit '0' versus the Arabic digit '1' and '2'.So since zero in Arabic is written in small area and random direction, so the digit zero should be recognized separately.

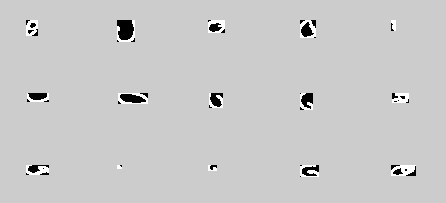


Figure 5: The images shows different samples of the Arabic Digit '0'

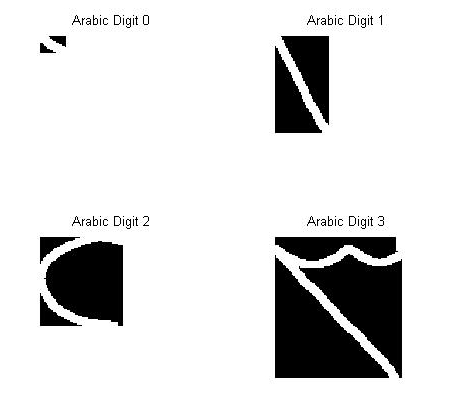


Figure 6: The images show the difference in size between the digits '0' and other digits.

To solve the digit Zero problem, we had to separate detecting the Arabic digit zero from the rest of the digits. Our final system goes through two stages, the first stage recognizes the digit zero and the second stage recognizes the remaining digits. Figure 7 shows a block diagram of the recognition system. The system is divided into three main blocks; Pre processing, Zero recognizer and Final Classifier. In Pre Processing the stroke are re-sampled and smoothed. The second stage is recognizing the zero digits using a classifier the output of this stage is either labeling the digit as a zero or continuing to the next stage. The final stage is an OVO classifier that discriminates between Arabic digits from 1 to 9.

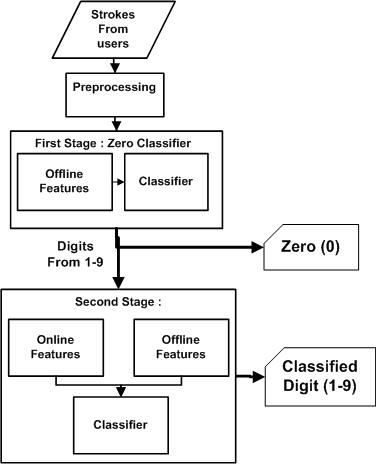


Figure 7: The chart shows the system block diagram.

**5.3.1** **The Zero Classifier**

As mentioned above it was clear that the Arabic Digit '0' will need offline feature to be discriminable from others digits. The main feature that differentiates between 0 and other digits is the size, number of points and aspect ratio. Trying to get the best result we tried various feature sets to finally choose the best feature set. The following list describes the three different features set that were used to test the Zero Classifier.

• Connectivity-FS: Some of the features mentioned in section 5.2.2. The feature vector consist of only ('White 2', 'White 3', 'White 4', 'White 5', 'White 6','White 8' 'White 9').

• Transitions-FS : The number of transition from black to white and vise vera. To reduce the dimensionality of the feature vector we choose to take only 3 in every direction of vertical, horizontal, left diagonal and right diagonal.

• Projection-FS : The projections of the bitmap image of the digit. To reduce the dimensionality of the feature vector we choose to take only 3 in every direction of vertical, horizontal , left diagonal and right diagonal.

As mentioned, the area, aspect ratio and number of points in the stroke are main features for the Arabic digit Zero. These three main features were added to each of the previous feature set and a test was conducted to choose the best feature set. The comparison for each of the feature sets are explained in the results section.

**6** **Result**

The system is composed of two stages (see Fig 11 ). The two stages are using a multi-class nonlinear SVM with input feature vector of length for first stage ... and a multi-class nonlinear SVM with input feature vector of length ... for second stage. The (164 )-element feature vector is a concatenation of the following feature: F1 (154 elements), F2 (1 element), F3 (9 elements).

Since the first stage focus only on the zero digits we used single linear SVM classifier. However, we needed a multi class classifier for the second stage. The nonlinear SVM is originally designed for 2-class problem. Extending it to multi-class can be done using the One Versus One (OVO) or the One Versus All (OVA) schemes. We used the OVO scheme to classify 9 digits which means we have 36 linear classifiers.

The collected AOD dataset is divided into 80% trainset and 20% testset. All the result shown are computed on the testset. The accurcay is defined as the nubmer of correctly classified digits divided by the total number of digits in the testset. Table 2 shows the accuracy of the system first stage. The table shows the result of each of the feature set discribed in section 4.3. The results shows that the Connectivity-FS and Transitions-FS features set have the best recognition rate.

Table 3 shows the accuracy of the whole system. The accuracies of each features for each stage of the system and the accuracy of whole system are presented. The table clearly shows that the adding offline feature to the second stage improves the accuracy

Table 2: Table shows the result of the zero classifier

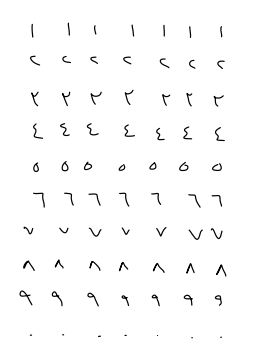
|  |  |  |
| --- | --- | --- |
| Features | Recognition % without (Zeros features) | Accuracy after Main features |
| Connectivity-FS | 91.7407 | 99.6054 |
| Transitions-FS | 90.6554 | 99.6899 |
| Projection-FS | 93.1219 | 99.4926 |

Table 3: Table compares between Results of different feature sets

|  |  |
| --- | --- |
| Features | Accuracy |
| Offline | 97.48% |
| Online | 98.62 |
| Online and Offline | 98.73% |

**6.1** **Comparison with commercial products**

Vision Objects: They have built cursive Arabic handwriting recognition system for ICDAR 2009 Online Arabic Handwriting Recognition Competition [?] based on MyScript handwriting recognition technology. They have designed the recognizer according to two different criteria. The first system (VisionObjects-1) provides the best accuracy whereas the second system (VisionObjects-2) is faster in exchange for a somewhat lower accuracy. We compared our system with MyScript Studio Notes Edition and we found that: The recognition rate for digits in MyScript system is lower than our system because the digits are confused with isolated Arabic characters as shown in Fig11. Figure 11 shows that MyScript confuse Arabic digit '1' with Arabic letter Alef and Arabic digit '4' with Arabic letter Ein.



res.JPG

Figure 8: My Script Result

The results for MyScript Studio Notes Edition are shown in Table 4 which shows the result of 10 writers of the dataset compared to result from our system. The result shows that MyScript achieve lower result than our system because our systems focus on the specific digit problem.

Table 4: Table compares between Results of different feature sets

|  |  |
| --- | --- |
| Program | Accuracy |
| our System | 0 |
| MyScript | 42 % |

**7** **Conclusion**

In this paper we presented a system that uses both on-line and offline information to recognize Arabic on-line digits. A new on-line Arabic digit dataset was collected to test the performance of the system presented. The dataset consist of 30,000 samples collected from 300 different writers. The system presented was divided into two main stages, the first stage to recognize the Arabic digit zero and the second stage to recognize the remaining digits. A comparison between different combination of on-line and offline features demonstrated that the best result was achieved using 98.6%. The paper also presented a comparison with another commercial system and the results showed that the system is more efficient as it focus on digit recognition problem.

**REFERENCES**

[1] V. M¨argner and H. E. Abed, “ICDAR 2009 Arabic Handwriting Recognition Competition,” 2009, pp. 1383–1387.

[2] S. Mozaffari, K. Faez, and V. Mnn”argner, “Application of Fractal Theory for On-Line and Off-Line Farsi Digit Recognition,” Machine Learning and Data Mining in Pattern Recognition, pp. 868–882, 2007.

[3] D. A. Karras, C. Li, Z. Majkic, and S. R. M. Prasanna, Eds., International Conference on Artificial Intelligence and Pattern Recognition, AIPR-07, Orlando, Florida, USA, July 9-12, 2007. ISRST, 2007.

[4] S. Mahmoud and S. Awaida, “RECOGNITION OF OFF-LINE HANDWRITTEN ARABIC (INDIAN) NUMERALS USING MULTI-SCALE FEATURES AND SUPPORT VECTOR MACHINES VS. HIDDEN MARKOV MODELS,” Arabian Journal for Science and Engineering, vol. 34, no. 2B, p. 430, 2009

[5] M. Kherallah, L. Haddad, A. M. Alimi, and A. Mitiche, “Online handwritten digit recognition based on trajectory and velocity modeling,” Pattern Recognition Letters, vol. 29, pp. 580–594, April 2008.

[6] C. Bahlmann, “Directional features in online handwriting recognition,” Pattern Recognition, vol. 39, p. 115 125, 2005.

[7] B. Chakraborty and G. Chakraborty, “A new feature extraction technique for on-line recognition of handwritten alphanumeric characters,” Inf. Sci. Appl., vol. 148, pp. 55–70, December 2002.

[8] A. Vinciarelli and M. Perrone, “Combining online and offline handwriting recognition,” in Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2, ser. ICDAR ’03. IEEE Computer Society, 2003, pp. 844–.

[9] S. Abdelazeem, “A novel domain-specific feature extraction scheme for arabic handwritten digits recognition,” Machine Learning and Applications, Fourth International Conference on, vol. 0, pp. 247–252, 2009.

[10] C. Liu, “Handwritten digit recognition: benchmarking of state-of-the-art techniques,” Pattern Recognition, vol. 36, no. 10, pp. 2271–2285, Oct. 2003.

[11] M. Pastor, A. Toselli, and E. Vidal, “Writing speed normalization for on-line handwritten text recognition,” in Proceedings of the Eighth International Conference on Document Analysis and Recognition, ser. ICDAR ’05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 1131–1135.

[12] S. J¨ager, S. Manke, J. Reichert, and A. Waibel, “Online handwriting recognition: the npen++ recognizer,” IJDAR, vol. 3, no. 3, pp. 169–180, 2001.