An On-Line Arabic Handwriting Recognition System

Based on a new On-line Graphemes Segmentation Technique

Hesham M. Eraqi, Sherif Abdel Azeem
Electronics Engineering Department
The American University in Cairo (AUC)
Cairo, Egypt
hesham.eraqi@gmail.com, shazeem@aucegypt.edu

Abstract— **In this paper we present a new system for on-line Arabic handwriting recognition based on a new on-line graphemes segmentation technique that depends on the local writing direction. Baseline detection, delayed strokes detection, PAW (Piece of Arabic Word) main stroke construction, and characters construction from the basic graphemes are issues that are addressed in this paper. Experiments are performed on the ADAB-database to validate the system and the segmentation method. The results show a significant improvement in terms of the contribution of segmentation errors to the overall system errors while providing high performance with a simple on-line feature extraction.**

Keywords- on-line handwriting; baseline detection; graphemes segmentation

I. INTRODUCTION

Arabic writing is cursive for both of its printed and handwritten forms, which require the segmentation of words into characters or part of characters, i.e. graphemes, either explicitly or implicitly like approach of using the Hidden Markov Mode HMM. The shape of the character varies according to its position in the word. Besides that, some characters share the same primary part and distinguished from each other by the secondary part which we call in this paper "delayed strokes". Moreover, some characters, especially in Arabic handwriting, may overlap with their neighboring characters forming what is called "ligature". In Arabic writing, each word consists of at least one PAW, while each PAW is composed of some connected characters (at least one character). More details about Arabic writing characteristics can be found in [1].

The above features, besides other characteristics of Arabic language, make Arabic recognition more difficult than other languages such as Latin or Chinese [4,8] and show the importance of the segmentation process role in the system, segmentation errors may produce character rejection and add more confusion in the recognition phase.

In on-line handwriting systems, the trace of the pen is used for the classification and recognition of the input information, which provides us with temporal features that are used to infer the dynamics of the writing in the different phases of the system.

In this paper, we introduce an on-Line Arabic handwriting recognition system based on a new explicit graphemes segmentation technique. In the coming sections, we are going to explore the different stages of the system, discussing the challenges of each stage and presenting our approaches of solving it. Finally, we will report our testing results based on the on-line ADAB-database [5].

II. PRE-PROCESSING

A. *Smoothing*

In order to remove the jaggedness of the contour resulting from the handwriting irregularity and the imperfection caused by the acquisition device, every point $P_t = [x(t), y(t)]$ in the trajectory is replaced according to the following equation:

$$P_{t_{new}} = \sum_{k=-n}^{n} \alpha_k P_{t+k} \text{ , } \sum_{k=-n}^{n} \alpha_k = 1$$

For each point to be the mean value of itself and its (2n) neighbors, $\alpha_k = (2n+1)^{-1}$. Smoothing is important for proper segmentation and has improved the final recognition rate.

B. *Resampling*

It's important to have an equal points' density along the writing contour for proper segmentation as the proposed segmentation algorithm is based on the local writing direction and independent of the speed of writing. Besides, the SVM classifier used in the system needs to have a constant number-of-points grapheme strokes for a constant-length feature vectors [6]. Generally, resampling may be used to reduce the size of the data samples and speed up the recognition time [8].

A writing speed normalization (resampling) algorithm based on trace segmentation method explained in [8], is used to redistribute data points (originally sampled in equal time intervals) to enforce even spacing (resampling distance) between them. In order to have a constant number of points for every stroke, a new resampling distance should be calculated for each stroke, where:

$$Resampling\ distance = \frac{Stroke\ contour\ length}{Target\ number\ of\ points}$$

For the proposed segmentation technique to segment properly, points of every stroke should be equidistant, regardless how many points are produced in each stroke (Figure 1). After segmentation, each grapheme is resampled to get a constant number of points to have a constant-length feature vectors.

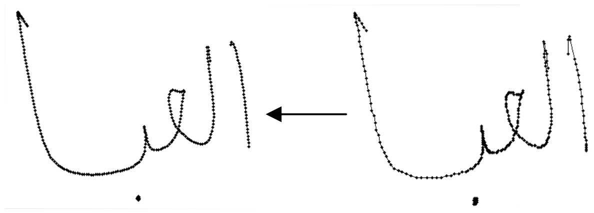

Figure 1. Smoothing and Resampling Effect for Segmentation

C. *Baseline Detection*

The following two stages in pre-processing, delayed strokes detection and PAW stroke construction, depend on the global baseline of the word under test. The diacritics represented in delayed strokes, word slope, and words that are constructed from more than one PAW are the main problems of detecting the Arabic handwriting baseline [2]. Baseline detection in our system is based on horizontal projection method, which is commonly used by the OCR researchers to detect Arabic baseline [9]. The following steps summarize the algorithm:

1- Construct an offline bounded image by interpolating every stroke of the word.

2- Remove some of the delayed strokes that are easy to detect by their small area, constant writing direction, and having writing above or below it.

3- Increase the thickness of every pixel vertically (around 5 pixels) as shown in Figure 2. This increases the chance that different PAWs baselines meet in the same vertical position.

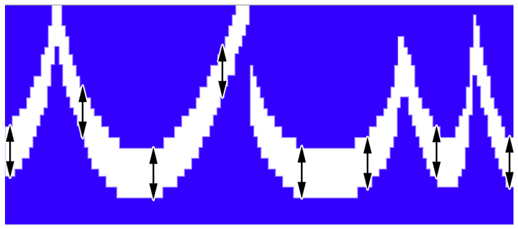

Figure 2. Increasing writing thickness vertically (zoomed)

4- Arbitrary baseline is selected according to the horizontal projection histogram maximum value as shown in Figure 3.

5- Search the histogram for a value higher than 80% of maximum projection value within the narrow area under the arbitrary baseline (20% under it),
<u>IF</u> it exists, this vertical position is selected to be the arbitrary baseline instead.
This step solves the problem caused by some Arabic letters that have an upper horizontal junction that may result into a histogram peak (Figure 3(A)), as like letters "ـصـ" and "ـحـ".

6- <u>IF</u> the arbitrary baseline is within the image upper part (upper 20% part of the image),
<u>THEN</u>, search the other part of the image for a projection value higher than 60% of the current baseline projection value. If it exists, this vertical position is selected to be the baseline.
This problem happens with the letter "ـكـ" (Figure 3(B)).
<u>ELSE IF</u> the arbitrary baseline is within the image lower part (lowest 40% of the image),
<u>THEN</u>, search the other part of the image for a projection higher than 60% of the current baseline projection value. If it exists, this vertical position is selected to be the baseline.
This problem happens with some of the Arabic letters like "ـر" and "ـن" (Figure 3(C)).

7- <u>IF</u> step 6 didn't select a new baseline,
<u>THEN</u>, the arbitrary baseline is selected to be the baseline.

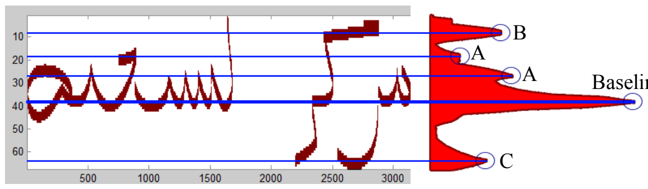

Figure 3. Baseline detection problems using horizontal projection histogram for the Arabic word "مركز الشيحية"

D. *Delayed Strokes Detection*

The processes of segmentation and graphemes classification are done on the main part of the word, so delayed strokes (secondary parts including diacritics) should be distinguished and excluded to be used later on the character construction phase. Figure 4 shows some examples of what we call delayed strokes, in red.

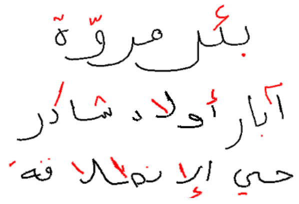

Figure 4. Delayed Strokes

Delayed strokes are detected and each delayed stroke is assigned to its main stroke using a holistic approach that is based on a set of Boolean expressions describing the stroke dimensions, shape, number of points, trace duration, and the vertical distance from baseline. Besides the associated main stroke relative area and position to the delayed stroke.

E. *PAW Main Stroke Construction*

In Arabic, there are a big number of classes (graphemes) that represent 28 characters, where the shape of each character is context sensitive and may have up to 4 shapes according to its position within the word. However, this big number of classes can be significantly reduced by:

(1) Isolation of the delayed strokes of the characters and recognizing them separately.

(2) Categorizing the classes according to the grapheme position within the PAW.

In our system, we have 4 sets of classes; beginning, middle, end, and isolated graphemes as shown in Table I. Grouping the PAW main strokes to form one stoke is necessary for determining the grapheme position within the PAW (beginning, middle, or end).

PAW stroke construction phase solves this problem by detecting and concatenating the main strokes which belong to the same PAW to form one PAW main stroke, while maintaining the spatial spacing between them that is important to differentiate some letters from each others like letter “ـحـ” and “ـعـ”. The time duration between strokes (time stamp), could be of great importance in this phase of the system. But it’s not available with the ADAB-database on-line information. Figure 5 shows two main strokes that should belong to the same PAW, where:

- CD: Concatenation threshold distance.
- MD: Minimum distance between the first and second strokes.
- P1, P2: The points corresponding to MD.
- P3, P4: The last and first point of the first and second stroke.
- Baseline upper range threshold is double the lower one.

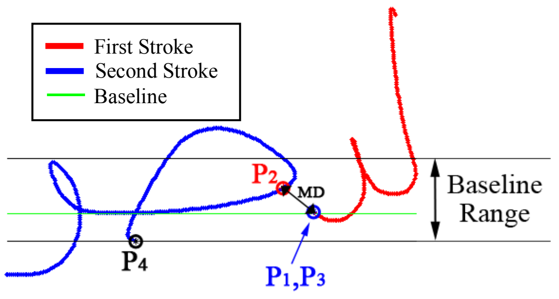

Figure 5. Two main strokes with MD<CD

The following algorithm iterates on the word main strokes (without delayed strokes), and concatenates the same PAW main strokes to form one PAW main stroke:

(A) Every two consecutive strokes where MD<CD are concatenated, <u>except if one the following 10 conditions is satisfied</u>:

1- Both P_1 <u>AND</u> P_2 are above the baseline range.

2- Both P_3 <u>AND</u> P_4 are below the baseline range.

3- (Either P_1 <u>OR</u> P_2 is below the baseline range) <u>AND</u> (MD $\geq$ 30% of CD).

4- (Either P_3 <u>OR</u> P_4 is below the baseline range) <u>AND</u> (MD $\geq$ 80% of CD).

5- P_1 is not within the first stroke end points (~10% of points) <u>AND</u> P_2 is not within the second stroke beginning points (~10% of points).

6- More than 70% of inter-points angles of the first stroke <u>OR</u> the second stroke are vertical and up directed (45º<angle<135º).

7- More than 70% of inter-points angles of the first stroke <u>OR</u> the second stroke are vertical and down directed (225º<angle<315º).

8- Both the end of the first stroke <u>AND</u> the beginning of the second strokes are not horizontal junctions.

9- First stroke vertical minimum point is far below the baseline range.

10- Second stroke is recognized to be isolated letter “ى”.

(B) Repeat step A until no concatenation is done for all the word main strokes.

III. Segmentation

A new segmentation algorithm has been developed that segments each PAW main stroke into its basic graphemes (one grapheme at least). Statistics made on handwriting strokes extracted from samples of the ADAB-database are used to define the thresholds of the algorithm. The algorithm is independent of the baseline, so that baseline errors won’t lead to segmentation errors. Table I shows the four sets of graphemes classes associated with our segmentation algorithm (without the delayed strokes).

TABLE I. Graphemes Classes

Beginning	Middle	End	Isolated
1. Laam	1. Laam-I	1. Alif	1. Alif
2. Laam_Alif	2. Kaf-I	2. Nuun_Tail	2. Laam_Alif
3. Kaaf-I	3. Nabra	3. Daal_Tail	3. Raa'
4. Nabra	4. Laam-II	4. Laam_Alif	4. Daad
5. Daal	5. Nuun/Raa'	5. Daal	5. Hamza
6. Nuun	6. Raa'	6. Raa'	6. Haa'
7. Raa'	7. H`aa'	7. Raa'_Tail	7. Taa'
8. H`aa'	8. Kaf-II	8. Nabra_Tail	8. Waaw
9. Eyn	9. Eyn	9. Waaw	9. Miim-I
10. Saad	10. Faa'	10. Miim-I	10. Miim-II
11. Faa'	11. Miim	11. Saad	11. H`aa'
12. Waaw	12. Saad	12. Haa'	12. Eyn
13. Miim	13. Haa'-I	13. Miim-II	
14. Haa'	14. Haa'-II	14. Miim_Tail	
15. Yaa'	15. Waaw	15. H`aa'	
16. Kaf-II	16. Yaa'	16. Eyn	
17. Laam_H`aa'			
18. Miim_H`aa'			
19. Laam_Miim			

-Red graphemes are considered only for PAWs of two graphemes
- Green graphemes represent the popular ligatures of Arabic according to the ADAB-database
-Blue graphemes are considered only for the before last grapheme of the PAW

The algorithm is summarized in the following steps:

3.1. **Arbitrary Segmentation.** Let every point that makes a horizontal angle with its next point (angle less than ~22º empirically) be called *a segmentation point*, these points are shown in Figures 6, 7 and 10 in red. This step is similar to the first step of a recognition-based segmentation algorithm discussed in [4].

3.2. **Points with a Hat Filtration.** For every segmentation point that have above writing (the up vertical line drawn from it, intersects the writing contour):

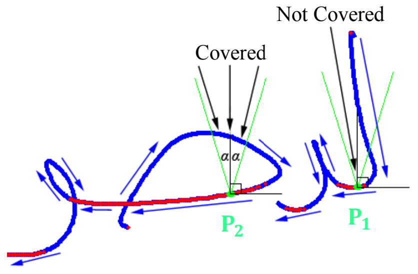

Figure 6. Umbrella Filtration Examples, where P_2 is a rejected segmentation point and P_1 is accepted

IF the whole range defined by the angle (90-α)º to (90+α)º is coverd from above (like P_2), THEN this point is rejected from the group of segmentation points (α is 20º empirically).

3.3. **Segmentation Junctions**. Let each group of consecutive segmentation points be called a segmentation junction (at least one point in each junction).

3.4. **Segmentation Junctions Smoothing.** For each two consecutive segmentation junctions:
IF the number of points between them is less than a pre-defined threshold (5 points empirically), THEN these points join the two junctions forming a bigger segmentation junction. Some examples of smoothed points are shown circled in Figure 7.

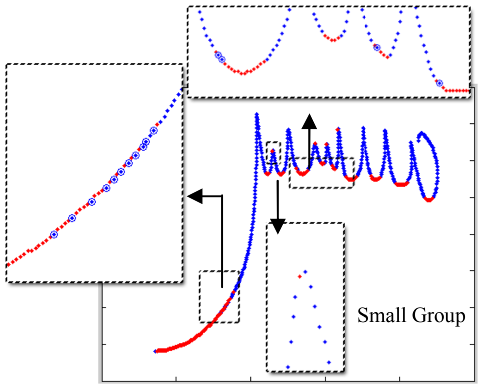

Figure 7. Segmentation Junctions and Smoothing

3.5. **Small Segmentation Junctions Filteration.**

(1) Let every point that makes a vertical to-down angle with its next point (angle within ~210º:330º) be called a down point, and that makes a vertical to-up angle (angle within ~30º:150º) be called up point, and other points be called netutral points. Figure 8 shows an exmaple of these points.

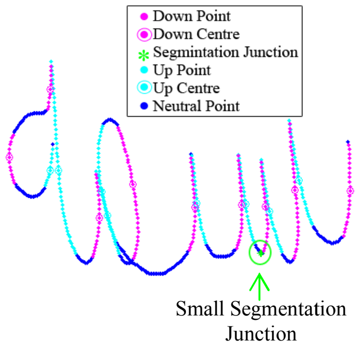

Figure 8. Up Points, Down Points and Neutral Points

(2) Smoothing the up and down points separatly like step 3.3.

(3) Each group of consecutive up/down points that is composed of less than Y points (3 points empirically) is rejected from the up/down points.

(4) Let the centre point of each consecutive group of up/down points be called up/down centre.

(5) For each small segmentation junction (composed of less than 5 points empirically):
IF it is preceded by up then down centres (or up within the stroke's beginning) and followed by up then down centres,
THEN segmentation junction is accepted.
ELSE segmentation junction is rejected.

3.6. **Hill Segmentation Junctions Filtration.** Figure 9 shows two examples of hill junctions circled in green.

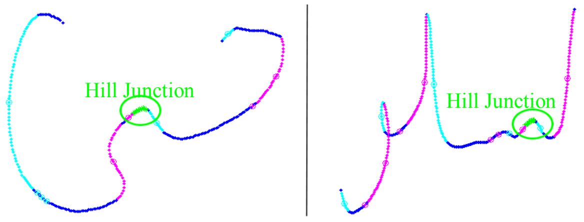

Figure 9. Hill Junctions

(1) Getting the up and down centres.

(2) For each segmentation junction:
IF it is preceded by up centre and followed by down centre.
THEN segmentation junction is rejected.

3.7. **Circular Segmentation Junctions Filteration.**
For each segmentation junction Si:

(1) Let Ci=0 and itereate on all the points of Si (Pij):
a- Get Gij; the group of consecutive points after Pij (the next 150 points empirically).
b- IF Pij have a below writing resulting from Gij, THEN increment Ci.

(2) IF Ci ≥ 70% of Si number of points,
THEN segmentation junction Si is rejected.
ELSE segmentation junction Si is accepted.

Figure 10 shows two examples of Pij, where Si in the right PAW is rejected, and in the left PAW is accepted.

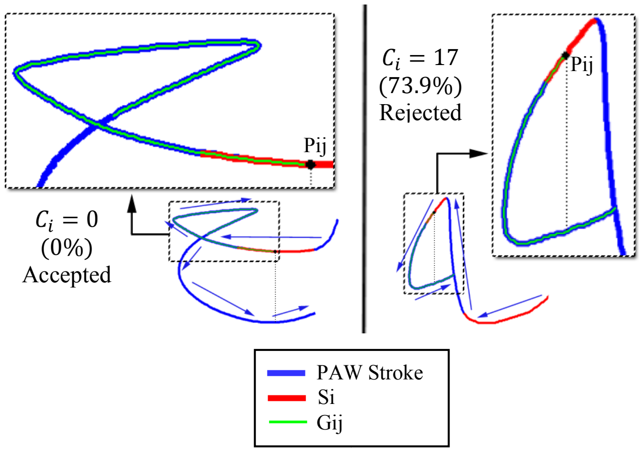

Figure 10. Circled Junctions

3.8. **Tail Segmentation Junction Filteration**.
IF the last point of the last segmentation junction is within the end of the stroke (last 6% of stroke points empirically),
THEN this segmentation junction is rejected (which is the case in Figure 6 last segmentation junction).

3.9. **Early Junction Filteration**. Each small segmentation junction that have no down centres before it and comes so early within the stokre is rejected. Figure 10 shows an example of early segmentation junction circled in green.

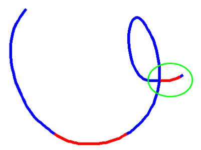

Figure 11. Early Junction Example

3.10. Final Segmentation Thresholds: The centre points of all the valid segmentation junctions are the final segmenatation points that separate the word graphemes.

IV. Characters Construction

In this phase of the system, characters are being constructed form the basic graphemes (Table I) and a set of associated delayed strokes. Experiments with the on-line system discussed in this paper have proved that delayed strokes handling and character construction rules are crucial phases that affect the system performance directly in terms of recognition rate and time.

Delayed strokes membership to a certain grapheme is determined according to the grapheme that makes maximum x-axis histogram overlap with the delayed stroke, after making a slight horizontal shift to right for all the delayed strokes, as diacritics in Arabic handwriting are often shifted to the left of its main grapheme [3]. However the delayed stroke associated with the Arabic letter "ک" is treated directly according to the nearest grapheme to it.

Characters are decided according to a set of rules that deal with the sequence of the recognized graphemes, associated delayed strokes' types, associated delayed strokes' vertical position (above or below the main stroke).

Table II shows some examples of the character construction rules for the Arabic letters "ف", "ق", and "ج":

TABLE II. Character Construction Rules Examples

Character	Graphemes Sequences and Associated Delayed Strokes		
ف	Class Beg. 11 & 1 Dot Up	Class Mid. 10 & 1 Dot Up	
ق	Class Beg. 11 & 1 Two-Dots Up	Class Beg. 11 & 2 Dot Up	Class Mid. 10 & 1 Two-Dots Up
ج	Class Beg. 8 & 1 Dot Down	Class Mid. 7 & 1 Dot Down	Class Iso. 11 & 1 Dot Middle

V. Results

We applied the system on the on-line database ADAB of Tunisian town names. A one-versus-one (OVO) multiclass fuzzy support vector machines (multiclass fuzzy SVM) model using RBF kernel was constructed based on the ADAB-database, using only one on-line feature; direction feature ($cos\, \alpha(t)$ and $sin\, \alpha(t)$) discussed in [6]. The number of word PAWs, delayed strokes formation of each PAW, and the number of characters of the word are used for lexicon reduction. While the lexicon is implemented using a dynamic programming technique called "*Minimum Edit Distance*" with equal penalty costs of insertion, deletion and substitution [7].

Table III shows the testing results in each set of the ADAB-database compared to another on-line recognition system [3] that is based on graphemes segmentation too. For our system, in each test, the data of the other two sets is used for training, while the training graphemes are obtained by a manual characters segmentation process followed by automatic graphemes segmentation using the segmentation technique discussed in this paper.

TABLE III. System Performance on ADAB Database

Test Set	**Recognition Rate** Our System		**Recognition Rate** [3]	
	Training Sets	***Top1***	***Training Sets***	***Top1***
Set1	2,3	87	1 to 3	87.1
Set2	1,3	86		84.79
Set3	1,2	87.6		-

VI. Conclusion

We presented in this paper a new complete system for on-line Arabic handwriting recognition. The main contributions of this work are the new on-line graphemes segmentation technique that depends on the local writing direction, and that our system achieves high recognition rate using a simple feature extraction and character construction rules. Recognition errors analysis shows that the contribution of segmentation errors in system errors is very little compared to other problems which are almost confined to two main problems; writing strokes disorder and delayed strokes drifting.

References

[1] B. Alsallakh and H. Safadi, "AraPen: an Arabic online handwriting recognition system," in *Proc of the 2nd Conf. on Information and Communication Technologies (ICTTA)*, vol. 1, pp. 1844-1849 (2006).

[2] A. Al-Shatnawi and K. Omar, "Methods of Arabic language baseline detection – the state of art," IJCSNS, vol. 8, no. 10, pp. 137-143 (October 2008).

[3] H. Boubaker, A. Elbaati, M. Kherallah, A.M. Alimi, and H. Elabed, "Online Arabic Handwriting Modeling System Based on the Graphemes Segmentation," in *Proc. ICPR*, pp.2061-2064 (2010).

[4] M. S. Khorsheed, "Off-line Arabic character recognition - a review," Pattern Analysis & Apps., vol. 5, pp. 31-45 (2002).

[5] K. Daifallah, N. Zarka, and H. Jamous, "Recognition-based segmentation algorithm for on-line Arabic handwriting," in *8th Inter. Conf. on Document Analysis and Recognition*, 2005.

[6] H. El-Abed, M. Kherallah, V. Märgner and A. M. Alimi, "On-line Arabic handwriting recognition competition - ADAB database and participating systems," International Journal on Document Analysis and Recognition (2010).

[7] I. Guyon, P. Albrecht, Y. Le Cun, J. Denker and W. Hubbard, "Design of a Neural Network Character Recognizer for a Touch Terminal," Pattern Recognition, vol. 24(2), pp. 105–119 (1991).

[8] M. Pastor, A. Toselli and E. Vidal, "Writing speed normalization for on-line handwritten text recognition," in *Proc. ICDAR,* 2005.

[9] M. Pechwitz and V. Maergner, "Baseline estimation for arabic handwritten words," in *8th Inter. Workshop on Frontiers in Handwriting Recognition (IWFHR'02)*, pp. 479–484 (2002).