Recognition-Based Segmentation Algorithm for On-Line Arabic Handwriting

Khaled Daifallah
Damascus University
Faculty of Information Technology Eng.
Khaled.daifallah@gmail.com

Dr. Nizar Zarka
Higher Institute for Applied Science and Technology
Damascus – Syria
Nizar.zarka@gmail.com

Hassan Jamous
Damascus University
Faculty of Information Technology Eng.
Hassan.jamous@gmail.com

Abstract

In this paper, we introduce an on-line Arabic handwritten recognition system based on new stroke segmentation algorithm. The proposed algorithm uses an over segmentation method that has the advantage of giving all correct segments at least. It is based on arbitrary segmentation followed by segmentation enhancement, consecutive joints connection and finally segmentation point locating. The proposed system gives an excellent recognition rate up to 97% and 92% for words and letter recognition.

1. Introduction

In the past years, many techniques have been applied by researchers in order to recognize printed or hand-written characters, and there are an immense number of researches which investigated in Latin and Chinese characters. But just a very few number of these researches considered the Arabic language, because of the difficulty in the nature of the Arabic script. In this paper, we propose an on-line Arabic handwritten recognition system based on new approach for segmentation. Testing results prove that the proposed segmentation gives high recognition rate in both letter and words. We first explain the basic characteristics of the Arabic script and an overview related work in on-line Arabic handwriting recognition. Then, we present the preprocessing phase, followed by our new segmentation algorithm, and feature extraction, and finally the hidden Markov models for the recognition phase.

2. Arabic Script Characteristics

The connection of Arabic letters and the variation of most of Arabic letters shapes according to its position in the word presented a great difficulty in processing Arabic script. Some of the Arabic script properties are:

- Arabic script consists of 28 letter + 10 marks.
- Some of the letter appears in 4 different forms according to its position in the word, these forms are: (Beginning form, BF; Middle form, MF; Isolated form, IF; and End form, EF) as it shows in table 1.

Table 1. Some Arabic characters in their different forms

BF	MF	EF	IF
أ	ـا	ـا	أ
بـ	ـبـ	ـب	ب
تـ	ـتـ	ـت	ت
عـ	ـعـ	ـع	ع
جـ	ـجـ	ـج	ج
د	ـد	ـد	د
سـ	ـسـ	ـس	س

- Characters in Arabic script are always connected (even when typed or printed).
- Some Arabic character has dots, these dots can be positioned above or below the letter body, and dots can be single, double or triple. Different Arabic letters may have the same letter body and only differ in these dots which identify them. Figure 1 shows three letters: "tha", "ta", and "ba" from left to right.

ث ت ب

Figure 1. Different character with the same letter body

These properties and many other characteristics of the Arabic script make it not easy to directly adapt English or any another script recognition system to Arabic script recognition.

3. Previous Work

Segmenting Arabic words into letters is the most difficult problem in Arabic script recognition. Amin [1] proposed segmentation based on the facts that connections between characters appear after an intersection or cusp points, or a change of curvature. But segmentation has to proceed through successive essays and the character recognition module is responsible for validating each essay. El-Sheikh and El-Taweel [2] proposed segmentation based on dividing letters into 4 groups of letters (initial, media, final and isolated). This approach would be extremely sensitive to noisy data in terms of number of strokes since the recognition system was built on counting the exact number of stroke. In Al-Emami and Usher [3] words were segmented into strokes by finding extreme curvature which was sensitive to rotation. In this paper we propose a new approach for stroke segmentation based over segmentation which gives all correct segments at least.

4. System Architecture

In this section, we describe the architecture of our online Arabic handwriting recognition system, and we develop our solution to letter segmentation and extraction problems. The system consists of the following phases: Preprocessing, letter segmentation, possible segmentations generation, extracting and normalizing letters, features extraction, recognition phase and choosing best results.

4.1. Preprocessing

Most of the recognition and classification techniques require the data to be in a predefined type and format which satisfy several requirements like size, quality, invariance…etc. These requirements are generally not met in the case of handwriting, because of many factors such as noise during digitalizing, handwriting irregularity, and handwriting styles variations. Preprocessing for on-line handwriting is performed to overcome these problems by performing smoothing, point clustering and dehooking. First smoothing is performed using low pass filter algorithm to reduce noise and remove imperfection caused by acquisition device. Point clustering is then performed in order to eliminate redundant points by averaging the neighboring points. Dehooking is the final preprocessing procedure to eliminate the part of the stroke which contains hooks which are commonly encountered at the strokes ends, caused by users while fast writing. Figure 3 shows respectively the results of the smoothing, point clustering and dehooking

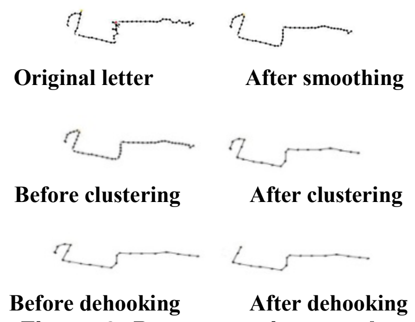

Figure 3. Preprocessing results

4.2. Letter segmentation

Our system works on strokes, this means that the system tries to recognize the stroke in order to recognize the whole word, because every Arabic word consists of one stroke at least, and every stroke is an Arabic letter (or a mark) at least. Figure 2 shows on the right a one stroke word of 3 letters, and on the left a 4 strokes word of 5 letters. So the system considers a sequence of strokes belong to the same word if the time between every two strokes in this sequence is smaller than specific threshold.

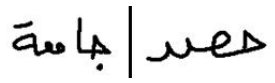

Figure 2. Left: 4 strokes word of 5 letters. Right: one stroke word of 3 letters.

To achieve letter segmentation (by segmenting strokes) we developed a new algorithm called KHDJ-1.

This algorithm works on strokes and segments them into letters by four stages: arbitrary segmentation, segmentation enhancement, connecting consecutive joints and finally locating segmentation points.

4.2.1. Arbitrary Segmentation. In this stage the algorithm searches for the joints between letters. For every line in the stroke $\overline{p_i p_{i+1}}$ this will be considered as a joint between letters if the angle between $\overline{p_i p_{i+1}}$ and the horizontal axis is smaller than θ ($\theta = \frac{\pi}{6}$ empirically), and the direction of line $\overline{p_i p_{i+1}}$ is from right to left.

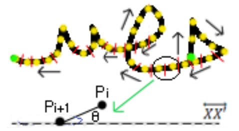

Figure 3. Suggested joints group

Figure 3 shows a stroke (a word consists of one stroke) with the group of suggested joints marked, in addition to an example of an accepted line $\overline{p_i p_{i+1}}$ as a joint. The complexity of this stage in the algorithm is Θ(n), where n is the number of points P_i in the stroke. This stage is the core of the segmentation algorithm because it must extract all joints. Because in definition the joint between two Arabic letters is a semi-horizontal line moving from right to left.

4.2.2. Segmentation enhancement. The previous stage performs over-segmentation; therefore this stage will eliminate some of the elements in the group of suggested joints. Accepted joints are the joints which don't have any above or below writings in the same stroke. The acceptance of a point p_i is determined as follow: IF the vertical line of p_i doesn't cross with the stroke THEN the point p_i is accepted (see blue points in figure 4).

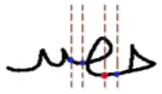
Figure 4. Accepted points.

ELSE (see red point in figure 4) check the pencil of lines which passes through the point p_i with the range α on the right and the left of p_i vertical line (see figure 5).

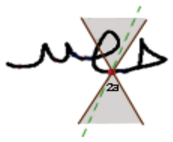

Figure 5. Pencil of lines with 2α angle

IF any of these lines doesn't intersect with the stroke (see dashed line in figure 5), THEN the point p_i is accepted ($\alpha = 25°$ empirically).
According to this IF the two end points of the joint are accepted THEN the joint is considered accepted.
Complexity of this stage is Θ(2k), where k is the number of joints.

4.2.3. Connecting consecutive joints. For every consecutive joints L_i, L_{i+1} if the end point of L_i equals the beginning point of L_{i+1}, we connect the two joints as it shows in figure 6. Complexity of this stage is Θ(k)

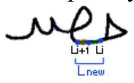

Figure 6. Connecting two consecutive joints

4.2.4. Locating segmentation points. By arriving to this stage our algorithm KHDJ-1 found M possible joints between letters. The segmentation point S_i for every joint $L_i : 1 \leq i \leq M$ will be its middle. (See figure 7).

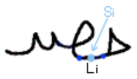

Figure 7. Segmentation point inside the joint

5. Possible segmentations generation

After segmenting the Arabic handwriting stroke into letters, some of these segments will remain wrong. But these wrong joints are extra (over-segmentation), and this is a good feature in the algorithm KHDJ-1. Therefore KHDJ-1 will encounter all right segments at least. Because of that the system needs a stage to generate all possible letters out of the suggested segments.

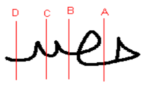

Figure 8. Suggested segmentation points

In the example of figure 8 we found segmenting points A,B,C,D, meanwhile the right segmentation is A and C. So in order to discover that, this stage will generate all the possible letters as follow: if there are N suggested segmentation point by the algorithm KHDJ-1, this means that the number of right segmentation points will be: N, N-1, …, 1, or 0. In figure 8 it could be seen that N=4 so the possible segmentation points will be either: 4 segmentation points: A,B,C,D, or 3 segmentation points: A,B,C-A,B,D-A,C,D-B,C,D, or 2 segmentation points: A,B-A,C-A,D-B,C-B,D-C,D, or 1 segmentation points: A-B-C-D, or 0 segmentation points: the stroke is one letter. Complexity of this generation equals to the total number of possible segmentations $Total = (\sum_{i=1}^{N} C_N^i) + 1$.

6. Extracting and normalizing letters

Extracting letters out of segmentation points is not straightforward, in figure 9; one of the possible segmentations of the stroke could be seen (generated by possible segmentations generation stage).

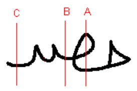

Figure 9. Possible segmentation

The First letter: between right edge and A, The second letter: between A and B, The third letter: between B and C, The fourth letter: between C and left edge. So if the first and the second letters are extracted directly, we will get the result shown in figure 10:

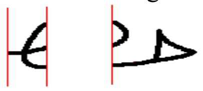
Figure 10. Extracting first and second letters

The first letter on the right (HAA) contains parts of the second letter (SAD), which makes recognizing it more difficult (but it still possible by considering this extra part as noise). But the big problem will appear in the second letter (SAD) which lost a part, this makes recognizing it extremely difficult and maybe impossible because it lost some of its features. So we propose in the following new method in order to extract letters out of segmentation points correctly.

6.1. Letters extracting method

Letters are located between segmenting points, so for every segmentation point S_i we search for the closest point in the stroke, and then the letter will be extracted by taking the points from the stroke between the determined points. Supposing S_i , S_{i+1} are segmentation points, so there is a letter enclosed between them (see figure 11).

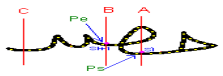

Figure 11. Taking points from stoke according to segmentation points

We search for the point P_s the closest to S_i, then we search for the point P_e the closest to S_{i+1}. The required letter is the group of points: $\{P_i\}_{i=s}^{e}$

6.2. Letters normalization

After letters extracting, every letter will be a group of points $\{P_i\}_{i=s}^{e}$ these points coordinates are according to the writing tablet. To transfer this coordinated to coordinates related to the letter; we calculate the smallest rectangle which contains the stroke (Figure 12), then we determine the center of the rectangle and the center of the rectangle which contains the letter. Then we perform the required translation to put the letter in the center of the stroke rectangle. Finally we perform scaling for every letter to put it in window frame with the size 128×128

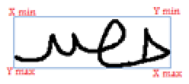

Figure 12. Smallest rectangle contains the stroke

7. Features extraction

A feature of an image is a distinguishing primitive characteristic or attribute of it. Probability theory has been applied to shape analysis by Hu [4]. He found that object shape is characterized by a few of the low-order moments. Hu has developed a set of seven compound spatial moments that are invariant in the continuous image domain to translation, rotation and scale change. So every letter will be featured by Hu's seven moments $h_1 , h_2 , h_3 , h_4 , h_5 , h_6 , h_7$. The equations to calculate Hu's invariant moments contain differences of relatively large quantities, and therefore, are sometimes subject to significant round off error.

8. Recognition

Every letter now is a vector of 7 dimensions (Hu's moments) and ready for being recognized. Hidden Markov Model HMM [5] is used as follow.

8.1. Using HMM

In order to use HMM the number of states should be specified [7], there are different opinions about how many states should be us, some said this number should be experimental, while others say that we should link the number of states with physical observations (like letters number N=28). Through our experiments we found that the number 28 for the states is large, and we didn't get the expected results, in addition to the large calculation time which the model took (solving the first HMM problem complexity is N^2T). Therefore we chose an experimental number of states (N=7) which gave as good results. Then symbols number M [7] must be specified and its representation $V_1, V_2, \ldots, V_M$. The same letter could have different representations because of writing styles variation between users, and variations of tablets devices. So Vector quantization is used, to achieve that we clustered the seven dimensions features vectors of the training set by k-means algorithm in 4096 clusters, then new features vectors will be approximated to clusters by Euclidean distance function. So we specified N=7 and M=4096, and other parameters A, B, π are initialized according to [7].

8.2. Training HMM

Our system recognizes letters without points or marks (above or below letters).

We divided the letters into 4 different groups:

1. Beginning form (BF) letters (without points) consists of 9 letters:

ٮـ ، حـ ، سـ ، صـ ، عـ ، ڡـ ، كـ ، لـ ، هـ

2. Middle form (MF) letters (without points) consists of 10 letters:

ـٮـ ، ـحـ ، ـسـ ، ـصـ ، ـعـ ، ـڡـ ، ـكـ ، ـلـ ، ـهـ ، ـمـ

3. End form (EF) letters (without points) consists of 14 letters:

ـٮ ، ـح ، ـد ، ـر ، ـس ، ـص ، ـط
ـع ، ـڡ ، ـل ، ـه ، ـم ، ـو ، ـى

4. Isolated form (IF) letters (without points) consists of 14 letters:

ٮ ، ح ، د ، ر ، س ، ص ، ط ، ع ، ڡ ، ل ، ه ، م ، و ، ى

We created a HMM for every letter in each group (47 HMM) then we trained each HMM on its letter from 8 to 10 times.

8.3. Recognizing letters by HMM

Dividing letters into the previous 4 groups will improve the recognition performance dramatically, because it reduces the problem complexity by feeding the features vector of each letter in the recognition phase to every trained HMM in the appropriate letters group only, as in the following:

- IF segments number = 0 (stroke is only one isolated letter THEN apply on the fourth group only (IF).
- IF segments number =1 (stroke is 2 letters: BF+EF) THEN apply the first letter on the first group (BF) and apply the second letter on the third group (EF).
- IF segments number >1 (stoke is more than 2 letters: BF+MF$^+$+IF) THEN apply the first letter on the first group (BF), apply the last letter on the third group (EF) and apply the remaining letters on the second group (MF$^+$). Dividing letters into groups is a new idea and it's not followed in off-line HMM based Arabic recognition systems [6], but from other aspects our HMM is so much like similar systems.

9. Choosing best result

let's suppose that our algorithm suggested for a stroke 8 segmentation points {A,B,C,D,E,F,G,H}, and let's take G_i one of the possible segmentations of the word: $G_i = \{A, B, F, H\}$, then every letter in G_i pass through the system till recognizing it by HMM with the probability P_k, then we computed efficiency of the possible segmentation G_i as follow: $V_i = \frac{1}{L}\sum_{k=1}^{L} P_k$
Where L is letters number in G_i. Then the system will consider the right segmentation of the stroke G_{Best} which satisfies: $V_{Best} = \max\{V_i\} : 1 \le i \le N$

10. Experimental results

While we were building and testing the system we encountered that the results is related with two factors: The accuracy of letters segmentation algorithm and the accuracy of the recognition model HMM. For letter segmentation accuracy we found that it's so much related with writing speed and writing style. For example we were able to write in a way which makes our algorithm gives many segmentation points, or in a way that makes it give only the right segmentation points. Finally, we came to the approximate relation that the maximum number of the suggested segmentation points exceeds the right segmentation points by 50%-60%. For the recognition model HMM accuracy, we couldn't test our system on a database of Arabic words due to two reasons: 1. our system handle on-line data, 2. our system handle letters without points or marks. So we tested the system on 150 non-pointed words written on-line by hand, and we got the results in table 2.

Table 2. Words recognition results

Testing type	Result
Same training user	92.6% - 139 words were recognized correctly.
New user knows how to use the tablet and the pen	85.3% - 128 words were recognized correctly.
New user doesn't know how to use the tablet and the pen	71.3% - 107 words were recognized correctly.

On the other hand, we calculated the recognition accuracy of letters in words; we had 720 letters inside the 150 words (4.8 letters per word). See table 3.

Table 3. Letters recognition results

Testing type	Result
Same training user	97.2% - 699 letters were recognized correctly.
New user knows how to use the tablet and the pen	88.8% - 639 letters were recognized correctly.
New user doesn't know how to use the tablet and the pen	79% - 569 letters were recognized correctly.

11. Conclusion and Future work

This paper introduced an on-line recognition system for Arabic hand-written words with new approach for letter segmentation. It is based on arbitrary segmentation, segmentation enhancement, connecting consecutive joints and finally locating segmentation points. This approach has the advantage of giving all correct segments at least. An evaluation of the system shows a high rate of word recognition for trained users. In the future we plan to handle the case of letters with points or marks (above or below letters).

12. References

[1] M. Schambach, J.Rottland, T. Alary, How to Convert a Latin Handwriting Recognition System to Arabic, *ICFHR*, 2008.

[2] El-Sheik, and El-Taweel, "Real-Time Arabic Handwritten Character Recognition", *Pattern Recognition, volume 23* (1990), No 12, pp. 1323-1332.

[3] S. Al-Emami and M. Usher. "On-line recognition of handwritten Arabic characters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 704-710, July 1990.

[4] M. K. Hu, "Visual Pattern Recognition by Moment Invariants," *IRE Trans. Information Theory*, IT-8, 2, February 1962, 179–187.

[5] Lawrence R. Rabiner, "A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE*, Vol. 77, No. 2, Feb. 1989.

[6] V. Margner, SARAT- A system for the recognition of Arabic printed text, *11th Int. Conf. on Pattern Recognition*, 1992, 561-564.