International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:5, 2007

Simultaneous Segmentation and Recognition of Arabic Characters in an Unconstrained On-Line Cursive Handwritten Document

Randa I. Elanwar, Mohsen A. Rashwan, and Samia A. Mashali

***Abstract*—**The last two decades witnessed some advances in the development of an Arabic character recognition (CR) system. Arabic CR faces technical problems not encountered in any other language that make Arabic CR systems achieve relatively low accuracy and retards establishing them as market products. We propose the basic stages towards a system that attacks the problem of recognizing on-line Arabic cursive handwriting. Rule-based methods are used to perform simultaneous segmentation and recognition of word portions in an unconstrained cursively handwritten document using dynamic programming. The output of these stages is in the form of a ranked list of the possible decisions. A new technique for text line separation is also used.

***Keywords*—**Arabic handwriting, character recognition, cursive handwriting, on-line recognition.

I. Introduction

MACHINE simulation of human reading has been the subject of intensive research for the last three decades. The interest devoted to this field is not explained only by the exciting challenges involved, but also the huge benefits that a system, designed in the context of a commercial application, could bring.

Handwriting is a skill that is personal to individuals. It consists of artificial graphical marks on a surface; its purpose is to communicate something [1]. It has continued to persist as a means of communication and recording information in day-to-day life due to the convenience of paper and pen as compared to keyboards for numerous day-to-day situations. The Character Recognition (CR) is the task of transforming language represented in its spatial form of graphical marks into its symbolic representation.

The recognition of handwritten characters is quite difficult due to the wide variability of hand printing and cursive script. Most of the efforts done in the field of character recognition were dedicated to recognize Latin, Japanese and Chinese characters. The Arabic language differs greatly from other Latin languages not only in its characters cursiveness but also in the language structure as well.

Unlike Latin characters, Arabic script is always written from right to left and no upper or lower case exists. Generally an Arabic word consists of one or more portions, and every portion has one or more characters. The discontinuities between points are due to some characters that are not connectable from the left side with the succeeding characters. Those characters appear only at the tail of connected portions, and the succeeding character forms the head of the next portion [2]. Many characters differ only by the presence and the number of dots above or below the main part of the character shape. Sometimes, the ambiguity of the positions of these dots in handwritten texts brings out many possible readings for one word. Moreover, every character has more than one shape, depending on its position within a connected portion of the word. According to this, Arabic CR systems still need more research to be established commercially [3].

For a CR system, when the input device is a digitizer tablet that transmits the signal in real time (as in pen-based computers and personal digital assistants) or includes timing information together with pen position (as in signature capture) we speak of on-line or dynamic recognition. Whereas, when the input device is a still camera or a scanner, which captures the position of digital ink on the page but not the order in which it was laid down, we speak of off-line or image-based OCR. The Arabic handwriting recognition problem, either off-line or on-line, is very much challenging.

Classically, on-line recognizers consist of a preprocessor, a classifier which provides estimates of probabilities for the different categories of characters (or other subword units) and a dynamic programming postprocessor, which eventually incorporates a language model [4]. The role of the postprocessor is to choose the best character category matching the context based on linguistics. We propose a rule-based algorithm for the two early stages of an on-line recognizer of cursive Arabic handwriting. Rule-based methods were used to perform simultaneous segmentation and recognition of word portions using dynamic programming. The output of these stages is in the form of a ranked list of the possible decisions. In the future, linguistics can be used to

Manuscript received March 26, 2007.

Randa I. Elanwar is Researcher Assistant in computers and systems Department, Electronic research institute, Cairo, Egypt (phone: 202-33310515; e-mail: eng_r_i_elanwar@yahoo.com).

Mohsen A. Rashwan is Professor of Digital Signal Processing, Electronics and communication Department, Cairo University, Cairo, Egypt (e-mail: Mohsen_Rashwan@rdi-eg.com).

Samia A. Mashali is Head of computers and systems Department, Electronic research institute, Cairo, Egypt (e-mail: samia@eri.sci.eg).

select the best decision from this list.

The organization of this paper goes as follows: in section 2 we give full description of the proposed system stages. Section 3 contains the results and conclusions and section 4 contains the future work.

II. System Description

Our on-line recognizer proposed stages are respectively: preprocessing, pattern definition, feature extraction, training and recognition stages.

A. Preprocessing

In this stage the handwritten document is broken up to text lines and to words or subwords. On-line English handwriting text line extraction techniques, depending on the y-axis histogram projection and character geometry (width, height, etc.) like those described in [5] and [6], do not function well when applied to Arabic handwriting due to the special characteristics of Arabic language described before. Thus, we thought of a new technique, more suitable to the Arabic language nature.

A stroke is defined as all data-point samples drawn/written between a certain pen-down (Start of writing action) action and the following pen-up (Lifting the pen up after writing) action. Thus, a stroke may represent one or more character, or even
a part of character, or sometimes a dot.

By examining the states of successively written Arabic strokes (either main-type strokes or complementary-type like dots for example) we found them related spatially to each other by one of the following relations:

1. **Touching**: then the two strokes should belong to the same word.
2. **Not touching but having an x-axis histogram overlap**: then the two strokes should belong to the same word.
3. **Neither touching nor overlapping on x-axis**: If the inter-stroke distance is less than the average stroke width, then the two strokes should belong to the same word. Otherwise, the two strokes should belong to two different words.

As shown in the example in Fig. 1:

- Strokes 1 and 2: neither touching nor overlapping but belong to the same word.
- Strokes 2 and 5: neither touching nor overlapping but belong to two different words.
- Strokes 1 and 3: overlapping and belong to the same word.
- Strokes 7 and 8: touching and belong to the same word.

The result of applying this word separation procedure is in the form of several independent groups of strokes. Each group of strokes contains the main and complementary strokes of the same word regardless the sequence/order by which they were written.

As explained before in section 1, there are many Arabic characters having the same main body and differ only by the

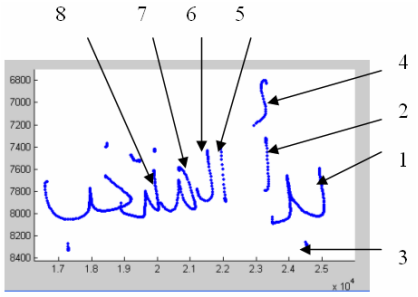

Fig. 1 The successive Arabic strokes states

number and position of the associated complementary strokes (dots). Thus; we tended to simplify the classification process and decrease confusion by removing these dots. This step caused the number of defined patterns/classes to reduce. The test input patterns are recognized after erasing these dots to find a list of all possible classification decisions. A later stage is added to restore the dots and delete the inconvenient decisions from the list according to the Arabic language characteristics.

B. Pattern Definition

We worked on Naskh font patterns with many different writing techniques for each character in all its possible positions in the Arabic word (Isolated, Start, Middle and End forms). Patterns were defined without their dots. Some examples are shown in Fig. 2.

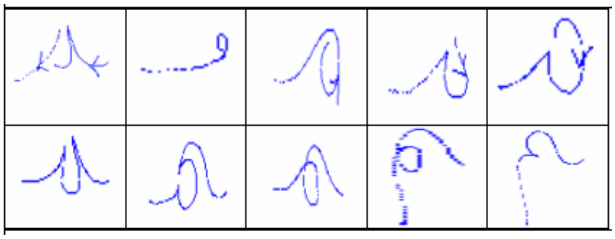

Fig. 2 Examples of the writing techniques of 'MEEM' in all positions

C. Feature Extraction

Almost all the researchers working on the on-line handwriting recognition problem take the pen trajectory directions as the main feature representing the on-line handwriting and some of them make use of unsampled pen movements (in air) as in [7], [8] and [9].

Depending on the directions, lengths, and pen-up/down movements of strokes, three freeman chain codes are defined as shown in the Fig. 3: eight long strokes (A–H), eight short strokes (a–h), eight pen-up (unsampled movement in air) movements (1-8) and one pen-up-down movement (0). In addition to this, an inter character movement representing the pen-up movement between the last pen-down stroke of the preceding character and the first pen-down stroke of the succeeding character is defined also. It is named pen-up movement ’9’.

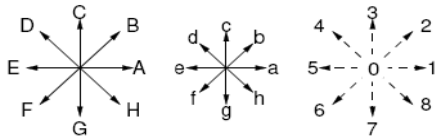

Fig. 3 Chain codes for feature extraction

D. Training

For each document in the training stage, after preprocessing, strokes are manually segmented according to the pre-defined pattern shapes and are structurally simplified as a sequence of directions of the straight-line segments. These line segments directions are encoded by the three freeman chain codes. The resulting sequence of direction code is called a *skeleton pattern*. It is regarded as a simplified representation of the pen movement.

For each defined pattern shape encountered in the training documents, all the representative skeleton patterns (frequently occurring) are grouped into a cluster and stored in a *registry*.

E. Recognition

After preprocessing the test document, words are tested sequentially. For each word, each main stroke (consisting of 1, 2 or more characters) is passed to the feature extraction stage to extract the direction codes for the whole stroke. The direction code under test is compared element-by-element versus the skeleton patterns in the registry built during the training stage to detect the pattern shape/shapes comprising this stroke and the segmentation locations as well.

This comparison between each two skeleton patterns is performed using a dynamic programming technique called "*Minimum Edit Distance*" [10]. The algorithm of this technique goes as follows:

```
Function Min-Edit-Distance (target, source) returns min-distance
n ← Length(target)
m ← Length(source)
Create a distance matrix distance [n+1, m+1]
distance [0,0] ← 0
for each column i from 0 to n do
   for each row j from 0 to m do
      distance [i,j] ← MIN (distance [i-1,j] + ins-cost(target_i),
                            distance [i-1,j-1] + subst-cost(source_j, target_i),
                            distance [i,j-1] + del-cost(source_j))
```

where the ins-cost is the cost of insertion error, del-cost is the cost of deletion error and subst-cost is the cost of substitution error. These costs are decided for the three direction groups: Group 1{A, B, C, D, E, F, G, H}, Group 2{a, b, c, d, e, f, g, h} and Group 3{1, 2, 3, 4, 5, 6, 7, 8, 9} as explained in Table 1.

The insertion and the deletion costs are the same and are equal to half the substitution cost since the substitution error can be considered as a deletion followed by an insertion.

TABLE I
THE SUBSTITUTION COST CALCULATION METHOD

Source String Element	Target String Element	Substitution Cost
Equal 'x' ∈ Group1	Equal 'x' ∈ Group1	Zero
Equal 'x' ∈ Group2	Equal 'x' ∈ Group2	Zero
Equal 'x' ∈ Group3	Equal 'x' ∈ Group3	Zero
Equal 'x' ∈ Group1	Equal 'y' ∈ Group1, 'y' ≠ 'x'	Absolute angle between 'x' and 'y'
Equal 'x' ∈ Group2	Equal 'y' ∈ Group2, 'y' ≠ 'x'	Absolute angle between 'x' and 'y'
Equal 'x' ∈ Group3	Equal 'y' ∈ Group3, 'y' ≠ 'x'	Absolute angle between 'x' and 'y'
Equal 'x' ∈ Group1	Equal 'y' ∈ Group2	4 * Absolute angle between 'x' and 'y'
Equal 'x' ∈ Group2	Equal 'y' ∈ Group1	4 * Absolute angle between 'x' and 'y'
Equal 'x' ∈ Group1 or Group2	Equal 'y' ∈ Group3	16 * Absolute angle between 'x' and 'y'
Equal 'x' ∈ Group3	Equal 'y' ∈ Group1 or Group2	16 * Absolute angle between 'x' and 'y'

The factors '4' and '16' come from the assumption that long strokes (represented by Group 1 directions) are almost twice the length of short strokes (represented by Group 2 directions) thus moving from Group 2 axes to Group 1 axes (and vice versa) may be multiplied by square this factor (2^2=4) to increase the substitution penalty reasonably while in the case of moving from Group 2 or Group 1 axes to Group 3 axes (and vice versa) is not logically accepted thus the substitution penalty should be increased significantly by multiplying a factor ((2^2)^2 = 16).

A cost function 'Distance' is defined to be equal to the minimum-edit-distance between the test skeleton pattern and the training skeleton pattern multiplied by a factor representing the amount of resemblance between them. This amount of resemblance is determined by string matching to find out the number of matches between the skeleton patterns from the registry and the test vector.

$$Distance = \text{minimum-edit-distance} \times \frac{\text{Length of skeleton pattern}}{\text{Number of matches}} \quad (1)$$

Thus, as the number of matches increases, the value of resemblance factor tends to 1 and remains the minimum-edit-distance measure only. While, as the number of matches decreases, the value of resemblance factor increases greatly and the '*Distance*' value will be multiples of the minimum-edit-distance.

The basic idea is to use a dynamic programming algorithm to find a globally optimal set of cuts through the input test string (feature vector) which minimizes the defined cost function. The set of cuts and their precise shape are found simultaneously.

The segmentation-recognition procedure goes as follows: The whole test skeleton pattern will first be compared to each training skeleton patterns in the registry element-by-element to choose the best first segmentation point having the minimum edit distance and maximum resemblance with this training pattern (start-end points of the first character).

This comparison is performed against all training pattern shapes. Thus, more than one possible answer can be found. The probable pattern shapes of the first character in the stroke are stored as roots of individual trees. Each tree is completed by comparing the unidentified part of the feature vector against the registry again and again to find the probable pattern shapes (and the next segmentation points) of the second, third, and fourth characters till the whole stroke is totally recognized.

We can obtain a ranked list. Each list member shows a choice of the possible patterns (without dots) representing the stroke. The rank is computed by accumulating the value of the cost function '*Distance*' associated with each recognized pattern.

The last step left in this stage is the dot restoration. By checking the validity of the number and location of the dots assigned to each character choice in each list member, we can remove the inconvenient choices. A new ranked list is created after renaming the valid patterns according to the associated dots (for example: Nabra + dot up will be renamed as Noon).

III. Results and Conclusion

The database we used for training is composed of 317 words (1814 characters), written by four writers the test database is composed of 94 words (435 characters) written by other four writers.

The correct choices of the test strokes occur within the Top50 list members 92% of the time (95% of the time for the test characters) as explained in Table II.

TABLE II
THE RESULTS SUMMARY

	Total Number	Correctly Recognized
Characters	435	415
Strokes	305	279
Words	94	70

The 5% loss in the number of recognized characters is the consequence of two problems:

1. Imperfect segmentation: Not having a large number of training samples (*representative skeleton patterns*) of each pattern shape defined may cause incorrect segmentation decision for test patterns having large degree of variety.
2. Wrong dot assignment: writer drifts and strokes overlap makes it hard to assign dots to the strokes at the dot restoration step. An example is shown below in Fig. 4.

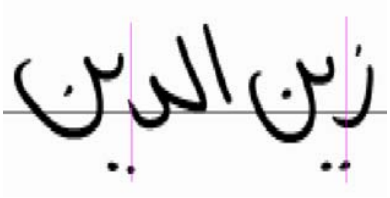

Fig. 4 Writer drifts while placing dots

IV. Future Work

Increasing training samples from multi writers and avoiding overlaps is expected to give much better results. Single output can be obtained by introducing a Language Model stage. Further more, the final character recognition accuracy can be enhanced and a large degree of writing variability can be encountered by using a multiple classifier system. Their outputs can be fused to have a final decision.

References

[1] Re'jean Plamondon and N. Srihari, "On-line and off-line handwriting recognition: a comprehensive survey," IEEE transactions on pattern analysis and machine intelligence, vol. 22, No. 1, January 2000.
[2] Yasser Hifny, "On-line Arabic handwriting recognition," M.S. thesis, Dept. Communication and Electronics Eng., Cairo Univ., Egypt, 2000.
[3] Hazem Y. Abdelazim, "Recent trends in Arabic OCR," in *Proc. 5th* Conference of Engineering Language, Ain Shams University, 2005.
[4] R. A. Cole, J. Mariani, H. Uszkoreit, A. Zaenen, and V. Zue, Survey of the State of the Art in Human Language Technology. Center for Spoken Language Understanding CSLU, Carnegie Mellon University, Pittsburgh, PA, 1995.
[5] E.H. Ratzlaff, "Inter-line Distance Estimation and Text Line Extraction For Unconstrained Online Handwriting," in *Proc. 7th* International Workshop on Frontiers in Handwriting Recognition, pp. 33–42, 2000.
[6] Gareth Loudon, Olle Pellijeff, and LI Zhong-Wei, "A Method for Handwriting Input and Correction on Smartphones," in *Proc. 7th* International Workshop on Frontiers in Handwriting Recognition, pp. 481–485, 2000.
[7] H. Shimodaira, T. Sudo, M. Nakai, and S. Sagayama, "On-line Overlaid-Handwriting Recognition Based on Substroke HMMs," in *Proc. ICDAR*, 2003.
[8] J. Lee, J. Kim, and J. H. Kim, "Data driven design of HMM topology for on-line handwriting recognition," in *Proc. 7th* International Workshop on Frontiers in Handwriting Recognition, pp. 239–248, 2000.
[9] Han Shu, "On-Line Handwriting Recognition Using Hidden Markov Models," M.S. thesis, Massachusetts Institute of Technology, 1997.
[10] Daniel Jurafski and James H. Martin, "Speech and Language Processing: An introduction to Natural Language processing, computational Linguistic and Speech recognition," Prentice-Hall, 2000, pp. 156.