International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:11, 2007

A Novel Approach to Persian Online Hand Writing Recognition

Ramin Halavati, Mansour Jamzad, and Mahdieh Soleymani

Abstract**—**Persian (Farsi) script is totally cursive and each character is written in several different forms depending on its former and later characters in the word. These complexities make automatic handwriting recognition of Persian a very hard problem and there are few contributions trying to work it out. This paper presents a novel practical approach to online recognition of Persian handwriting which is based on representation of inputs and patterns with very simple visual features and comparison of these simple terms. This recognition approach is tested over a set of Persian words and the results have been quite acceptable when the possible words where unknown and they were almost all correct in cases that the words where chosen from a prespecified list.

Keywords**—**Image Processing, Pattern Recognition.

I. Introduction

PERSIAN[1] handwriting is highly cursive, characters stick together and overlap, they can be written in several different forms and each character has several forms when it is written in different positions of a word. These complexities make Persian one of the most complex handwritings for automatic recognition systems, therefore there are few contributions trying to work out this problem.

To mention some, an approach has been introduced in [1] that uses a neural network to recognize online Arabic characters or [2], [3], and [4] which recognize offline segmented Arabic handwriting or [5] which uses a decision tree to recognize segmented Arabic text. The major problem of all mentioned methods is the high-dependability on small writing perturbations and very vast variety of writing styles which results in slow, yet imprecise results.

To overcome the complexities of Persian handwriting, a novel approach to online hand writing recognition is presented in this paper which has high tolerance versus perturbation of writing and can be easily expanded to adapt to different writing styles.

In this approach, the online input stream is first segmented using the visual features of the data and the segments are represented using very simple features. This is very much the same as what pupils who are learning Persian writing do. The characters are similarly defined with simple patterns which are used by school kids and to make the recognition, the input pattern and character patterns are compared.

The rest of paper is organized as follows: Section two presents a brief introduction to Persian handwriting and its problems. Section three presents the input segmentation and representation method. Section four is about the pattern definition approach and finally section five presents the pattern comparison method. Section six is on experimental results and the last section presents the conclusions and future works.

Authors are with Computer Engineering Department, Sharif University of Technology, Tehran, Iran ({halavati, soleyman}@ce.sharif.edu, jamzad@sharif.edu).

[1] This handwriting is used in most of Persian or Arabic speaking countries, making a population of more than half billion people.

II. About Persian Handwriting

Persian handwriting has a 32 letter alphabet and up to four different forms for each letter. 28 of these alphabet are exactly the same as Arabic but 4 belong only to Persian language. Words are separated from each other but the characters inside a word are sometimes concatenated and sometimes separated. Figure 1 right, presents samples of three different characters and their forms.

Also, the characters may be written in different forms based on personal writing styles, Figure 1 left, presents some samples of this kind. Despite the Far East characters which are composed of a set of several isolated linear parts that can be identified separately, most of Persian characters have just one curved stroke and most of them have one to three points. In addition, two or three points are usually written connected.

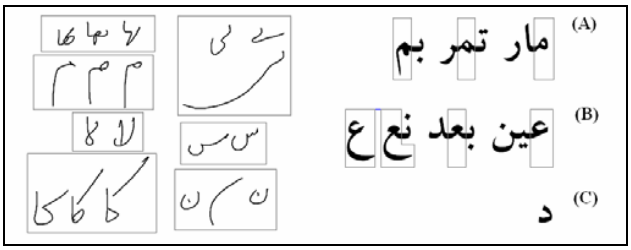

Fig. 1 RIGHT: Sample of Persian Printed Characters; (A) different forms of character "m"; (B) the four forms of character "eyn"; and (C) the only form of character "d". The character is selected in the two top samples with the gray rectangle. LEFT: Different hand writing variations by different people. All writings inside one gray rectangle are the same character or character combination

III. Segmentation And Representation

Our segmentation method is presented in this section. The target of this part is to separate the online input data from one

pen-down to the next pen-up[2] into a set of lines and curves and represent the input using simple features expressing these lines and curves and their properties. The properties and segment types are taken from a small case-study on primary school children.

A. Segmentation

To start the task, we assume the data as a set of vectors, each starting from one of the input points and ending in the next point. At this stage, we have a set of consecutive vectors which all have almost the same angles and can compose a line. Similarly, a set of consecutive vectors whose consecutive angle differences are almost the same can compose an arc. Thus, to make the segmentation we start from the first vector and go forth, computing the average of their angles and their change of angles. These two terms are shown with Avg_i and $AvgDelta_i$ in (1).

$$\begin{cases} Input: V_1 \ldots V_n,\ where V_i = (A_i, L_i) \\ Delta_i = A_{i+1} - A_i \\ Avg_i = \sum_{x=0}^{i} \frac{A_x}{i} \\ AvgDelta_i = \sum_{x=0}^{i} \frac{Delta_x}{i} \end{cases} \quad (1)$$

Any point in which the difference between Avg_i and A_i exceeds a certain threshold, can be a candidate for the end of a line part and when the difference between $AvgDelta_i$ and $Delta_i$ exceeds another threshold, an end of arc may be found. To have fewer number of segments, we compute and maintain both measures and when both of them exceed their thresholds, we end one segment, and start the process from where the next point is. Figure 2 right, shows some samples of segmentation with thresholds 20 and 7, respectively for Line and Curve parts.

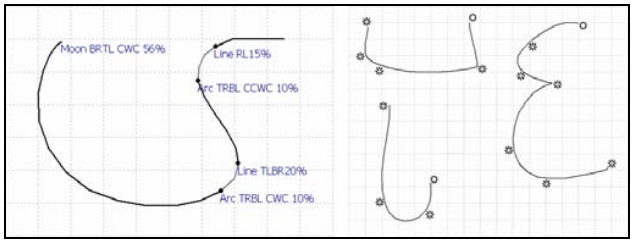

Fig. 2 RIGHT: Three samples of segmentation. In each trial, the start point is marked with circle and each segment end is shown with an asterisk. The top left character is called Be, The right one is Eyn and the bottom left is a combination of two concatenated characters pronounced Ba. LEFT: A sample of segmentation and representation. Segments are separated using dots and each segment's description is written at its end. RL stands for Right to Left, TRBL for Top Right to Bottom Left and so on for TRBL and BRTL

B. Representation

After the input data is segmented, each segment will be described using a set of features. Four different properties will be derived for each segments, its type, direction, curvature direction and relative length.

1) Type: There are three choices for segment's type: Line, Arc and Moon where Moon is a highly curved arc and a half-circle. To select among these three, we use with some minor modifications a straightness measure which is previously introduced in [6]. This measure is the ratio of the distance between the first and last point of the segment to the sum of distances between consecutive points of the segment. This measure is presented in (2).

$$\begin{cases} S = \frac{dist(P_n, P_0)}{\sum_{i=0\ldots n-1} dist(P_i, P_{i+1})} \\ Strightness = \frac{e^S - 1}{e - 1} \end{cases} \quad (2)$$

$$\begin{cases} 0.92 < S: & LINE \\ 0.6 < S \le 0.92: & ARC \\ S < 0.6: & MOON \end{cases} \quad (3)$$

If the straightness measure is above a certain threshold (we have practically used 0.92), the part will be called a line, otherwise, if the measure lies above another threshold (we have used 0.6), the part is called an arc and if none goes true, it will be categorized as a highly curved arc which for simplicity we call it Moon. These decisions are shown in (3).

2) Direction: We have eight possible choices for direction which are shown in Figure 3. The measure we have used for direction is the direct angle from the first point of the segment to its last point and the naming is done as presented in Figure □

3) Curvature Side: If the segment is curved, the next measure specifies that the curvature is on which side of the direct line from its first point to last point. To do so, the center of gravity of segment is found and the angle from first point to that point is computed. If this angle resides in left side of the direct angle from first to last point, it is assumed to be clockwise curvature and otherwise, it is a counter clockwise curvature. These two terms are referred to as C.W.C. and C.C.W.C. through out the paper.

4) Relative Length: The last and most trivial attribute is the relative length of the segment in comparison with the total length of the input data.

Figure 2 right, presents a sample of segment descriptions. As presented in this Figure, the input is separated into 5 segments and each segment is described using the above features.

IV. Pattern Definition

The previous section described how we translate the input stream into our simple representation. The next step in recognition is to compare that representation with a set of predefined patterns and choose the pattern which fits best. To define the patterns, we must note that some letters are written in several different forms and all of these forms must be defined for the recognizer. It is worth nothing that the complete set of patterns for Persian writing is theoretically uncountable and practically too many to store and compare, because any number of characters can be concatenated in a word, but we practically have very few combinations of more

[2] From the when the pen is put on the sensing tablet till when it is picked up.

than 4 or 5 concatenated letters. Thus, a suitable recognition system for Persian must be able to identify sub patterns and as our recognizer has this capability, we only need to store a subset of all possible patterns for the system which includes all single letters and some combinations between them which can form all combined forms. At this stage, we have defined a set of 30 letters and combinations we designed an active learning system that can learn these patterns based on user inputs.

To define the patterns, we have used a set of fuzzy membership functions to describe pattern lengths. The sets are in trapezoidal shapes as presented in Figure 4. They represent the following length values: 'ignorable', 'short', 'medium', 'long' and 'very long'.

A set of consecutive segments including their types, angles and sizes (stated in fuzzy values) compose a pattern. Table I presents three sample definitions which are used to define the three characters of Figure 2-Right.

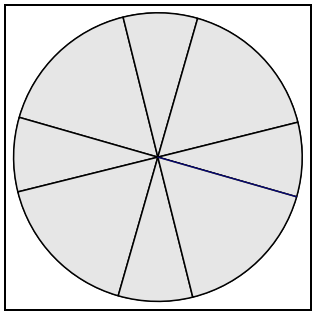

Fig. 3 Naming mechanism for segment directions. Once the direct angle from the first point of segment to its last point lay in any of the regions, the specified name will be tagged to that segment

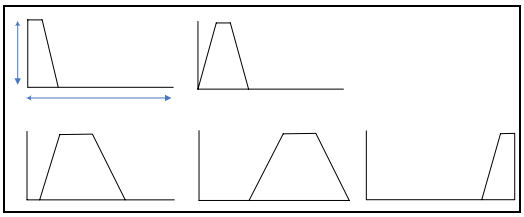

Fig. 4 Fuzzy Sets for Length Definition

TABLE I
SAMPLES OF PATTERN DEFINITION

	Type	Direction	Length
Letter B			
#1	Line	Top to Bottom	Short
#2	Line	Right to Left	Long
#3	Line	Bottom to Top	Short
Letter 'Eyn'			
#1	Moon	Top to Bottom	Medium
#2	Moon	Top to Bottom	Long
Combination 'Ba' (Letter Be + Letter A)			
#1	Line	Top to Bottom	Short
#2	Line	Right to Left	Short
#3	Line	Bottom to Top	Long

V. PATTERN RECOGNITION

In the previous parts we described how the input data is prepared and how the patterns are defined. This section presents the comparison algorithm which receives one character pattern and one input representation and returns a measure of compatibility between them. To do so, four important notes must be taken into account:

1) One single part of the input might be exploited in more than one part of the pattern: For example, as presented in lower left side of Figure 5, the input 'BA' might be segmented into two parts, a Top-Right to Bottom-Left Arc and a Bottom to Top Line. But we have defined three parts for this pattern (in Table I) and the first two parts of the pattern match the first part of the input. (The Top-Right to Bottom-Left Arc matches the consequence of Top to Bottom Line and Right to Left Line.)

2) Several parts of input may compose one part of the pattern: For example, as presented in upper left corner of Figure 5, (letter Be) we might have defined a long part in the pattern but have two smaller segments in input which both match the type and direction of the pattern and they are consecutive. In such cases, the two (or more) input segments might add up and compose one part of the pattern.

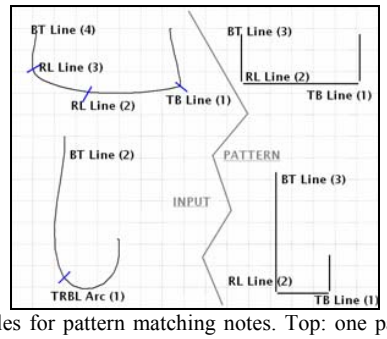

Fig. 5 Samples for pattern matching notes. Top: one part of pattern matches two segments of input. (letter Be) Bottom: One part of input matches two parts of the pattern (Ba)

3) The input might have extra parts: There might be extra parts in input which do not match any part of the pattern. If they are at the end of input, they may belong to another pattern and they show that this input is a concatenation of more than one pattern. But if they are between the meaningful parts, they can be the result of small perturbations in writing or specific writing styles. When the extra parts are less than a certain amount, they can be ignored but when they are more, they decrease the degree of compatibility between the pattern and input.

4) All parts of the pattern must be found in input: Although input might have extra parts, the pattern can not have such parts and all of it must be found in input.

We compare each pattern with the input in two phases: At first phase we compare each segment of input with each part of the pattern and draw out some single-part comparison results. In second phase, these results are used to find the best match between that pattern and the input.

A. Single Part Comparison

The first part of the matching process is to compare each

part of input with each part of the segment regardless of their positions in the sequence. When an input segment is compared with a pattern segment, three measures can be derived: How much do they match in shape and direction, how much extra part the input has before the matching part and how much extra part it has after the matching part. For example, as presented in Figure 4, we might wish to compare a *Top-Right to Bottom-Left Arc* from input with a *Top to Bottom Line* from the pattern. In this case, we can say that the input matches the pattern for 50% and it has 50% extra after the matching part. Also, if we compare the same input with the *Right to Left Line* part of the pattern, we have a 50% match which it has 50% extra before the matching part.

To make these comparisons, we use a small rule set which states the similarity of two parts and the extra parts before and after the similar section. Table II shows the rules we use to compare a line input with different pattern types and Figure 6 represents the visual form of these rules.

TABLE II
SINGLE-PART COMPARISON RULES FOR LINE PATTERNS

Rule #	Input Type	Angle Difference	% Extra Before	% Matching	% Extra After
1	Line	0	0	100	0
2	Line	+1 or -1 [3]	25	50	25
3	Arc	None	5	90	5
4	Arc-C.W.	+1		75	25
5	Arc-C.C.W.	+1	25	75	0
6	Arc-C.W.	-1	0	75	25
7	Arc-C.C.W.	-1	0	75	25
8	Moon	None	15	70	15
9	Moon-C.W.	+1	0	50	50
10	Moon-C.W.	-1	50	50	0
11	Moon-C.C.W.	-1	0	50	50
12	Moon-C.C.W.	+1	50	50	0

At this stage, we have a 2D Matrix, whose one side is the input parts and the other side is the pattern parts and each cell has three values: the compatibility, extra before and extra after. This matrix is called *comparison matrix* hence forth.

B. Whole-Pattern Matching

When the comparison matrix for a (pattern, input) pair is build, each of its rows stand for a pattern part and each of its columns stand for an input segment. A path through this matrix starting from the first row and ending in the last row can be regarded as a match between pattern parts and input parts. For example, Figure 7 presents a comparison matrix for an input with 5 segments and a pattern with 4 parts. The marked path says that the two first segments of input compose the first part of pattern; segment three of input is used for parts 2 and 3 of the pattern while pattern part 3 is made using segments 3, 4, and 5 of the input. And at last, part 4 of the pattern is matched using input segment 5, alone.

[3] They are in neighbor areas in the directions circle (Figure 4), for example, one is Top to Bottom and the other is Top-Left to Bottom-Right or Top-Right to Bottom-Left. +1 stands for clock wise different and -1 for counter clockwise.

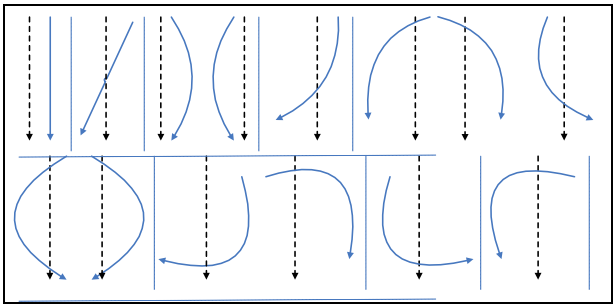

Fig. 6 Visual representation of matching rules of Table II. In each rule, dashed arrow shows the pattern and solid arrow shows the input

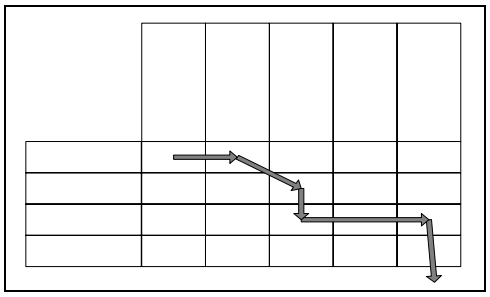

Fig. 7 The comparison matrix and a sample path inside it

The only rule of movement in the matrix is that you can not step back, neither vertically nor horizontally. While a path is drawn, a positive and a negative measure are computed for each part of the pattern. The positive measure (PM) represents how well that part of the pattern is fitted by the input; and negative measure (NM) is the sum of extra parts that lay before, after or inside that part of the pattern. For example, in Figure 7, the PM parts for pattern part 1 are the matching parts of input segments 1 and 2 and NM is the extra parts at the beginning and at the end of input segments 1 and 2. But for pattern part 2, the cost is only the extra parts at the beginning of input segment 3 because the tailing extra is used in pattern part 3.

To compute the PMs, one can simply add up the matching measures of the segments of path that lay in that part's row, but for NMs, the extra part is only added when ever it is not used by this part or another part of the pattern.

Once PM is computed, it is matched versus the fuzzy set representing that part's length and the result will be the *Matching Measure* for that Part. The final match of the pattern and the input will be the multiplication of Matching Measures minus the sum of Negative Measures.

Figure 8 presents a complete example on how the compatibility is computed. To find the compatibility between a pattern and an input, all possible paths are checked and the one with highest value is selected.

VI. EXPERIMENTAL RESULTS

As there is no general bench mark for online hand writing recognition in Persian and also there is no other hand writing recognizer for Persian which works with un-segmented data, we could not compare our result with another contribution and

only some practical results are presented.

It must be noted that we haven't tested the algorithm with the complete set of Persian letters, writing styles and letter combinations. We have put that to a later contribution with active learning to complete the pattern set. Some samples of patterns recognized cases are presented in Figure 9 and Table III presents the results for different cases.

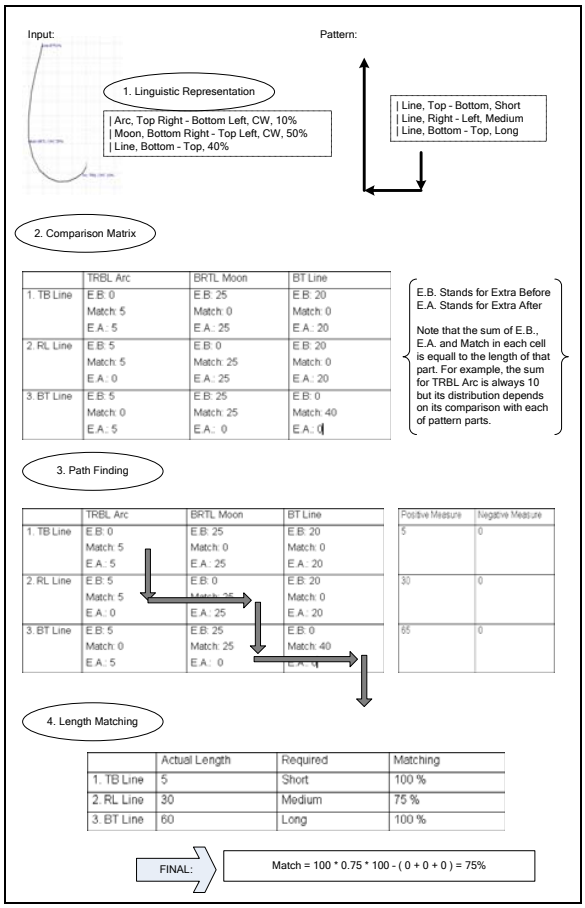

Fig. 8 A Complete sample for the recognition process

TABLE III
RECOGNITION RESULTS

Situation	Correct Answers
All letters without any guidance.	77 %
Only letters which can be defined with 3 or less pattern parts, without any guidance.	98 %
All letters, but the words where selected from a set of prespecified words.	95 %

VII. CONCLUSIONS AND FUTURE WORKS

There are very few approaches for recognition of online hand writing in Persian and among the existing ones, none works for un-segment and continues test. In this approach, a novel method which is based on representation of input tokens with very simple features, and comparison of these representations with patterns of the same form is presented. This approach is very much the same as what primary school kids do at the first weeks of reading.

As presented in results, our approach has been quite successful in accepting a wide range of variations for each letter and has shown about 95% correct recognition for the subset of Persian hand writing that is used in this contribution while the words where selected from a prespecifed list.

As a next necessary step to make this approach a practical method, we are now working on an active pattern learning algorithm which would be able to learn new writing styles and cope with individual writers.

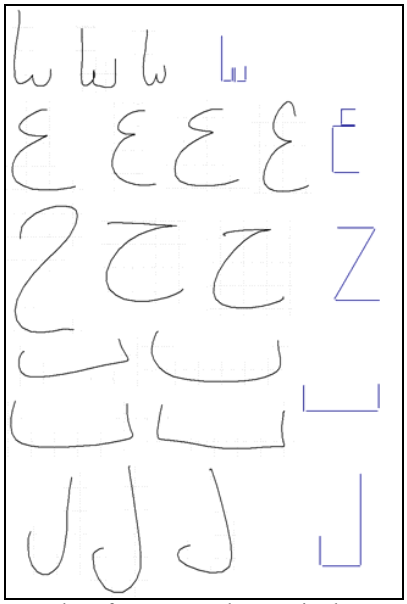

Fig. 9 Some samples of patterns and recognized test cases. In each line, the right most drawing is the pattern and the other ones are correctly recognised writings

REFERENCES

[1] A. M. Alimi, "*A Neuro-Fuzzy Approach to Recognize Arabic Handwritten Characters*", IEEE International Conference on Neural Network, vol. 3, pp. 1397 - 1400, 1997.

[2] I. S. I. Abuhaiba, M. J. J. Holt, S. Datta, "*Recognition of Off-Line Cursive Handwriting*", Computer Vision and Image Processing, Vol. 71, No. 1, pp. 19-38, 1998.

[3] I. S. I. Abuhaiba, S. A. Mahmoud, R. J. Green, "*Recognition of Handwritten Cursive Arabic Characters*", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 16, No. 6, 1994.

[4] A.Cheung, M. Bennamoun, N. W. Bergmann, "*An Arabic Optical Character Recognition System Using Recognition-Based Segmentation*", Pattern Recognition, vol. 34, pp. 215-233, 2001.

[5] A. Al-Emami, M. Usher, "*On-line Recognition of Handwritten Arabic Characters*", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 12, No. 7, 1990.

[6] Romesh Ranawana, Vasile Palade, G.E.M.D.C. Bandara, "*An Efficient Fuzzy Method for Handwritten Character Recognition*", Proceedings of KES-2004.