International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 21, No. 3 (2007) 491–513
© World Scientific Publishing Company

World Scientific
www.worldscientific.com

RECOGNITION OF PERSIAN ONLINE HANDWRITING USING ELASTIC FUZZY PATTERN RECOGNITION

RAMIN HALAVATI* and SAEED BAGHERI SHOURAKI†

Sharif University of Technology, Tehran, Iran
**halavati@ce.sharif.edu*
†*sbagheri@ce.sharif.edu*

Persian is a fully cursive handwriting in which each character may take different forms in different parts of the word, characters overlap and there is a wide range of possible styles. These complexities make automatic recognition of Persian a very hard task. This paper presents a novel approach on recognition of such writings systems which is based on the description of input stream by a sequence of fuzzy linguistic terms; representation of character patterns with the same descriptive language; and comparison of inputs with character patterns using a novel elastic pattern matching approach. As there is no general benchmark for recognition of Persian handwriting, the approach has been tested on the set of words in first primary Iranian school books including 1250 words resulting in 78% correct recognition without dictionary and 96% with dictionary.

Keywords: Online handwriting recognition; elastic pattern matching; fuzzy modeling.

1. Introduction

Persian character set and writing style along with other members of Arabic family character set are used by more than 30% of the world's population and serve in the writing of many widespread languages such as Farsi, Arabic, and Urdu.[2] In contrast to advances in online Latin and Far East handwriting recognition, relatively few studies have been devoted to these languages. This is mainly due to its highly cursive nature which makes it a very complex pattern for both segmentation and recognition: characters are written in up to four different forms in different parts of word, they usually stick together, they may overlap, and there are several different writing styles (Figs. 1–3).

A number of online handwriting recognition systems have been proposed in the last two decades incorporating stochastic, neural network, model matching or structural/syntactical techniques such as Refs. 1, 4–8, 10, 11, 14, 15, 19 and 20. A detailed review of these approaches is presented in Sec. 6 during comparison of the presented approach with them. The major problem of all existing methods is high sensitivity to writing perturbations due to the vast variety of writing styles which results in imprecise and slow recognition or writer dependency.

Letter Name:	Alef	Pe	Jhe	Shin	Ein	He
Isolated Form:	ا	پ	ژ	ش	ع	ه
Beginning of the Word:	اسب	پا	ژرف	شب	عقرب	هوا
Middle of the Word:	باران	تپه	مژه	نشان	بعد	مهیار
End of the Word:	دارا	لپ	تاژ	میش	برقع	رفته

Fig. 1. Different forms of a character based on its position.

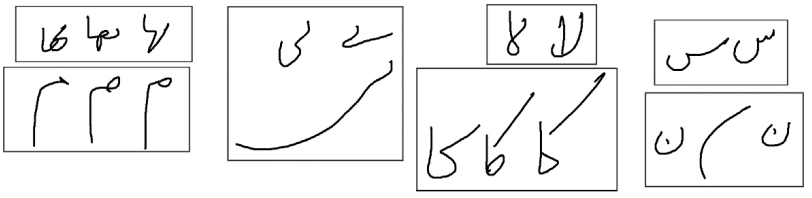

Fig. 2. Different writing variation samples by different people. All writings within a rectangle are the same characters, written by different people.

Fig. 3. A sample of Persian calligraphy. Most of the writing rules can be neglected for the sake of beauty and harmony. This paper deals only with normal writing styles.

Considering the limitations and drawbacks of the existing systems, it can be concluded that a practical handwriting recognition method for complex writing systems such as Persian should be both fast and flexible. To achieve these requirements, the knowledge base must be small but robust and changes in style or orientation should be covered by flexible prototypes that contain widely valid description of character information.[16]

To have the above features together, this paper presents a fuzzy elastic approach with the following three major characteristics: first, it represents input data and patterns using a fuzzy linguistic description that is derived from a case study on school kids that learn how to read and write Persian. This description method results in ignorance of some perturbations and more robustness when dealing with different writing styles. Second, the proposed recognition approach performs the recognition and segmentation of input stream simultaneously and therefore overcomes the common problem of recognition/segmentation priority. And third, the recognition part uses a novel elastic approach that is very much robust to writing innovations and perturbations.

The rest of paper is organized as follows: In the next section, we discuss the main rationales behind our method and the general ideas. Section 3 represents the segmentation and description processes. In Sec. 4, the main pattern comparison algorithm is introduced and Sec. 5 presents our experimental results. Section 6 will be on results and feature's comparison between our approach and some other contributions and finally the conclusions and future works.

2. The Rationales Behind Our Method

Based on a case study on school kids at the very first stages of learning how to read and write Persian in Iranian schools, it was noted that they try to describe characters using simpler tokens called "lohe" which mainly consist of straight lines, arcs and circles. Separation of a written word into lohes by different children is not exactly the same and has a lot of variance but description of letters and words using these lohes is very much the same.[12]

These findings suggest that when a kid has not yet learned to recognize the word as a whole, s/he first tries to separate it into a set of linguistically describable terms and compare it with some other linguistic patterns which define letters and words. And as another important fact, this separation is not unique, but it is somehow compared with word/letter descriptions that the differences are ignored and the main pattern is recognized. Therefore, a description of input data and desired patterns by a set of fuzzy linguistic terms and a robust comparison between these terms can be an appropriate method to solve the hard problem of cursive handwriting recognition for computers, which are still in the primary school days of learning to solve human level problems.

To implement the above idea, we first try to separate the sequence of input points received from a pointing device into a sequence of tokens, each in the form of a line or an arc. To make the separation, we implement the fact that when the eye follows a sequence of points, it assumes a line, when the comprising vectors have almost similar angles, and it assumes an are when succeeding vectors change their angles almost in a constant manner (Fig. 4). This idea is fully represented in Sec. 3.1.

Once the input is segmented, we can describe each segment with fuzzy linguistic terms by simple rules of thumb such as if its angle is around zero, it is a top to bottom part, etc. The characters are also described with similar linguistic terms and comprise a fuzzy knowledge base of patterns. When this is done, the input is described using expressions such as "*a short arc from top right to bottom left followed by a long line from bottom to top*" and it must match with a character pattern such as "*a small line from top to left, then a small line from left to right and finally a long line from bottom to left*". The description method is expressed in detail in Sec. 3.2.

Due to unavoidable uncertainty in segmentation phase regarding writers' style, writing speed, writing device's friction, etc. the comparison between the input and

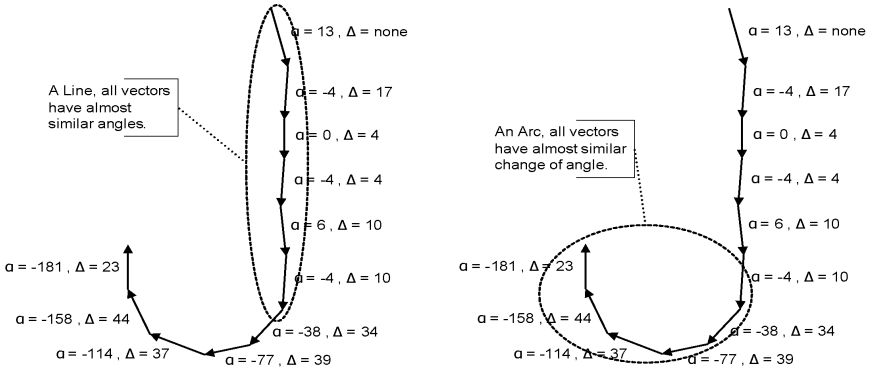

Fig. 4. Two samples for line and arc detection ideas. In both images, the angle of each vector (α) and its difference with previous vector's angle (Δ) are presented. In the left image, the selected vectors have almost similar angles (all around zero) and they compose a line and in the right image, selected vectors have similar change of angles (all around 35) and comprise an arc.

fuzzy patterns must deal with different unequal number of segments in input and pattern, unwanted segments, and differences between writer's style and defined patterns.

To cope with these necessities, we introduce a novel elastic fuzzy pattern comparison method which works as follows: at the reception of first segment of input, it generates some hypotheses on what the input is. By receiving further parts, it tries to produce new hypotheses based on the previous ones and prune those that are falsified. Once the input is complete, the hypothesis with the highest degree of belief is chosen as the matched pattern. This approach is described in detail in Sec. 4.

Different contributions exist for this task such as model matching approaches,[13,22,24] dynamic programming methods,[9,17,23] and dynamic time warping[3] mostly with the basic idea of a point to point mapping of inputs and patterns and a way of finding the best mapping. The major advantages of this approach compared to cited methods are, in general:

(i) Description of input and pattern with fuzzy linguistic terms reduces the search space by a big factor.
(ii) As the compared terms are all described with fuzzy terms, robustness versus perturbations is much more than point to point comparisons.
(iii) The pruning methods prevent searching many unwanted states.

3. Segmentation and Representation Method

Based on the ideas presented in Sec. 2, this section presents the details on input segmentation, representation with fuzzy linguistic terms and character patterns definition.

3.1. *Input primary segmentation*

The input data is a sequence of vectors, separated from each other when the writer has picked up the writing device. We will call the sequence of points/vectors placing the pen on a writing device with its next pick up, an episode. Using this view, segmentation will be to separate an episode into one or more groups of consecutive vectors for which each composes one line or an arc. To do so and based on the ideas stated in Sec. 2, we start from the first point of an episode and compute the average of vector angles and average of vector angle changes (Formula 1, AvgAngle_i and AvgDelta_i).

To have a measure of how much a point is a suitable candidate for being the end of a line or an arc, LineMeasure_i and ArcMeasure_i are computed as the absolute difference between vector's angle and angle change with the respective averages of previous points. Once both of these measures exceed certain thresholds, the point is marked as the end of the segment and the sequence restarts from the next point. We have practically chosen 7 and 20 as thresholds for ArcMeasure and LineMeasure, but as stated in Table 5 of the experimental results section, the system is not sensitive to these exact values. Once the segmentation is completed, we refine it by merging very small segments with their neighbor segments. Figure 5 presents the complete algorithm.

$$\begin{cases} \text{Delta}_i = \text{Angle}_i - \text{Angle}_{i-1} & \text{LineMeasure}_i = |\text{Angle}_i - \text{AvgAngle}_{i-1}| \\ \text{AvgAngle}_i = \sum_{x=0}^{i} \text{Angle}_x / i & \text{ArcMeasure}_i = |\text{Delta}_i - \text{AvgDelta}_{i-1}| \\ \text{AvgDelta}_i = \sum_{x=0}^{i} \text{Delta}_x / i \end{cases} \quad (1)$$

3.2. *Input description*

Once the input sequence is segmented, it is described using four fuzzy linguistic terms, namely segment's type, direction, length and direction of curvature:

3.2.1. *Segment type*

Three descriptive terms are used for segment's type: line, arc and semicircle. To assign one of these terms to the segment, a straightness measure is used which was previously introduced in Ref. 24 with minor modifications. This measure is the ratio of the distance between the first and last points of the segment to the sum of the distances between consecutive points of the segment. This measure is presented in Eq. (2), and fuzzy membership sets that choose the appropriate linguistic term are presented in Fig. 6.

$$\text{Straightness} = 100 \times \exp\left(\frac{\text{dist}(P_n, P_0)}{\sum \text{dist}(P_i, P_{i+1})}\right) \Big/ (e - 1) \quad (2)$$

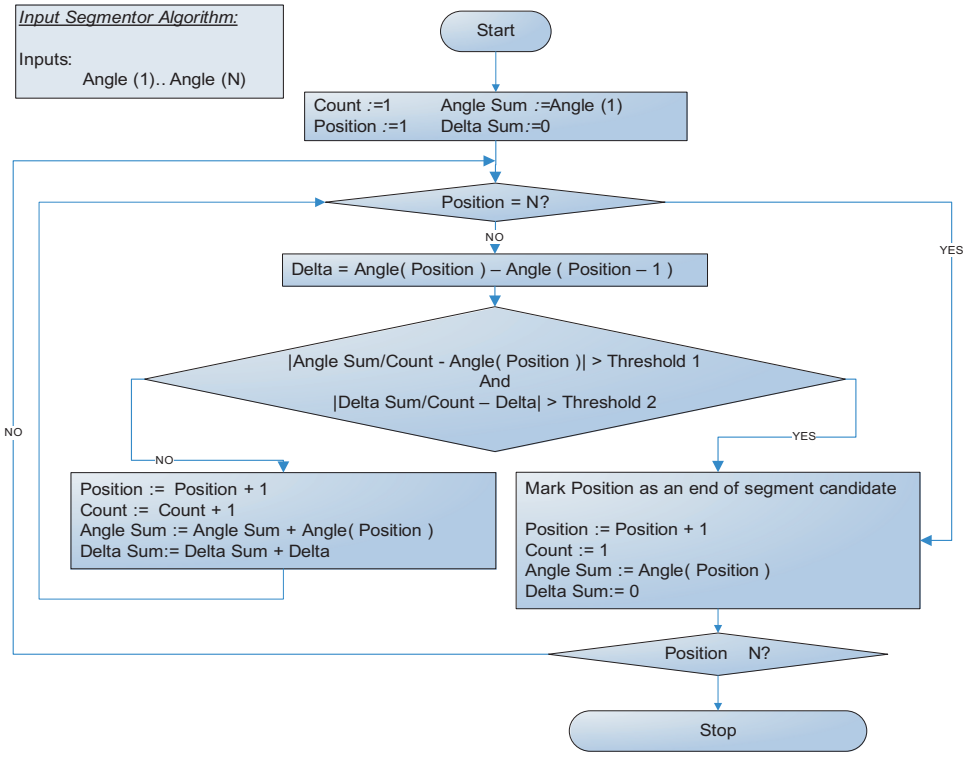

Fig. 5. The segmentation algorithm.

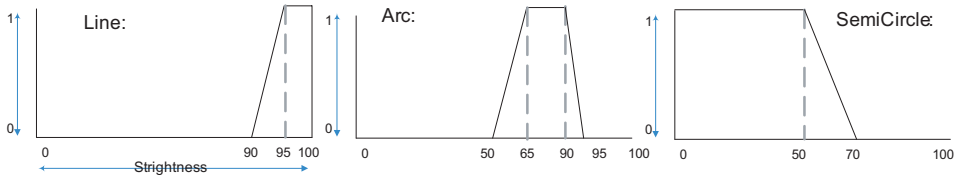

Fig. 6. Membership functions for segment type linguistic variables.

3.2.2. *Segment direction*

The second feature is the segment's direction which is described using eight fuzzy linguistic terms as shown in Fig. 7. The measure we have used for direction is the direct angle from the first point of the segment to its last point.

3.2.3. *Curvature direction*

If the segment is an arc or semi-circle, the next measure specifies the side of curvature on the direct line from the first point of the segment to its last point. To do so, the center of gravity of the segment is computed and the vector from the first

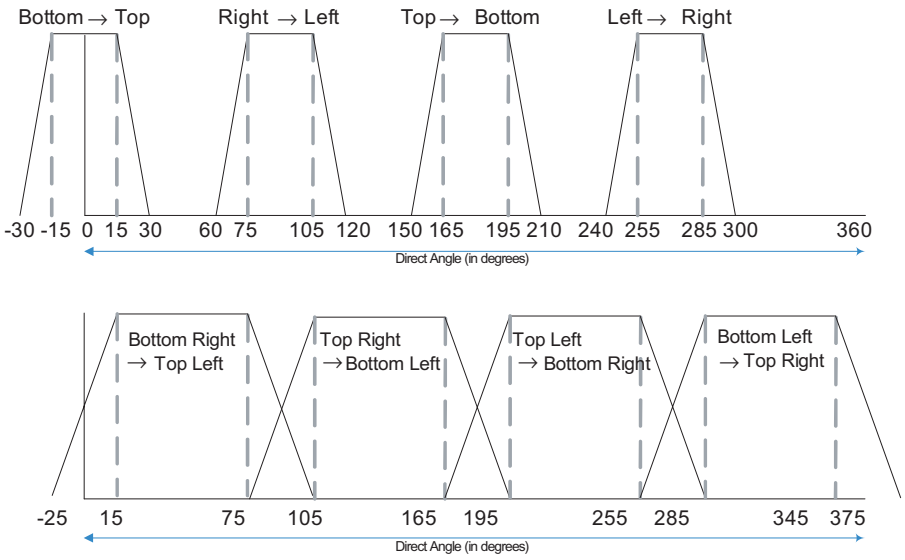

Fig. 7. Direction linguistic terms. Horizontal axis represents the direct angle from the first point of segment to its last point in degrees and vertical axis is the degree of belief in specified membership function.

point to COG is found. If this vector resides on the left of the direct vector from the first to last points, it is called clockwise curvature and otherwise, it is a counter clockwise curvature. This feature is the only non-fuzzy feature and has two crisp values: clockwise (C), counter clockwise (C.C). Figure 8 visualizes this measure.

3.2.4. *Segment length*

Segment length is the last and most trivial feature and is represented in Script Height Units (SHU) during computational phases and has five fuzzy linguistic terms in decision rule sets. SHU is a dynamic measure that is computed based on the

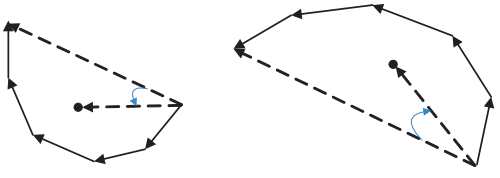

Fig. 8. Visualization of curvature direction. In the left picture, the point connecting the first point of the segment to its center of gravity (COG) lies on the left of the line connecting first to last points and is noted as Clockwise Curvature (C). In the right picture, the first point to COG lies on the right of the first to last points and therefore, it is called Counter Clockwise Curvature (C.C).

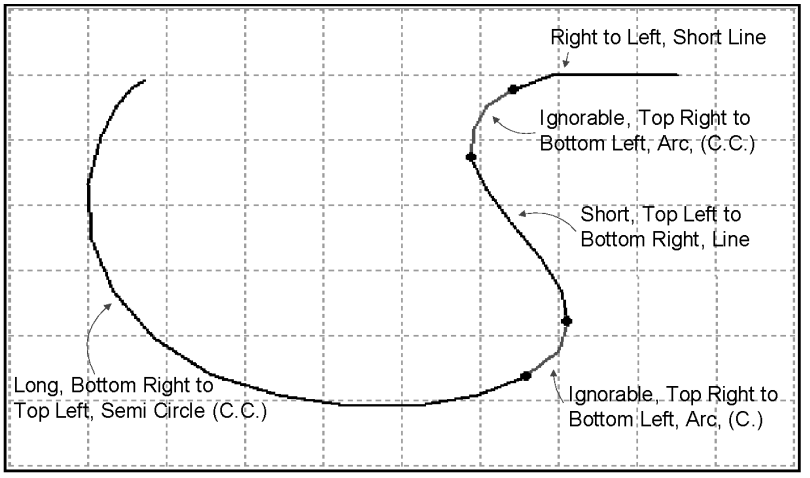

Fig. 9. A sample of segmentation and linguistic representation of the segmented input.

height of first few input episodes. To compute it, we delay the recognition till a few episodes are written, find the maximum height of recorded items, and set SHU equal to 0.1 of this value. This way, we dynamically adapt the length descriptions to writer's script size and use 0.1 of her/his writing size as our description precision. To specify this value using linguistic terms, five fuzzy linguistic variables namely *ignorable*, *short*, *medium*, *long*, and *very long* are used.

Figure 9 presents a sample of one segmented input and its description using the above feature set. Note that Persian is written right to left.

3.3. *Characters' pattern description*

Character patterns are defined with the same descriptive language as the inputs, but to make the comparisons simpler, we have restricted pattern definitions' segment types just to lines, therefore, each character will be represented by a sequence of line segments, each having the previously stated three attributes (direction, curvature, and length). Note that type attribute is omitted as it is always line.

The database includes 60 samples, including the main body of all letters in all possible forms. It must be noted that as we ignore the dots and other additive marks and also not all letters have all four possible writing forms, 60 samples described the characters in their different common writing styles. Figure 10 presents some samples of character definitions.

4. Pattern Matching

Up to this point, we translated the input data and character patterns into fuzzy linguistic expressions. This section presents our method to compare these expressions.

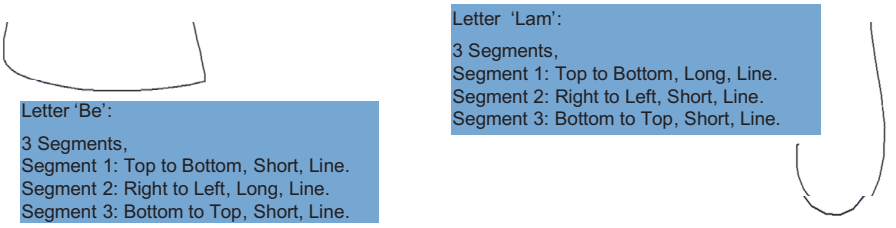

Fig. 10. Some samples for character pattern definitions, again note that Persian is written from right to left.

The major necessities of this part are:

- Coping with perturbations due to segmentation uncertainties.
- Ability to compare pattern and inputs with different number of segments, due to over segmentation or under segmentation (Fig. 11).
- There might be extra ignorable parts in the written text that are due to writer's style such as serifs, etc. or writing perturbations.

To have a process of creating multiple hypotheses and proving/disproving them as stated in Sec. 2, we divided the task into two levels: at the lower level, one part of one pattern is compared with one segment of input and some measures of difference are computed using a comparison look-up table; and at the higher level, a complete pattern is matched against the whole input.

4.1. *Low level comparison — one segment and one pattern part*

The aim of this subsection is to compare one part of input with one segment part. As shown in Fig. 12, there are cases in which input segment may not be exactly similar to the pattern part, but do partly match. For example, the input curve (A) of Fig. 12 can be assumed similar with pattern (B) but it has some extra length before the similar section, or the input arc (C) of the same figure has some similarity with pattern (D), but it also has some extra, non-similar length after the similar section. To make such comparisons, we define the comparison measure as a triple

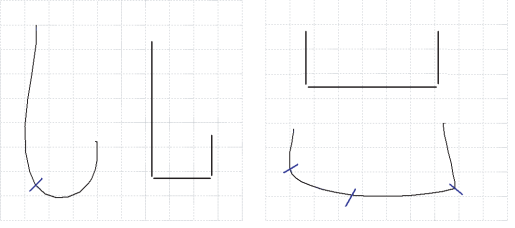

Fig. 11. A sample in which the input (the sketchy drawing) is separated into more/less number of parts from that of the pattern. The left input is called under-segmentation and the right one, over-segmentation.

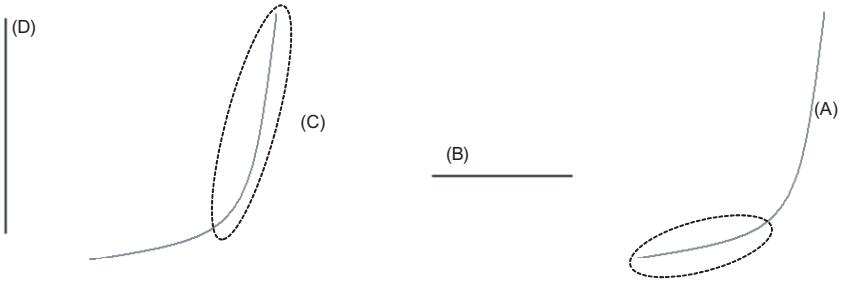

Fig. 12. Two samples for part to part comparison. (A) and (B) have some similar parts, preceded by some non-similarities in (A). (C) and (D) have a similar part, followed by a non-similar section in (C).

(*Extra Before, Matching Size, Extra After*) which specifies what percent of the two parts match, and how much extra did the input have before the matching part and how much did it have after the matching section. For example, input (A) of Fig. 12 matches with pattern Part (B) of the same figure with (70%, 30%, 0%) meaning that (A) has 30% similar Part with (B), but the similar section is preceded by 70% non-similar Part. Also, (C) matches (D) with (0%, 70%, 30%) meaning that (C) has 70% similar part with (D), followed by 30% non-similar part.

To make such comparisons, we have created a look up table, comparing a line segment with each possible input segment, along with their possible relative angles. Table 1 presents this table and Fig. 13 represents the visualization of these rules' if-parts. For example, third row of the table states that if the input part is an arc and it is in the same direction with the pattern part (which is a line), they match for 90% but the arc has 5% extra before and after the matching part. As another example, line 4 implies that a clockwise curved arc whose angle is one unit more than the line's angle (for example, the line is top to bottom while the arc is top-left to bottom-right) match in 75% of arc's length, but the arc has 25% extra after the matching part.

4.2. *High level comparison — an input and a pattern*

The second layer of comparison is on top of the primary comparisons of pattern and input parts. To do so and to compare a complete input, i.e. a sequence of segments, with a pattern, or a sequence of pattern parts, we treat the pattern sequence as an elastic definition for the character that can stretch or skew based on the requirements of the input sequence. To do so, we use a non-deterministic finite state automaton with some modification where the sequence of input segments will be regarded as machine's inputs and pattern parts are machine's states (Fig. 14).

But the input is not necessarily segmented the same as the given abstract pattern. Figure 15 presents three other possible segmentations for different writings that all indicate the pattern of Fig. 14. As presented there, the required machine for these inputs differs from that of Fig. 14 and therefore, we need a structure that would be flexible enough to overcome all such possibilities.

Table 1. Single part comparison rules for line patterns. This table is visualized in Fig. 13.

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

[a]Their difference based on our direction definitions. For example, top to bottom angle has one difference with either of {top left to bottom right} and {top right to bottom left}.

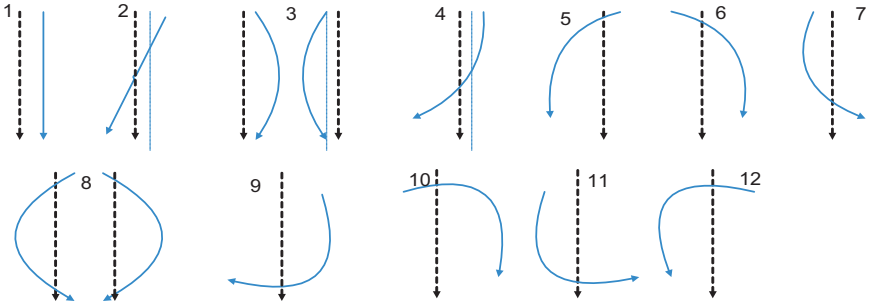

Fig. 13. Visualization of comparison rules for one line pattern (the dashed arrow) and different input segments.

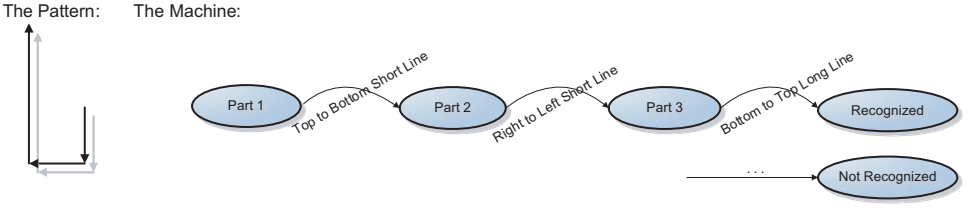

Fig. 14. Sample recognizer machine for an abstract pattern. Note that the other inputs that lead to failure in recognition are not drawn.

To overcome these problems, our machine has three major differences with conventional None Deterministic Finite State Automata (NDFSA)[21]:

(i) We maintain a set of current possible states instead of one single current state; each current possible state has a degree of belief and some other properties.

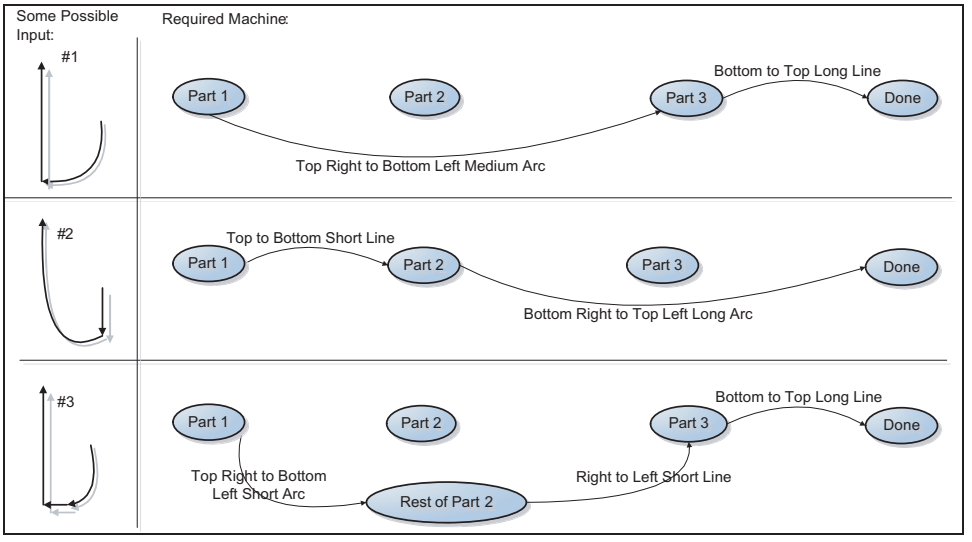

Fig. 15. Three sample inputs for the abstract pattern of Fig. 14.

(ii) Instead of defining strict state transfer rules, we have a set of templates for state transfer where each pattern has some perquisites.
(iii) Once a state loses certain capabilities, it is pruned out of a set of current possible states.

4.2.1. *Set of current possible states*

In a conventional NDFSA, there are usually several choices for the next state given a certain input, and if the machine is set to process the inputs chronologically, it has to choose one of these possibilities randomly.[21] To overcome this unknown random factor, we maintain all possible hypotheses about the meaning of the input sequence given so far, proceed with all possible paths by keeping all of them as the set of Current Sates, and prune the set only when some states are falsified. The set of current possible states has just one member when the machine starts and once each input segment is given to the machine, all current possible states will produce their next possible state(s) and the set of all these next states will be the next step's possible states.

4.2.2. *Parameterized state definition*

As the input is fed into the machine, different hypotheses are made on the meaning of different parts of it. Each of these hypotheses may have certain properties such as what length of the pattern is matched so far or how good is it matched here. To express these parameters, a state is defined as a five-dimensional vector (*POS, BEL, NOIS, CM, CF*) standing for *Current Position, Previous Belief, Previous*

Noise, Current Match, & Current Flaw. Table 2 presents the description of these fields.

4.2.3. *Templates for state transfer rules*

To describe state transition rules for the pattern matching machine, let us first study another sample (Fig. 16 and Table 3). To match the input sample of Fig. 16 with the abstract given pattern, we can consciously follow this path of matching:

(i) First, we need a top to bottom short line, but the first input segment is a right to left very small arc and they do not match at all. So, we skip the first input segment and assume that it is a non-matching length, adding 1 unit to noise.
(ii) We still need to match Part 1 of the pattern, and input segment 2 well matches pattern Part 1. Thus, we accept it for pattern Part 1, and move the focus of reading to pattern Part 2.

Table 2. The descriptions of state properties.

Property	Description
Current Position	The machine in now accepting this part of the pattern, therefore, it ranges from 1 to last pattern part.
Previous Belief	The matching percentage of previous parts of the pattern. For example, we might be accepting Part 3 and have totally matched Parts 1 and 2 with 75% match.
Previous Noise	The total same of non-matching parts till current point. This measure is expressed in SHU.
Current Match	How much of the input has matched the current pattern part, in SHU
Current Flaw	How much of the input has not matched the current part, in SHU.

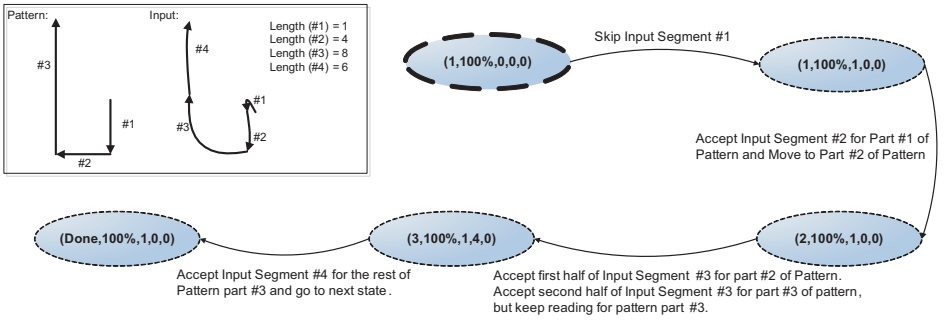

Fig. 16. A sample of input and abstract sample.

Table 3. Low level comparison table for Fig. 16. Each entree is (Extra Before, Matching Size, Extra After).

Input ↓	Pattern →	#1	#2	#3
#1		(0.5, 0, 0.5)	(0.2, 0.6, 0.2)	(0,0.2,0.8)
#2		(0, 4, 0)	(2, 0, 2)	(2, 0, 2)
#3		(4, 0, 4)	(0, 4, 4)	(4, 4, 0)
#4		(3, 0, 3)	(3, 0, 3)	(0, 6, 0)

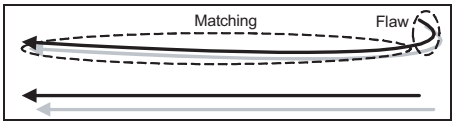

Fig. 17. Matching length and flaw.

(iii) Then comes input segment 3, which partly matches pattern Part 2, but it has a tailing that does not match pattern Part 2. But the extra tail matches pattern Part 3 while it is still short for it. Therefore, we accept input segment 3 as satisfying pattern Part 2 and some of pattern Part 3 and wait for more for pattern Part 3.

(iv) Finally, input segment 4 matches pattern Part 3, but it is short. Noting that we had an extra tail from input segment 3 which matched pattern Part 3 and that was short too, we can add these two parts and satisfy pattern Part 3 and state that the input and pattern are matched.

To express the above path following formally, we can first construct low level comparison table based on the triples of Sec. 4.1 which states how each segment of input matches each part of the pattern (Table 3). Then, following the state transitions of Fig. 16, we see that we have three classes of state transitions:

- SKIP: This is the action that happened at the first step of the above sample and happens when we want to ignore an input segment and add the flaws. Therefore, it changes the current state from (*POS, BEL, NOIS, CM, CF*) to (*POS, BEL, NOIS, CM, CF* + *Length of* (*I*)).
- ACCEPT, FORWARD MARCH: In this class of actions, the machine matches the given input with the current pattern part and moves to a next pattern part (what happened at step (ii) of previous example). Once this is done, the *Matching Size* of the current input is added to *Current Match* and its *Extra Before* and *Extra After* are added to *Current Flaw* as they do not take part in matching process and are not desired (see Fig. 17 for a sample).

But there are cases in which, the *Extra After* part matches the next pattern part and we must retain it, and this can even follow for a sequence of more than one pattern part. For example, as shown in Fig. 18, the input matches the first part

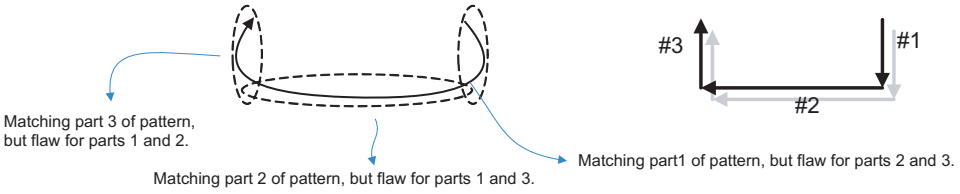

Fig. 18. Sample for forward marching more than one pattern part. The input segment matches partly each pattern part, and it can match the whole pattern.

of the pattern, but it has a long extra which matches the next pattern parts. In such cases, the machine can accept and pass all matching pattern parts and move to the next pattern part. The requirement of such transfers is that each of the matching patterns must have *Extra After* and *Extra Before* except the first and last parts where the first does not need *Extra Before* and the last does not need *Extra After*.

To express this formally, we use notation (3) and move from state *POS* to state $POS+n$ when reading I_i with formula (4) limitations and effects. It must be noted that if several (n)s satisfy the *if* part, all of them are applied and the results are added as next possible states.

$$\begin{cases} \textit{Input Segments} : I_1, \ldots, I_N \\ \textit{Pattern Parts} : P_1, \ldots, P_M \\ \textit{CurrentState} : (POS, BEL, NOIS, CM, CF) \\ \textit{Length}(I_x) : \textit{The length of input segment } x \textit{ in SHU units.} \\ \textit{LenMatch}(x, p) : \textit{The membership value of } x \textit{ in fuzzy length specified} \\ \qquad \textit{for pattern Part } p. \\ \textit{LowMatch}(I_i, P_j) = (EB_{i,j}, MS_{i,j}, EA_{i,j}), \textit{based on low level comparisons,} \\ \qquad \textit{representing Extra Before, Matching Size and Extra After.} \end{cases} \tag{3}$$

$$\textit{if} \begin{cases} POS + n \leq M + 1 \\ n > 0 \\ [n = 1] \vee \\ [(\forall x; POS < x < POS + n | EB_{i,x} > 0) \wedge \\ (\forall x; POS \leq x < POS + n - 1 | EA_{i,x} > 0)] \end{cases}$$
$$\textit{then add to Next Possible States}: \begin{cases} (POS + n, \\ BEL \times \textit{LenMatch}(CM + MS_{i,cp}) \\ \qquad \times \prod_{cp<x<cp+n} \textit{LenMatch}(MS_{i,cp+x}) \\ NOIS + EB_{i,CP} + EA_{i,CP+n-1}, \\ 0, 0) \end{cases} \tag{4}$$

$$
\begin{aligned}
&\textit{if}\\
&\begin{cases}
n \geq 0\\
POS + n \leq M\\
[n \leq 1] \vee\\
[(\forall x; POS < x \leq POS + n | EB_{i,x} > 0) \wedge\\
(\forall x; POS \leq x \leq POS + n - 1 | EA_{i,x} > 0)]
\end{cases}\\
&\textit{then add to Next Possible States}:\\
&\begin{cases}
(POS + n,\\
BEL \times LenMatch(CM + MS_{i,cp})\\
\qquad \times \prod_{cp<x<cp+n} LenMatch(MS_{i,cp+x})\\
NOIS + EB_{i,CP} + CF \times [n \neq 0]\\
\qquad + EA_{i,CP+n-1} \times [n = 0],\\
MS_{i,cp+x},\\
CF \times [n = 0] + EA_{i,CP})
\end{cases}
\end{aligned}
\tag{5}
$$

- ACCEPT, FORWARD MARCH, WAIT: As seen in the third step of Fig. 16 sample, there are cases where one input segment satisfies one or more pattern parts, and its tail partly satisfies the next pattern part. These cases are very much the same as the previous case (ACCEPT & FORWARD MARCH) but have a minor difference, formula (5) presents such transitions. Note that the bracket signs in the above formula are Iversonian brackets where $[x]$ is 0 when x is false and is 1 when x is true.

4.2.4. *State pruning*

Once a new state in (*POS, BEL, NOIS, CM, CF*) form is added to a set of next possible states, it is checked to see if its *BEL* is above a certain threshold and its *NOIS* is below another threshold and if the condition was not met, the state is thrown away and is not added to the next possible states. This is done to remove wrong hypotheses as soon as they are known to be not acceptable.

4.3. *The pattern recognition approach all put together*

Based on the definitions in Secs. 4.1 and 4.2, the complete pattern recognition system works as follows: once an episode of input is entered, segmented and described, the algorithm compares it with all stored character patterns and the pattern that gains the maximum similarity is chosen. To make the comparison for each pattern, *Compare Pattern* algorithm (Fig. 19), the current possible states are initialized with a single starting state (1, 100, 0, 0, 0). Each input segment transfers all *Current Possible States* to *Next Possible States* and during this transfer, the states that fail some validation rules are pruned. If during this comparison, a state reaches the end of the pattern while the input sequence is not finished yet, it can be assumed that the input episode may cover more than one pattern. Thus, these states are separated and at the end, the rest of the input is checked to match any other pattern(s).

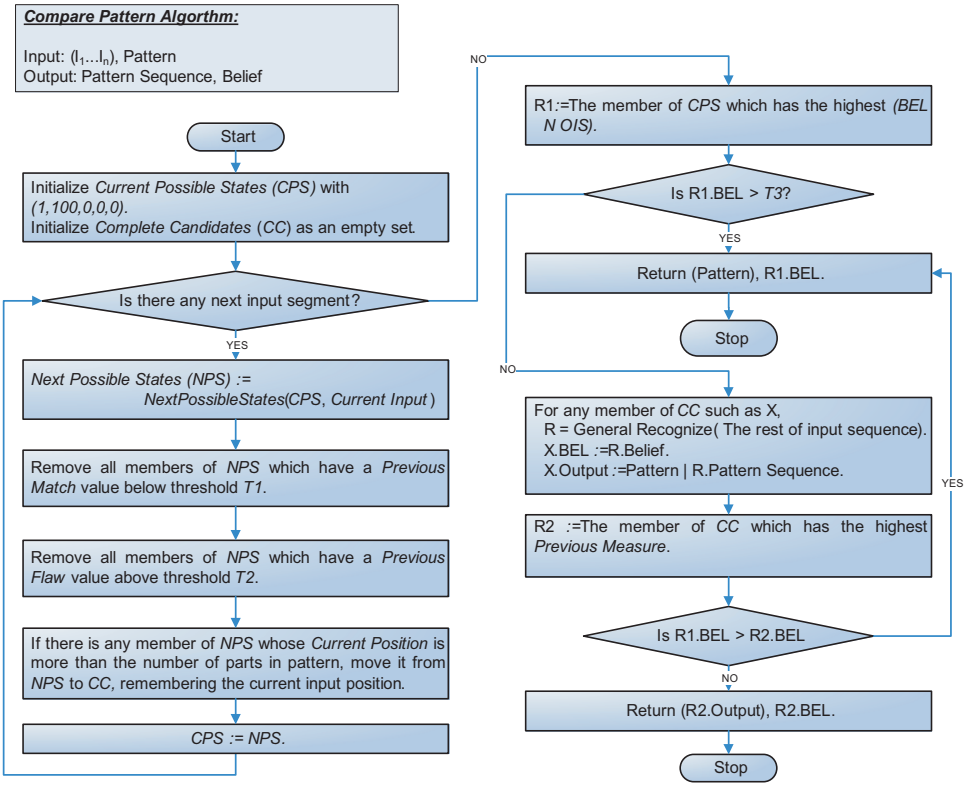

Fig. 19. Compare pattern algorithms.

Production of *Next Possible States* is also done through applying all possible state transfer templates on all *Current Possible States.* This is illustrated in Fig. 20.

5. Experimental Results

As there is no generally accepted benchmark to test the proposed algorithm for Persian handwriting and the test cases on none of the cited contributions for Persian or Arabic are available, we chose our test set from the first primary school book of Iranian schools which includes around 1250 words.[b] The test cases where written by 20 individuals where half of them were primary school kids and the other half were adults. The tests were also run once with a dictionary (the system knew the set of possible words) and once without dictionary. The test set was also simplified by removing the extra parts of the letters such as dots and other additives of Persian font. Table 4 presents the results.

[b]The list of words is available at http://ce.sharif.edu/∼halavati/Neveshtar/PrimarySchoolWords.txt

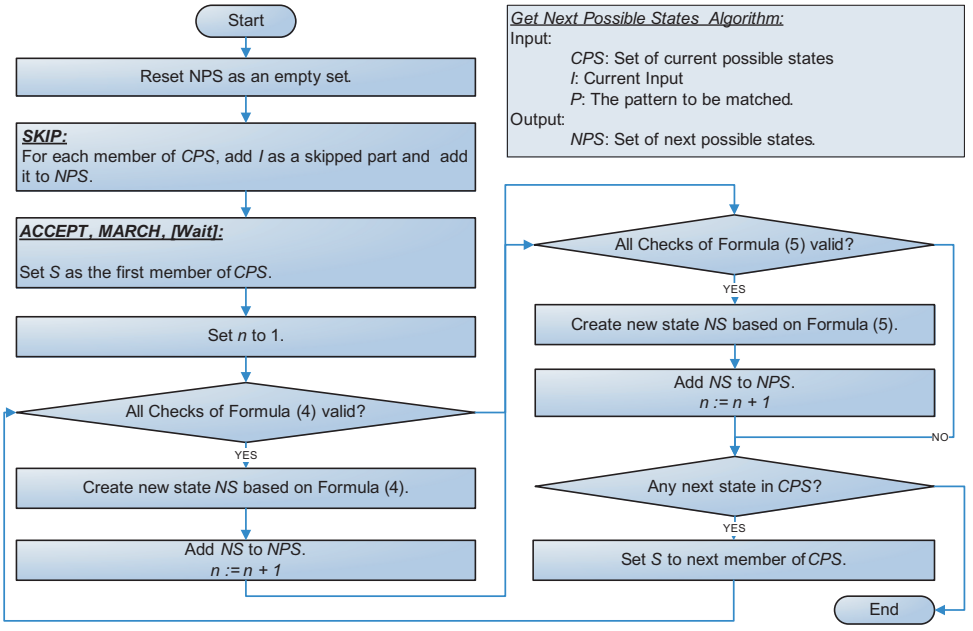

Fig. 20. Next possible state generator algorithm.

Table 4. Experimental results.

Writers Set	Without Dictionary	With Dictionary
Only Children	85%	98%
Only Adults	71%	92%
Both Together	78%	96%

As it was expected, the system has been notably more successful in recognizing writings of children as they are more faithful to writing rules. The system has successfully recognized children's writing in 98% of cases when the set of words was previously known and scored 85% when the input set was not previously mentioned. The results are 6% to 14% worse for grown up writers with and without dictionary due to their innovations in writing and having different styles.

Also to check the robustness of final recognition system to segmentation variations, we made a set of tests with a randomly chosen set of previous samples. To do so, the system was tested with 20 randomly generated parameter sets. In each set, the parameters were generated using a normal distribution whose averages were based on the values that we had used in previous experiments and the variances were 50% of average value (for example, the limit for selection of line in Part 3 was generated with $\mu = 20$ $\sigma = 10$). The recognition results had only 2% variance (Table 5). Therefore, we can conclude that this approach does not need a perfect

Table 5. Segmentation variation test results.

Line Threshold	Arc Threshold	Results
$\mu = 20$, $\sigma = 10$	$\mu = 7$, $\sigma = 3.5$	$\mu = 82$, $\sigma = 2$

tuning of parameters but there exist a set of best parameters, at least for a set of specific writers.

And to check the memory requirements of recognition during the hypothesis generation phase, we computed the average and maximum number of concurrent hypotheses and the results were 10 and 34, respectively indicating that the pruning rules did not let the number of hypotheses increase exponentially, as it could happen theoretically.

6. Comparisons with Other Approaches

In this section, we compare our approach's features and results with that of previous contributions on recognition of Persian or Arabic handwritings.

6.1. *Hierarchical rule-based approach*

In Ref. 11, El-Sheikh and El-Taweel presented an approach based on a hierarchical tree of rules based on the number of strokes in each segment of the written text while the text is segmented into characters before being processed by their algorithm. They have reported 100% recognition accuracy for their test cases and no specific description of training/test data is provided. Putting aside the assumption that a neat input to character segmentation algorithm must exist before application of their method, the approach is highly sensitive to writing perturbations as the rule's hierarchy is based on the number of strokes in each character and this can be a variable based on writers' styles and friction of writing device.

6.2. *Structural and fuzzy approach*

In Ref. 8, a hybrid system of structural and fuzzy techniques is presented, that receives segmented handwritten characters, specifies them with some fuzzy descriptive language and then, a handmade fuzzy rule base chooses the correct class of letter. They have claimed 100% recognition rate while their test set is not mentioned. Similar to previous approach, they need a neat input to character segmentation approach beforehand and they are also highly sensitive to additional or missing parts of the pattern.

6.3. *Template matching and dynamic programming*

Alimi and Ghobel,[6] presented an approach for recognition of isolated Arabic handwritten characters in which, the input data is compared with the set of character

prototypes after smoothing, normalization and coding. They have reported 96% recognition accuracy with 1 writer. Again the input is assumed to be neatly segmented. Also, the dynamic pattern matching is dot to dot, creating a very big search space and also sensitivity to writers' styles.

6.4. *K-nearest neighbor classification*

This approach[10] uses a K-Nearest Neighbor classifier system on a set of stable features such as the number of dots, relative position of dots, and number and position of other secondary strokes. The approach has resulted in 84% accurate recognition with seven writers. Initial character segmentation is preassumed and the approach is sensitive to addition/removal of any part.

6.5. *Evolutionary neuro-fuzzy approach*

Alimi[7] proposed a complete system that segments and recognizes input sequence using a set of six feature vectors. The input is segmented using a genetic process into letters and letters are identified using a fuzzy radial basis function. The system scored 89% for a single writer. Although it performs a robust segmentation using genetic algorithm, it becomes quite slow in average and with long words, and it is writer dependant.

6.6. *Neural network approach*

Mezghani[18] and Klassen[14,15] proposed approaches based on multilayer perceptron and self-organizing map neural networks, both to classify the main body of isolated letters. The first approach resulted in 95% correct recognition for multiple writer cases and the second resulted in 86.56% for single writer tests.

6.7. *Comparison with our approach*

As summarized in Table 6, in comparison with other approaches, the first major advantage of our approach is its ability to segment and recognize at the same time, while only one other approach does this and it requires considerable amount of computations (evolutionary approach). Also, the approach is one of the rare methods that are not writer dependant and can tolerate writing perturbations. As the data sets of other methods are not present, we cannot make a direct comparison in recognition accuracy but as it can be generally grasped, we have used a much wider test set including many writers.

Putting all together, we can conclude that this approach can recognize the written text user independently, insensitive to perturbation or minor styles, without supervision (for segmentation), and with acceptable result without dictionary and very good result with dictionary (1250 words).

Table 6. Comparison with other methods.

Approach	Requires Segmentation	Sensitivity to Perturbations	Number of Tested Writers	Uses Dictionary	Results
Hierarchical Rule Base	Yes	High	?	?	100%
Structural Fuzzy	Yes	High	?	?	100%
Template Matching	Yes	Average	1	?	96%
K-Nearest Neighbor	Yes	High	7	?	84%
Evolutionary Neuro Fuzzy	No	Average	1	?	89%
Our Approach without Dictionary	No	Low	20	No	78%
Our Approach with Dictionary	No	Low	20	Yes	96%

7. Conclusions and Future Works

Recognition of Persian handwriting is a complicated task as Persian is fully cursive, characters are written in up to four different forms in different parts of the world, and people write in several different styles. The presented approach works on online data, segments the input set into a sequence of lines, arcs and half-circles and represents these segments with fuzzy linguistic terms. While the letter patterns are also defined with similar language, a flexible comparison algorithm compares the input sequence with different character sequences.

As there is no general benchmark on Persian handwriting, the approach is tested with the set of words that are used in first primary Iranian school books and the results have been quite satisfactory for grown up writers and much better for children as they obey writing rules more than grown ups. The algorithm has also been tested with different parameter sets and it has shown very little sensibility to its settings.

The major advantages of this algorithm are: it is fast, yet quite robust to writing perturbations, it does not need adaptation phase for different writers, and it can segment input into letters automatically.

As the next step to this contribution, we are adding circles as another input token, making use of dots and other additives of Persian writing in the recognition system, and designing an active learning method to improve the ability of automatically refining definitions and adopting new styles.

Acknowledgments

We need to thank Ms. Mahdieh Soleymani and Mr. Reza Hesami Fard for their valuable discussions on different parts of this contribution and Dr. Mansour Jamzad for his support on project initiation.

References

1. K. Abbasian and E. Karbir, Online recognition of Persian script, *Proc. 6th Nat. Conf. Electrical Engineering*, Tehran, Iran (1999), pp. 146–141.

2. I. S. I. Abuhaiba, M. J. J. Holt and S. Datta, Recognition of off-line cursive handwriting, *Comput. Vis. Imag. Process.* **71**(1) (1998) 19–38.
3. M. Aksela, Handwritten character recognition: a palm-top implementation and adaptive committee experiments, Master's thesis, Department of Engineering, Physics, and Mathematics, Helsinki University of Technology (2000).
4. S. Al-Emami and M. Usher, On-line recognition of handwritten Arabic characters, *IEEE Trans. Patt. Anal. Mach. Intell.* **12**(7) (1990) 704–710.
5. A. M. Alimi, A neuro-fuzzy approach to recognize Arabic handwritten characters, *Proc. IEEE Int. Conf. Neural Network* **3** (1997) 1397–1400.
6. A. Alimi and O. Ghorbel, The analysis of error in an on-line recognition system of Arabic handwritten characters, *Proc. ICDAR 1995* (1995), pp. 890–893.
7. A. Alimi, An evolutionary neuro-fuzzy approach to recognize on-line Arabic handwriting, *Proc. 4th Int. Conf. Document Analysis and Recognition (ICDAR '97)* (1997), pp. 382–386.
8. F. Bouslama and A. Amin, Pen-based recognition system of Arabic character utilizing structural and fuzzy techniques, *Proc. Second Int. Conf. Knowledge-Based Intelligent Electronic Systems*, eds. L. C. Jain and R. K. Jain (1998), pp. 76–85.
9. W. Doster and R. Oed, Word processing with online script recognition, *IEEE MICRO* **4** (1984) 36–43.
10. M. El-Wakil and A. Shoukry, On-line recognition of handwritten isolated Arabic characters, *Patt. Recogn.* **22**(2) (1989) 97–105.
11. T. S. El-Sheikh and S. G. El-Taweel, Real time Arabic handwriting character recognition, *Patt. Recogn.* **24**(12) (1990) 1323–1332.
12. R. Halavati, S. H. Zadeh and Y. Mokri, How children read Persian?, *Sharif Uni. Technology Internal Report No 138412* (2004).
13. N. Joshi, G. Sita, A. G. Ramakrishnan and S. Madhvanath, Comparison of elastic matching algorithms for online Tamil handwritten character recognition, *Proc. Ninth Int. Workshop on Frontiers in Handwriting Recognition (IWFHR'04)* (2004).
14. T. J. Klassen, Toward neural network recognition of handwritten Arabic letters, Master of Computer Science Thesis, Dalhousie University, 2001.
15. T. J. Klassen and M. I. Heywood, Toward the on-line recognition of arabic characters, *Proc. 2002 Int. Joint Conf. Natural Networks (IJCNN'02)*, **2** (May 2002), pp. 1900–1905.
16. A. Malaviya and R. Klette, A fuzzy syntactic method for on-line handwriting recognition, *Advances in Structural and Syntactical Pattern recognition, SSPR'96*, Lecture Notes in Computer Science, Vol. 1121 (1996), pp. 381–392.
17. E. Mandler, Advance preprocessing technique for online script recognition of non connected symbols, *Proc. 3rd Int. Writing Computer Applications* (1987), pp. 64–66.
18. N. Mezghani and M. Cheriet, Combination of pruned Kohonen maps for online Arabic characters recognition, *Proc. 7th Int. Conf. Document Analysis and Recognition (ICDAR'03)* (2003), pp. 900–904.
19. R. Ranawana, V. Palade and G. E. M. D. C. Bandara, An efficient fuzzy method for handwritten character recognition, *Proc. KES2004* (2004), pp. 698–707.
20. S. M. Razavi and E. Kabir, Online recognition of Persian isolated characters, *Proc. 3rd Nat. Conf. Intelligent Systems, Kerman*, Iran (2005).
21. M. Sipser, Introduction to the theory of computation, PWS, Boston (1997), pp. 47–63.
22. C. C. Tappert, Cursive script recognition by elastic matching, *IBM J. Res. Develop.* **26**(6) (1982) 765–771.

23. C. C. Tappert and J. M. Kutzberg, Elastic matching for handwritten symbol recognition, *Proc. IBM Int. Conf. Image Processing and Pattern Recognition* (1978).
24. B. Wan, An interactive mathematical handwriting recognizer for the pocket PC, M.S. thesis, Department of Computer Science, University of Western Ontario (2002).

Ramin Halavati received his M.S. in artificial intelligence from Sharif University of Technology, (Tehran, Iran) in 2003, and is now pursuing the Ph.D. in artificial intelligence at the Computer Engineering Department of Sharif University of Technology.

His research interests include cognition, learning and consciousness, and his Ph.D. dissertation is on learning in pattern recognition.

Saeed Bagheri Shouraki received his Ph.D. in fuzzy control systems from the University of Electro-Communications (Tokyo, Japan) in 2000. He is now at the Sharif University of Technology (Tehran, Iran).

He has published more than 90 articles covering his research interests that include control, robotics, artificial life and soft computing.