Recognition of On-line Handwritten Arabic Digits Using Structural Features and Transition Network

Al-Taani Ahmad
Department of Computer Sciences, Yarmouk University, Jordan
E-mail: ahmadta@yu.edu.jo

Hammad Maen
Department of Computer Sciences, the Hashemite University, Jordan
E-mail: maen@hu.edu.jo

Keywords: on-line digit recognition, pattern recognition, feature extraction, structural primitives, document processing, transition networks

Received: September 12, 2006

In this paper, an efficient structural approach for recognizing on-line handwritten digits is proposed. After reading the digit from the user, the coordinates (x, y) of the pixels representing the drawn digit are used for calculating and normalizing slope values of these coordinates. Successive slope values are then used to record the change of direction which used to estimate the slope. Based on the changing of signs of the slope values, the primitives are identified and extracted. These primitives represent a specific string which is a production of a certain grammar. Each digit can be described by a specific string. In order to identify the digit we have to determine to which grammar the string belongs. A Finite Transition Network which contains the grammars of the digits is used to match the primitives' string with the corresponding digit to identify the digit. Finally, if there is any ambiguity, it will be resolved. The proposed method is tested on a sample of 3000 digits written by 100 different persons; each person wrote the 10 digits three times each. The method achieved accuracy of about 95% on the sample test. Experiments showed that this technique is flexible and can achieve high recognition accuracy for the shapes of the digits represented in this work.

Povzetek: V prispevku je opisana metoda prepoznavanja arabskih črk.

1 Introduction

In areas of automatic document analysis and recognition, the correct interpretation of digits is very important. Automatic recognition of on-line handwriting has a variety of applications at the interface between man and machine.

The performance of any system for handwriting recognition can be evaluated by several factors, such as size of the alphabet, independence of the writing style, and speed of recognition.

Automatic recognition of handwritten digits is difficult due to several reasons, including different writing styles of different persons, different writing devices, and the context of the digit. This leads to digits of different sizes and skews, and strokes that vary in width and shape.

Researchers in this field have proposed different approaches, such as statistical, structural, and neural network approaches [1, 2]. The main primitives that form digits are line segments and curves. Different arrangements of these primitives form different digits. To recognize a digit, we should first determine the structural relationships between the features make up the digit.

The syntactic and structural approaches require efficient extraction of primitives [3-5].

In this study, we propose an efficient approach for extracting features for handwritten digits recognition. First, we will review some related work. After introducing the Normalization and slope estimation method used in this paper, we will discuss the feature extraction algorithm used to extract the primary and secondary features. Then, we will give an overview of the proposed recognition approach. We will also illustrate how to resolve ambiguities in some digits. Finally, we will present and discuss the experimental results and draw some conclusions.

2 Previous works

The problem of handwriting recognition has been studied for decades and many methods have been developed. Verma [6] proposed a contour code feature in conjunction with a rule based segmentation for cursive handwriting recognition. A heuristic segmentation algorithm is used to segment each word. Then the segmentation points are passed through the rule-based module to discard the incorrect segmentation points and include any missing segmentation points.

You et al. [7] presented an approach for segmentation of handwritten touching numeral strings. They designed a neural network to deal with various types of touching observed frequently in numeral strings.

A numeral string image is split into a number of line segments while stroke extraction is being performed and the segments are represented with straight lines. Segmentation points are located using the neural network by interpreting the features collected from the primitives.

Olszewski [8] proposed a recognition approach that uses syntactic grammars to discriminate among digits for extracting shape-based or structural features and performing classification without relying on domain knowledge. This system employs a statistical classification technique to perform discrimination based on structural features is a natural solution. A set of shape-based features is suggested as the foundation for the development of a suite of structure detectors to perform generalized feature extraction for pattern recognition in time-series data.

Chan et al. [9] proposed a syntactic approach to structural analysis of on-line handwritten mathematical expressions. The authors used definite clause grammar (DCG) to define a set of replacement rules for parsing mathematical expressions. They also proposed some methods to increase the efficiency of the parsing process. The authors tested the proposed system on some commonly seen mathematical expressions and they claimed that their method has achieved satisfactory results, making mathematical expression recognition more flexible for real-world applications.

Chan et al. [10] discussed a structural approach for recognizing on-line handwriting. The recognition process starts when getting a sequence of points from the user and then by using these points to extract the structural primitives. These primitives include different types of line segments and curves. The authors demonstrated their approach on 62 character classes (digits, uppercase and lowercase letters). Each class has 150 different entries. They stated that experimental results showed that the recognition rates were 98.60% for digits, 98.49% for uppercase letters, 97.44% for lowercase letters, and 97.40% for the combined set.

Amin [11] reviewed the state of Arabic character recognition research throughout the last two decades. The author summarized all the work accomplished in the past two decades in off-line systems in an attempt to pin-out the different areas that need to be tackled.

Behnke et al. [12, 13] proposed a case study on the combination of classifiers for the recognition of handwritten digits. Four different classifiers are used and evaluated; Wavelet-Preprocessing Classifier, Structural Classifier, Neural Networks Classifier, and Combined Classifier.

3 Overview of the proposed system

3.1 Normalization and slope estimation

The user draws the digit on a special window using a digitizer (or mouse). Then the coordinates (x, y) of the pixels representing the drawn digit are saved on a file. These coordinate values are used for calculating and normalizing slope values. Successive slope values are then used to record the change of direction which used to estimate the slope [14].

The signs of the slope values (+ and -), the zero values, and the infinity values are saved and used in the feature extraction step. Figure 1 shows an example, the representation of the digit 2. Then, all primitives representing each digit are extracted. These primitives are identified by locating break points in the digit. Two types of break points are identified: Primary Break Points (PBP): slope values of infinity (∞) and Secondary Break Points (SBP): slope values of zero. The feature extraction process depends on the change of the slope signs around these break points. Infinity breakpoint is considered to be primary breakpoint since all primary features (Figure 3) needed for recognition have a slope value of infinity, and all the secondary features (Figure 5) have a slope of zero.

After the slope signs and values are computed, these values are normalized. The purpose of this step is to eliminate any distortion that might occur during the drawing process and to ease the primitive identification

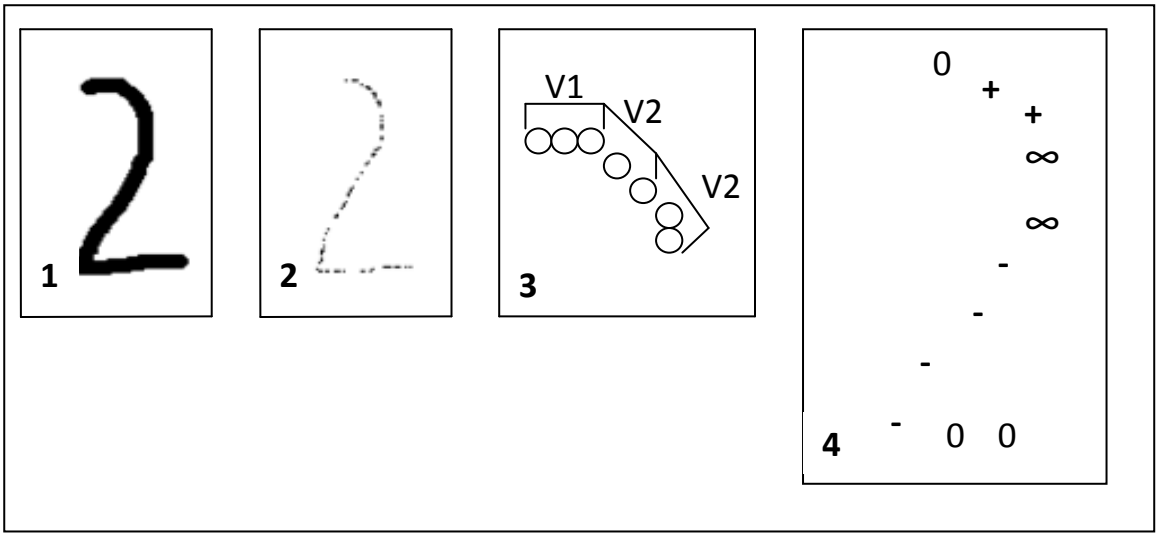

Figure 1: Representation of the digit 2

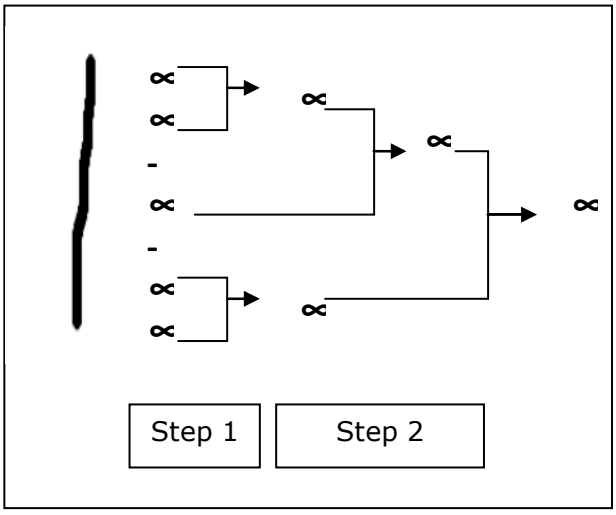

Figure 2: Removing Redundant Break Points

process and guarantee accurate identification. Two steps of normalization are done:

1. Removing redundant break points: Eliminating adjacent reference points of the same type, except the first one. This step is required to record the change of the signs; only one break point is needed.
2. A threshold value is used to determine the distance (number of slope value signs) between any two successive break points of the same type.

Figure 2 shows an example of these two steps for the digit 1. In step 1, the adjacent break points are removed, then in step 2 the threshold value was used to remove more redundant break points since the distance is less than the threshold value, e.g. the slope values between -0.1 and +0.1 is saved as zero. The result is only one break point with two different slope sings before and after it. This result will be used in the primitive identification process.

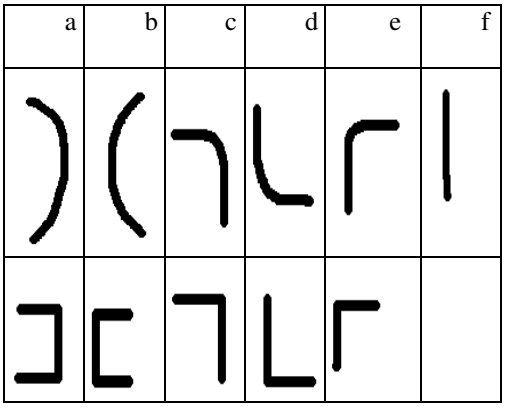

Figure 3: Primary Primitives

Final representation

In addition to the signs of the slope values and break points, the X and the Y positions for the middle pixel in which its neighbors (from both sides) used to calculate the slope. So the slope value is saved with its X and Y positions. Also, the sign of ΔY i.e. (Y2 – Y1) is used to determine the direction of writing or drawing (upward or downward). If ΔY > 0 and the reference point (0, 0) is located on the top left corner, then the directing of writing is downward, otherwise the directing of writing is upward. The final representation of the digit is represented as a list of vectors V1, V2, ... , Vn; each vector V contains the following data: (slope value (sign or break point), Y position, X position, ΔY sign) [15].

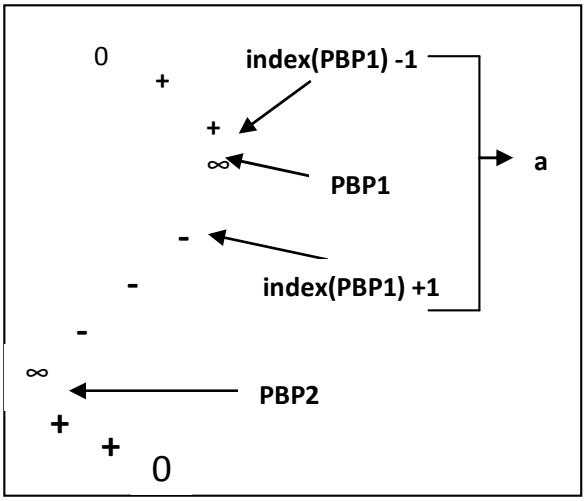

Figure 4: Signs and Break Points for the digit 2

3.2 Extracting primary primitives

The final representation of the digit is used to extract primary primitives [16]. Figure 3 shows these primitives (a, b, c, d, e, or f). These primitives are called primary primitives because the primary break points (PBP) are used to identify them.

To extract the primitives we use the following algorithm:

For each PBP do:
IF the slope sign before it is (+) and after it is (-) then the primitive is 'a'.
Else if the slope sign before it is (-) and after it is (+) then the primitive is 'b'
Else if the successive slope signs before it is (+) and end with SPB then the primitive is d'.
Else if the successive slope signs after it is (-) and end with SPB then the primitive is 'e'.
Else if the successive slope signs after it is (+) and end with SPB then the primitive is 'c'.

Figure 4 explains this step for the digit 2. Assume that

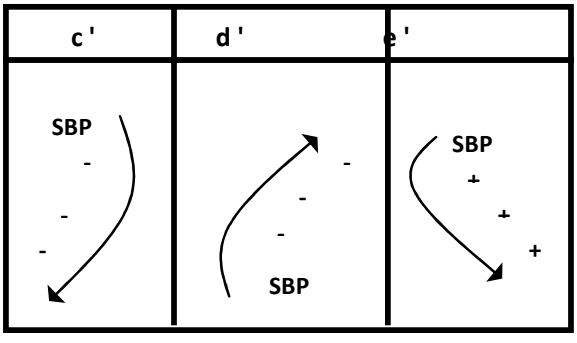

Figure 6: Identification process

the user draw the digit downward. In this case, ΔY is greater than zero for all points, so the algorithm proceeds as follows:

1. Take the first PBP1.
2. Now, primitive "a" is recognized.
3. Find the next PBP.
4. Take the next PBP (PBP2).
5. Now, primitive "b" is recognized.
6. No More PBP, end.

Now the vector contains the primitives "ab". If the user draws the digit from bottom to top, the vector will contain the primitives "ba". We need also to extract starting and ending points for the actual curves drawn that represented by primitives "a" and "b". This step is necessary to resolve ambiguity which will be explained in section 3.6. Each digit has different patterns which captured by the set of primitives that are described in Figure 3.

After feature extraction process, we need to identify these features. The primitives which are extracted in the previous phases represent a certain string which is a production of a certain grammar.

3.3 Extracting secondary primitives

After the process of extracting primary primitives finishes, another extracting process begins to identify another category of primitives called Secondary Primitives. Secondary Break Points (SBP), the zero values, and the slope signs around them are used to identify these primitives which shown in Figure 5. For example, the Secondary Primitive "c' " is the same as the primary primitive "c" but here there is no PBP and the primitive makes acute angle with SBP.

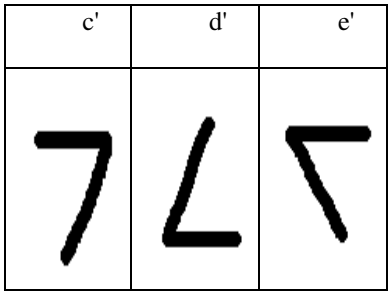

Figure 5: Secondary Primitives

To extract these primitives we use the following algorithm (Figure 6):

FOR each SBP that is not part of any primary primitive:

IF the slope signs after it are (-) and make lower acute angle then the primitive is "c'".

Else, if the slope signs after it are (+) and make lower acute angle then the primitive is "e"'.

Else, if the slope signs before it are (+) and make upper acute angle then the primitive is "d'".

3.4 Sorting primitives

At this stage, the primitives' vector contains primary and secondary primitives. The order of these primitives depends on the drawing style. For example, if the drawing style was downward (when drawing digit 2) then the primitives' vector will contain "ab", on the other hand if the drawing style was upward, it would contain "ba". This is confusing and increases the number of patterns for the digit. The order is very important in translating the primitives into digits. So we need a standard order to be used in sorting all primitives.

The order of the identified primitives must be independent from the drawing style and from the order of drawing the primitives. The standard order used here is the Y position for the break points; they are sorted in increasing order. After collecting all primitives, they are reordered according to the Y position for the break points that used to identify them. When two break points have the same Y position the order is based on the order of drawing. Indeed this is not a problem, because each character has many patterns, as we will see, to deal with such problems.

3.5 Identifying the digit

Each digit has different patterns which captured by the set of primitives, these patterns are shown in Figure 7. These primitives represent a specific string which is a production of a certain grammar. Each digit can be described by a specific string. In order to identify the digit we have to determine to which grammar the string belongs. A transition network has been used to match the primitives' string with corresponding digit. This network is shown in Figure 8.

3.6 Distinguish ambiguous digits

As we can see in Figure 8, there are multiple digits which have the same string of primitives, for example the string "ab" is common for digits 0 and 2. In this phase this ambiguity is removed, and more constrains on some digits are applied to guarantee the correct result. The key elements, that help us in resolving this ambiguity, are the starting and ending points, x-y coordinates, of the actual curves representing primitives "a" and "b". This process is done in phase (section) 3.2. Figures 9 and 10 show these points.

3.6.1 Ambiguity in "ab"

From Figure 8 we can see that the digits 0 and 2 both have one string "ba" which is one of their shapes. Now, the question is how this ambiguity can be distinguished? Assume that, as in Figure 9:

- (a1) is the x-y coordinates for the starting point of curve a.
- (a2) is the x-y coordinates for the ending point of curve a
- (b1) is the x-y coordinates for the starting point of curve b
- (b2) is the x-y coordinates for the ending point of curve b.

By using these definitions, we can easily distinguish between 2 and 0. Figure 10 shows the distinguishing criteria between digits 2 and 0.

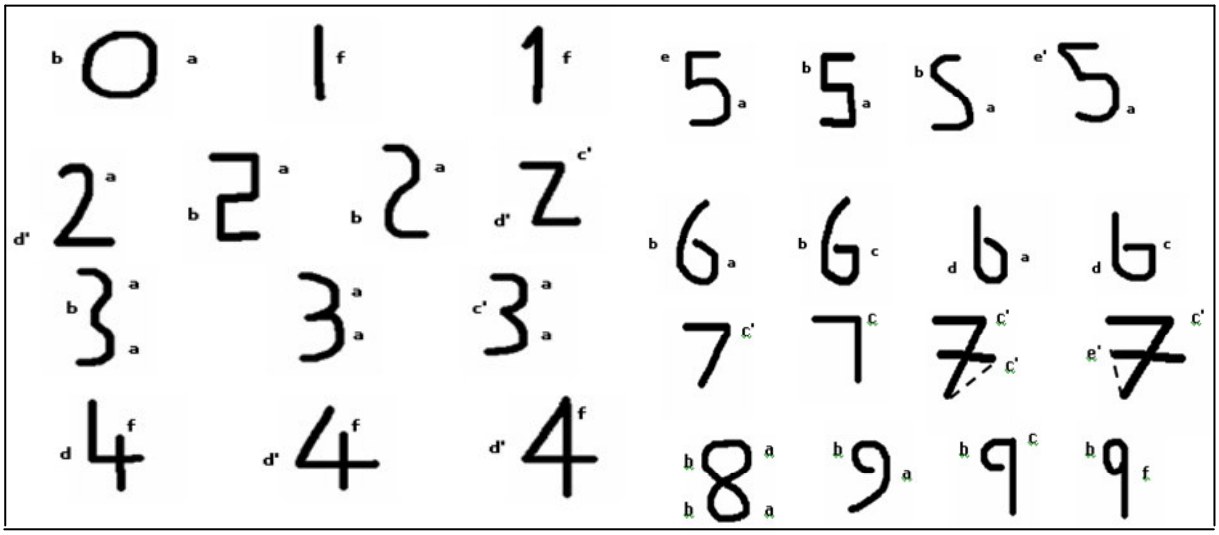

Figure 7: Possible handwritten patterns

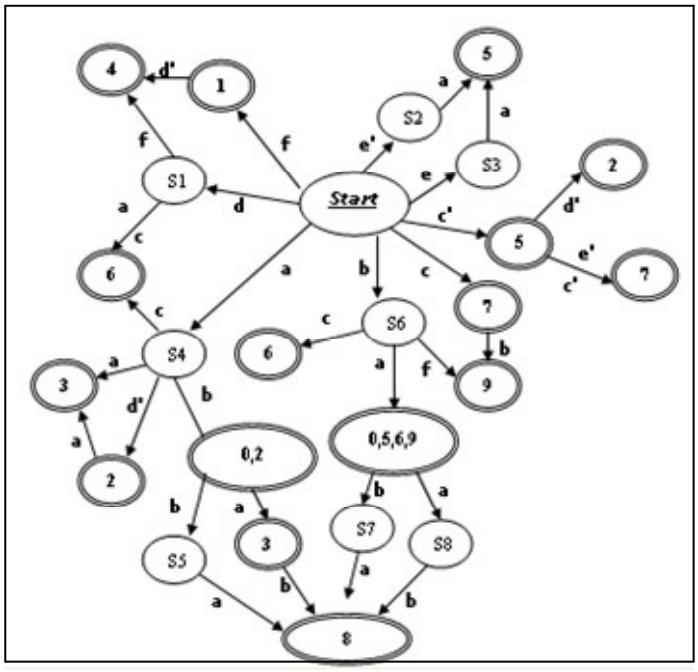

Figure 8: Transition Network

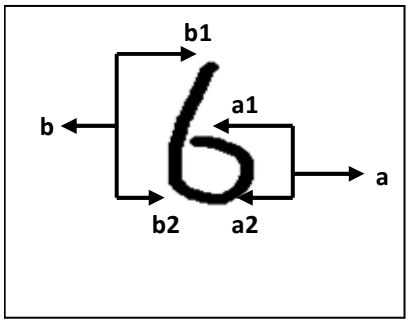

Figure 9: Ending Points

3.6.2 Ambiguity in "ba"

As we can see in Figure 8 the production string “ba" gives us the digits 0, 5, 6, and 9. The same definitions described in the above, in “ab”, are used here. The process is more complicated since we have 4 digits. The distinguishing criteria are described in Figure 11. The digit 2 can be easily distinguished by checking the point b1 if it above a1 or not. Now, to distinguish the three remaining digits we use the distance between the ending points of the drawing curves.

4 Experimental results and discussions

We used a digitizer with special software with a 1.6 MHZ PC. One hundred different persons tested the program. Each one of them drew all the numbers 3 times. So, for each number the program attempts to recognize it 300 times. As a result, there is only one possible outcome in the recognition process: correct or incorrect. We summarized the testing result for all digits in Figure 12. We can see that the system ability to recognize the drawing shape, shown in Figure 7, correctly is about 95%. There are 5% incorrect results, these results include both results; cannot identifying the drawing or identifying it incorrectly.

From the testing process we noticed the following important remarks:

1. The drawing speed may affect on the recognition process. If the user draws very quickly, the system might not capture all the input pixels representing the digit, i.e. the drawing must be connected, so the user has to draw the digit as one connected line. Only the digits 4 and 7 can be disconnected because

we included the suitable patterns to deal with the disconnected line in 4 and 7.

2. The accuracy rate for digits 6 and 9 is quite low. This is because they have the same patterns, i.e. they have the same production string "ba" (see Figure 11).
3. The proposed system works only on Arabic digits. We did not consider non-digits like letters and special characters. This task is left for future work

It can be noticed that the accuracy of the proposed approach is lower than the accuracy of Chan et al. [10] work in which they have achieved a recognition rate of 98.60%. The accuracy of the system depends on many factors like whether there is noise in the test data, if the digit is poorly written, deliberately written in some strange and unusual way, or with zig-zag line segments. We should take also into account that the writing process itself is subjective and depends on the person writing style. If the test data are carefully selected then the system could give higher accuracy rate.

Despite these factors our approach has the following advantages:

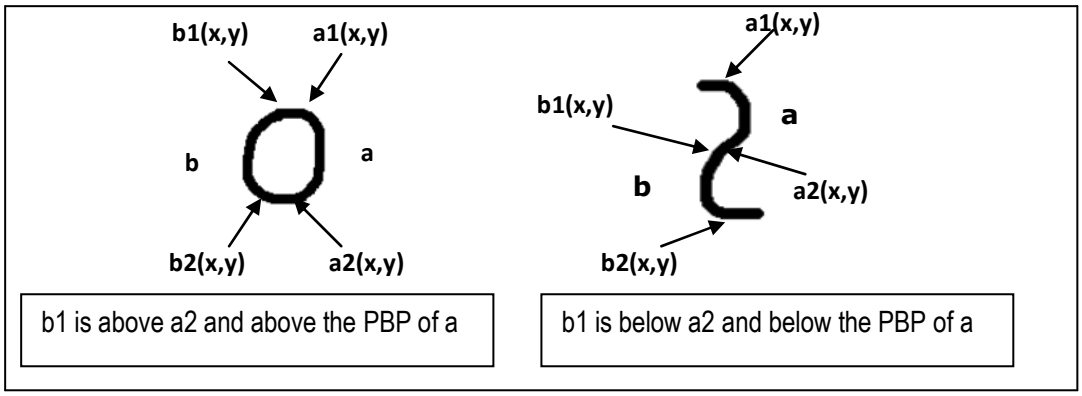

Figure 10: Resolving ambiguity in "ab"

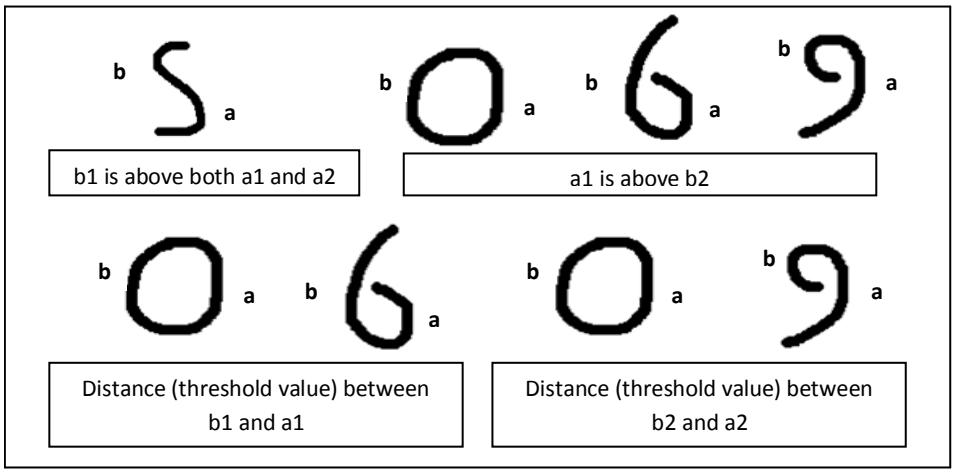

Figure 11: Resolving ambiguity in "ba"

Digit	Correct	Percent
1	288/300	0.96
2	290/300	0.97
3	289/300	0.96
4	288/300	0.96
5	285/300	0.95
6	274/300	0.91
7	287/300	0.96

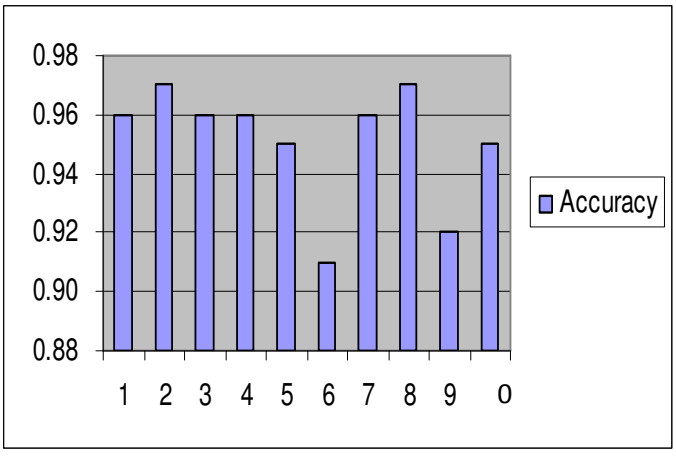

Figure 12: Test Results

1. In the proposed approach, we used shape-based features like curve, line, dot ... etc. together with a flexible Transition Network Grammar in the recognition process. Our experiment demonstrated that the use of shape-based features achieve fairly good recognition results since these features are used by people visually to recognize digits. Also, we used Transition Network since it works in the same manner as the human information processing system does which reflects one of our main objectives in this work, to design an intelligent agent which behaves rationally like humans.
2. The proposed approach can be modified to work on letters and other characters.
3. The proposed approach is unsupervised, i.e. training is not necessary.

5 Conclusions and future work

A new online structural pattern recognition approach is discussed. This approach recognizes the handwritten digits; the primitives are determined by identifying the changes in the slope's signs around the zero and the infinity values (break points). This technique is independent of the type of drawing (upward or downward). A special grammar has been used to match the string of primitives to the corresponding digit. The method is tested on an on-line dataset representing the digits 0-9 collected from 100 users. On the average, the recognition rate was about 95%. Future work considers testing the method on a larger data set to improve the effectiveness of the method, since we get most of the writing variations of the digits by different users.

The proposed method will be modified to deal with Arabic handwritten characters. In addition, the next important work is to add additional constraints on the primitives, for example the average length of one primitive according to another and do the primitives connected correctly or not. These constrains can guarantee an accurate results and do not directly match the resulting string of primitives to its corresponding digit unless the primitives form the digit correctly, one important note here is that, more constraints may reduce the probability of recognition

References

[1] C. C. Tappert, C. Y. Suen, and T. Wakahara. The state of the art in on-line handwriting recognition. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 12 (8), pp. 787-808, 1990.

[2] R. G. Casey and E. Lecolinet. Strategies in character segmentation: A survey. *Proceedings of International Conference on Document Analysis and Recognition*, pp. 1028-1033, 1995.

[3] K. S. Fu. *Syntactic Pattern Recognition and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1982.

[4] T. Pavlidis. *Structural Pattern Recognition*. Springer, New York, 1977.

[5] S. Lucas, E. Vidal, A. Amiri, S. Hanlon, and J.C. Amengual. A comparison of syntactic and statistical techniqes for off-line OCR. in: R. C. Carrasco, J. Oncina (Eds.), *Grammatical Inferrence and Applications (ICGI-94)*, Springer, Berlin, pp. 168-179, 1994.

[6] Brijesh Verma. A Contour Code Feature Based Segmentation For Handwriting Recognition. *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)*, Vol. 1, PP. 1203 – 1207. 2003.

[7] Daekeun You and Gyeonghwan Kim. An approach for locating segmentation points of handwritten digit strings using a neural network. *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003),* Vol. 1, PP. 142 – 146.

[8] Robert T. Olszewski. Generalized Feature Extraction for Structural Pattern Recognition in TimeSeries Data. *PhD thesis*, University-Pittsburgh, 2001.

[9] Kam-Fai Chan and Dit-Yan Yeung. An effecient syntactic approach to structural analysis of on-line handwritten mathematical expressions. *Pattern Recognition,* Vol. 33, pp. 375 - 384, 2000.

[10] Kam-Fai Chan and Dit-Yan Yeung. Recognizing on-line handwritten alphanumeric characters through flexible structural matching. *Pattern Recognition,* Vol 32, pp. 1099 - 1114, 1999.

[11] Adnan Amin. Off-Line Arabic Character Recognition: The State Of The Art. *Pattern Recognition*, 31 (5), pp. 517 - 530, 1998.

[12] Sven Behnke, Marcus Pfisher, and Raul Rojas, "A Study on the Combination of Classifiers for Handwritten Didit Recognition", Proceedings of Neural Networks in Applications, Third International Workshop (NN'98), Magdeburg, Germany, pp. 39-46, 1998.

[13] Sven Behnke, Raul Rojas, and Marcus Pfister. Recognition of Handwritten Digits using Structural Information. *Proceedings of the International Conference of Neural Network, Houston TX*, Vol. 3, pp. 139 1- 1396, 1997.

[14] S. Madhvanath, G. Kim, and V. Govindaraju. Chaincode Contour Processing for Handwritten Word Recognition. *IEEE Trans. On Pattern Analysis and Machine Intelligence*, 21 (9), pp. 928 - 932, 1999.

[15] Robert Schalkoff. *Pattern Recognition: Statistical, Structural, and Neural Approaches.* John Wiley and Sons Inc. 1992.

[16] Rafael C. Gonzalez and Michael G. Thomason. *Syntactic Pattern Recognition: An Introduction.* Addison Wesley Publishing Company, 1978.

[17] H. Freeman. Computer processing of line drawing images. *ACM Computing Surveys (CSUR),* 6 (1), pp. 57 – 97, 1974.