Application of Fuzzy Logic to Online Recognition of Handwritten Symbols

John A. Fitzgerald, Franz Geiselbrechtinger, and Tahar Kechadi
Department of Computer Science
University College Dublin
Belfield, Dublin 4, Ireland.
john.fitzgerald@ucd.ie

Abstract

Fuzzy logic is highly suitable for dealing with uncertainty and variation. Therefore it is seems reasonable to apply this technique to the recognition of handwritten symbols. This paper presents an approach to the task in which fuzzy logic is used extensively. We present a three-phase process, the central phase being feature extraction. Firstly a pre-processing phase generates a chord vector for each handwritten stroke, thereby eliminating noise and greatly reducing the number of sections of the input which need to be assessed as potential features. In the feature extraction phase fuzzy rules are used to determine membership values of chord sequences in fuzzy sets corresponding to feature types, and subsequently the most likely set of features is determined. In the final phase, fuzzy classification rules are used to determine the most likely identity of the symbol according to the feature extraction result. The approach has achieved high recognition rates in experiments on isolated symbols from the UNIPEN database.

1. Introduction

Many attempts at automatic handwriting recognition have involved feature extraction [1]. Feature extraction is a process whereby the input data is transformed into a set of features which characterise the input, and which can therefore be used to classify the input. Due to the nature of handwriting with its high degree of variability and imprecision, obtaining these features is a difficult task. A feature extraction algorithm must be robust enough that for a variety of instances of the same symbol, similar feature sets are generated, thereby making the subsequent classification task less difficult.

Fuzzy logic [2] is particularly useful for extracting features from handwritten symbols. It is clearly preferable to retain fuzzy rather than Boolean information regarding the extent to which sections of the symbols are curved, or horizontal. By extracting features using fuzzy logic [3][4] a greater understanding of what is present in the symbol is achieved in relation to neural network [5] or k-nearest neighbour [6] methods, and ultimately a more informed decision is made regarding the identity of each symbol. Furthermore, the extensive training phases required by neural network and k-nearest neighbour methods can be avoided.

In this paper an approach for online recognition of handwritten symbols is presented which incorporates a powerful new feature extraction technique using fuzzy logic. The approach is for recognition of symbols composed of one or more *strokes*. Each stroke is a sequence of points obtained by sampling the position of the pen at regular time intervals as it moves on a data tablet between a pen-down and a pen-up event. The approach consists of three phases: chording, feature extraction and classification.

The chording phase is a pre-processing phase which transforms each stroke into a vector of chords, such that each chord approximates a sector of a circle. This phase simplifies the input data so that feature extraction rules can be written in terms of chords rather than sequences of points.

In the feature extraction phase we distinguish only three types of feature: *Line*, *C-shape*, and O-*shape*. We believe that any symbol can be represented by a combination of these basic features, and that this is the most intuitive way to describe any symbol. Fuzzy rules have been developed to determine membership values for *substrokes* (sequences of consecutive chords) in fuzzy sets corresponding to the three feature types. Based on these membership values the best set of substrokes is chosen as the feature extraction result.

In the classification phase, hand-crafted fuzzy classification rules are used to determine the most likely identity of the symbol represented by the feature extraction result. In the remainder of the paper, the three phases of chording, feature extraction and classification will be discussed, followed by sections containing experimental results and conclusions.

Proceedings of the 9th Int'l Workshop on Frontiers in Handwriting Recognition (IWFHR-9 2004)
0-7695-2187-8/04 $20.00 © 2004 IEEE

2. Chording

The chording task amounts to partitioning each stroke s into a chord vector $\overrightarrow{C} = \langle c_0, \ldots, c_{n-1} \rangle$, where each chord c_i is a sequence of points ${p_1}^i, \ldots, {p_k}^i$ of s such that the last point of c_i and the first point of c_{i+1} coincide.

A partition is a *chording* if the points contained in each section $c = p_1, \ldots, p_k$ approximately form a sector of a circle with the line segment $[p_1, p_k]$ as chord.

Since our goal is to produce a chording with as few chords as possible an iterative solution based on merging chords suggests itself. Starting with the original stroke as coarsest chording the chording algorithm sweeps through the current chording and replaces two successive chords a and b by a new chord if the chords satisfy the *smoothing* and *shape formation* rules in addition to the requirement that the change in direction from chord a to b is less than 90 °.

2.1. Smoothing

Initially a *smoothing* rule is used to test if two successive chords a and b can be seen as a straight line. The decision is based on two properties of chords, *height* and *curvature*.

For a section c, $height(c)$ is defined as the maximal distance from c to the line segment $[c_f, c_l]$, where c_f and c_l are the first and last points of c respectively. We define the curvature $curv(c)$ as $2 * height(c)/|[c_f, c_l]|$, and we define c as *straight* if $height(c)$ and $curv(c)$ are below threshold values h_0 and v_0 respectively.

The smoothing rule can now be stated simply.

- Two straight chords a, b can be merged if the combined chord $a + b$ is also straight.

Smoothing compresses the original stroke data considerably. Since the calculations involved are rather cheap, smoothing is quite effective. Also this simple rule compares favorably to other techniques [7].

2.2. Shape Formation

Once smoothing is finished, a *shape formation* rule is applied which tests if two successive chords approximate a sector of a circle. The basic rule is as follows.

- Join two chords only if merging results in a chord with larger curvature, without significantly distorting the shapes of the original chords.

Whether or not the shapes of the chords are distorted significantly is tested as follows. Given two successive chords a and b determine the circle C through a_f, a_l and b_l and calculate distortion factors δ_a and δ_b. The distortion factor δ_a is obtained from the curvature $curv(a)$ and the curvature C_a of the circle sector of C for a. If δ_a and δ_b are below a threshold d_0, then C is a good approximation of the section formed by a and b, and therefore a and b can be merged.

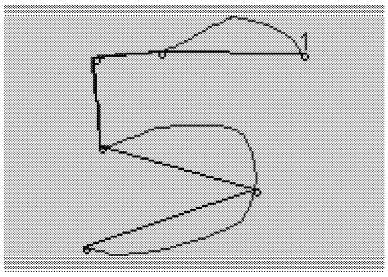

Figure 1. The chords produced for a 5

Chording generates chord vectors $\langle \overrightarrow{C}_0 \ldots, \overrightarrow{C}_v \rangle$ for the strokes $\{s_0, \ldots, s_v\}$ in the symbol. For instance, the chord vector for the stroke in Fig.1 is as follows.

chord	angle	length	height	curv
c_0	179	82	17	0.47
c_1	185	33	0	0
c_2	274	43	0	0
c_3	345	102	27	0.67
c_4	198	99	18	0.40

The purpose of chording is to eliminate noise in the original written stroke and to retain only the information which is essential for the feature extraction task. The chording algorithm presented here achieves this goal in a simple and efficient manner. Furthermore, chording identifies the locations in the stroke where new features may begin, such as turning points, flexion points or intersection points, and so the number of sections of the stroke which need to be assessed as potential features is drastically reduced.

3. Feature Extraction

The chord vectors $\langle \overrightarrow{C}_0 \ldots, \overrightarrow{C}_v \rangle$ are the input to the feature extraction phase, in which the objective is to identify the feature set for the symbol. The feature set will be the set of substrokes $F = \{f_0, \ldots, f_{m-1}\}$ encompassing the entire symbol which is of a higher quality than any other possible set of substrokes. Each substroke f_j is a sequence of consecutive chords $(c_a, \ldots, c_b)$ from a chord vector $\overrightarrow{C}_i = \langle c_0, \ldots, c_{n-1} \rangle$, where $0 \leq a \leq b \leq n$ and $0 \leq i \leq v$.

The quality of a set of substrokes, represented by $q(F)$, is dictated by the membership values of the substrokes in F in sets corresponding to feature types. The membership value of a substroke f_j in the set $Line$, for example, is expressed as $\mu_{Line}(f_j)$ or $Line(f_j)$, and represents the level of confidence that f_j is a line. In the definition of $q(F)$

below, T is whichever of the fuzzy sets *Line*, *C-shape* or *O-shape* f_j has highest membership in.

$$q(F) = \frac{\sum_{j=0}^{m-1} \mu_T(f_j)}{m}$$

Example: For the symbol in Fig. 1, the effect of feature extraction is a partition of the input $\overrightarrow{C} = \langle c_0, \ldots, c_4 \rangle$ into a set of features $F = \{(c_0, c_1), (c_2), (c_3, c_4)\}$, where $\mu_{Line}(c_0, c_1) = 0.66$, $\mu_{Line}(c_2) = 0.98$, and $\mu_{Cshape}(c_3, c_4) = 0.93$. It remains to be explained how such membership values are determined using fuzzy rules.

3.1. Fuzzy Feature Extraction Rules

The fuzzy rule base contains both *high-level* and *low-level* rules. Membership values in fuzzy sets corresponding to feature types are determined by high-level rules. Each high level fuzzy rule defines the *properties* required for a particular feature type, and is of the form:

$$T(Z) \leftarrow P_1(Z) \bigcap \ldots \bigcap P_k(Z)$$

This means that the likelihood of a substroke Z being of feature type T is determined by the extent to which the properties P_1 to P_k are present in Z. In mathematical terms,

$$\mu_T(Z) = \min(\mu_{P_1}(Z), \ldots, \mu_{P_k}(Z))$$

Membership values in fuzzy sets corresponding to properties are determined by low-level fuzzy rules. In each low-level rule the fuzzy value $\mu_{P_i}(Z)$ is defined in terms of values representing various aspects of the substroke Z. To express varying degrees of these aspects we use *fuzzy membership functions* such as the S-function [8], Π-shaped function and triangular function [9].

It is beyond the scope of this paper to discuss all the fuzzy rules in detail. A brief description of the properties required for each feature type will be given, along with the definitions for *Line*, *C-shape* and *O-shape*. To illustrate the concept of the low-level fuzzy rules the most basic rule will be described, the rule which determines the straightness of a substroke.

3.1.1 Definitions of Line, C-shape and O-shape

A line is characterised primarily by straightness. In addition it should be smooth (the amount of direction change should be minimal), it should be of significant length in relation to the length of the symbol, and it should not contain any sharp turning points. Therefore a substroke $Z's$ membership value in the set $Line$ is dictated by the extent to which straightness (STR), smoothness (SMO) and length ($LONG$) are present, and is adversely affected by the extent to which sharp points are present (SP).

C-shapes and *O-shapes* have certain properties in common. Both are not straight, do not contain major sharp turning points or discontinuity points (DP), do not contain both left and right turns (LR), and are continually curved throughout (CC). In addition to these common properties, a *C-shape* does not end near to where it started (SNE), does not turn through too many degrees (DEG), does not intersect itself (INT), and is open ($OPEN$) so that neither endpoint turns in to make it resemble shapes such as '6' or 'e'. An *O-shape* is round (RD) and its start point is in close proximity to its end point.

$Line\,(Z) \leftarrow\ STR(Z) \cap SMO(Z) \cap LONG(Z) \cap \neg\, SP(Z)$

$Cshape\,(Z) \leftarrow\ CC(Z) \cap LONG(Z) \cap OPEN(Z) \cap \neg\,(STR(Z) \cup SP(Z) \cup DP(Z) \cup LR(Z) \cup SNE(Z) \cup DEG(Z) \cup INT(Z))$

$Oshape\,(Z) \leftarrow\ SNE(Z) \cap CC(Z) \cap RD(Z) \cap LONG(Z) \cap \neg\,(SP(Z) \cup DP(Z) \cup LR(Z) \cup DEG(Z))$

The range of properties required for each feature type, and the rules which assess the extent of these properties, were continually updated over time until the memberships being produced for the feature types were deemed accurate.

3.1.2 Example of a Low-level Rule - STR

The straightness of a substroke depends on how direct a route it takes from its start point to its end point. Therefore the membership value of Z in the set STR is proportional to $ratio(Z)$, where $ratio(Z) = \frac{d(Z)}{len(Z)}$.

$d(Z)$ is the direct distance between Z's start point and end point, and $len(Z)$ is the absolute length of Z. If Z is a perfectly straight line, $ratio(Z) = 1$. After testing it was concluded that if $ratio(Z)$ is less than 0.8, Z is certainly not straight and should have a membership of 0 in the set STR. We therefore define $STR(Z)$ as follows.

$$STR(Z) = S(ratio(Z), 0.8, 0.9, 1)$$

The S-function, so named because of its appearance when plotted, is defined below. The first argument x represents some aspect of an object whose membership in some fuzzy set is being determined. The lower limit of acceptability for x is represented by a, below which the value of the function will be 0. As x increases between a and c, the value of the function increases from 0 to 1. The upper limit is represented by c, above which the result will be 1.

$$S(x, a, b, c) = \begin{cases} 0 & \text{if } x \leq a \\ 2(\frac{x-a}{c-a})^2 & \text{if } a \leq x \leq b \\ 1 - 2(\frac{x-c}{c-a})^2 & \text{if } b \leq x \leq c \\ 1 & \text{if } x \geq c \end{cases}$$

3.2. Feature Extraction Algorithm

The fuzzy feature extraction rules form the basis of the feature extraction algorithm. Given the chord vectors $\langle \overrightarrow{C}_0 \ldots, \overrightarrow{C}_v \rangle$ as input, $X = \{s_0, \ldots, s_{k-1}\}$ is the set of all possible substrokes. For each chord vector $\overrightarrow{C}_i = \langle c_0, \ldots, c_{n-1} \rangle$ there are $\frac{n(n+1)}{2}$ substrokes in X. Given X and the fuzzy rules, an algorithm is required to determine the best feature set as efficiently as possible.

The simplest feature extraction algorithm would involve the following steps.

1. Apply every low-level fuzzy rule to every substroke in X, generating membership values in sets corresponding to properties.
2. Subsequently apply high-level rules to every substroke in X, generating membership values in the sets *Line*, *C-shape* and *O-shape*.
3. Compile all possible sets of substrokes which encompass the entire symbol.
4. Compute $q(F)$ for each set of substrokes F and then choose the set for which $q(F)$ is maximal as the feature extraction result.

This algorithm involves an unnecessary amount of computation, and is improved considerably by the following efficiency measures.

- By initially identifying the sharp turning points in the symbol, the chord vectors are divided into subsets, whereby the sharp points are the boundaries between the subsets. Now the only substrokes which need to be evaluated are those which lie entirely within a subset, on the basis that a substroke containing a sharp point is unlikely to be a feature. This can drastically reduce the number of substrokes to be evaluated in step 1.
- During the evaluation of a substroke Z, once a membership of $\mu_{P_i}(Z) = 0$ is produced the investigation into Z being any feature type which requires the property P_i is halted.
- In step 3 *low-quality substrokes* (substrokes with memberships below a threshold l_0 for all three feature types) and *embedded substrokes* (e.g. a *C-shape* within an *O-shape*) are put aside. From the remaining substrokes all possible sets of substrokes are compiled (there is often only one possible set) and the set for which $q(F)$ is maximal is forwarded for classification.

It is possible that a low-quality substroke, an embedded substroke or a substroke containing a sharp turning point is crucial to the eventual recognition of the symbol. Such substrokes can be resorted to if the classification process fails (see section 4.3).

4. Classification

The objective of this phase is to determine the most likely identity of the symbol. The input is a set of substrokes $F = \{f_0, \ldots, f_m\}$ which can now be regarded as the features of the symbol. The likelihood that the features in F form an a, for example, is represented by the membership value $\mu_a(F)$. Such membership values are determined by fuzzy classification rules.

4.1. Fuzzy Classification Rules

To illustrate how the classification rules work this section includes a description of the rule for the digit 3 as an example. F must contain two features whose memberships in the set *C-shape* are greater than zero, otherwise $\mu_3(F)$ will not be evaluated according to this rule. If the appropriate features are present, the two features in F can be designated as c_1 and c_2, where c_1 is above c_2 in the symbol.

Each classification rule states which *attributes* the features in F should have and which *relationships* should exist between those features if they are to form a particular symbol. Each rule is designed to be writer independent, but the more the features in F deviate from the features in a classic version of the symbol, the lower the membership value for that symbol will be.

The requisite attributes in the rule below are that c_1 and c_2 should have the appropriate *orientations* and *lengths* for a 3, and should be of a high *quality*. The requisite relationships are that the *connectivity* between the two features should be suitable for a 3, and that the amount of *vertical overlap* for c_1 and c_2 should be minimal.

$$\mu_3(F) \leftarrow \mu_{Ori}(F) \cap \mu_{Len}(F) \cap \mu_{Qua}(F) \cap \mu_{Con}(F) \cap \mu_{Ver}(F)$$

As was the case with the feature extraction rules, the classification rules are divided into high-level and low-level rules. Rules which determine the likelihood that a feature set forms a symbol, such as the rule above, are high-level rules. Each low level rule determines the extent to which some attribute or relationship is appropriate for a particular symbol. The low-level rules for 3 are described below.

Orientation: The orientation of a *C-shape*, $orient(c_1)$, is a value between $0\,^\circ$ and $360\,^\circ$ representing the direction in which it is facing. $0\,^\circ$ is east, $90\,^\circ$ is north, $180\,^\circ$ is west and $270\,^\circ$ is south. In a 3 both c_1 and c_2 should be facing west. The further their orientations are from facing west, the lower the value $\mu_{Ori}(c_1, c_2)$.

$$\mu_{Ori}(c_1, c_2) \leftarrow \mu_{West}(c_1) \cap \mu_{West}(c_2)$$
$$\mu_{West}(c_1) \leftarrow 1 - S(\mid 180 - orient(c_1) \mid,\ 5,\ 40,\ 75)$$

Length: The relative length of a feature, $rlen(c_1)$, represents its length as a fraction of the symbol length. In a 3, ideally both c_1 and c_2 should occupy half of the symbol length. $\mu_{Len}(c_1, c_2)$ is proportional to the $rlen$ of the shorter feature.

$$\mu_{Len}(c_1, c_2) \leftarrow S(\min(rlen(c_1), rlen(c_2)),\ 0,\ 0.2,\ 0.4)$$

Quality: The membership values of c_1 and c_2 in the set *C-shape* must affect the likelihood that the symbol is a 3.

$$\mu_{Qua}(c_1, c_2) \leftarrow \mu_{Cshape}(c_1) \cap \mu_{Cshape}(c_2)$$

Connectivity: The lower endpoint of c_1 should be connected to the higher endpoint of c_2. $\mu_{Con}(c_1, c_2)$ is inversely proportional to the distance between these two points (represented by $dist(low(c_1), high(c_2))$) as a fraction of the symbol length.

$$\mu_{Con}(c_1, c_2) \leftarrow 1 - S(dist(low(c_1), high(c_2))/symbolLen,\ 0,\ 0.08,\ 0.16)$$

Vertical overlap: The more the bounding boxes of c_1 and c_2 overlap, the lower the membership value $\mu_{Ver}(c_1, c_2)$. $vov(c_1, c_2)$ is the fraction of the height of c_1 which is vertically in line with c_2.

$$\mu_{Ver}(c_1, c_2) \leftarrow 1 - S(\max(vov\,(c_1, c_2), vov\,(c_2, c_1)),\ 0.1,\ 0.4,\ 0.7)$$

Writing each classification rule requires a period of testing and a certain degree of trial and error. An alternative to creating fuzzy rules manually is automatic rule generation [10]. This method is advantageous for fuzzy systems which require vast numbers of rules. For our system classification rules could be generated according to feature extraction results produced during a training phase. However, the rules produced by an expert are likely to be superior, especially given that the subtle differences between similar symbols such as 1 and 7, or 5 and S, can be specified in the rules.

4.2. Classification Algorithm

Given the input F and a classification rule base, the algorithm must determine the most likely identity for F. This is achieved by applying rules to F and generating membership values for symbols. The result is the symbol α for which the highest membership value is produced.

Due to the following efficiency measures this process is very inexpensive computationally.

- The rules in the rule base are grouped according to the number of each type of feature they can accept (e.g. two *C-shapes* for the rule in 4.1). Only those rules in the relevant groups are applied to the input F, which is only a fraction of the rules in the rule base.
- The investigation into F being a particular symbol halts immediately when an unacceptable attribute or relationship is identified.

If the degree of confidence in the result $\mu_\alpha(F)$ is above a certain threshold t_0, the recognition process terminates and α is the result. If there is little or no confidence in the result, further investigation is done as regards the identity of the symbol.

4.3. Course of Action if no Confidence in Result

The following course of action terminates as soon as a result is found using a feature set F' such that $\mu_\alpha(F') \geq t_0$. A maximum of four feature sets are submitted for classification at each stage.

1. If other feature sets were compiled during feature extraction for which $q(F') < q(F)$, these are submitted to the classification process in order of quality.

2. This step involves a return to the feature extraction phase. The substrokes which contain sharp turning points are evaluated. A new feature type is introduced, *PC-shape*, which has the same definition as *C-shape* except that the sets SP and CC are not included. This means a *PC-shape* can contain sharp turning points and straight sections. Sets of substrokes which contain at least one *PC-shape* are submitted for classification in order of quality. A *PC-shape* can be accepted instead of a *C-shape* by the majority of the classification rules. However, $\mu_{Qua}(F')$ usually imposes a penalty in this situation.

3. All possible sets of substrokes which contain at least one embedded substroke are compiled. These sets are submitted to the classification phase in order of quality.

4. Low-quality substrokes are reintroduced to the process, and sets containing low-quality substrokes are compiled and submitted for classification.

5. It may be the case that the written symbol includes a *ligature*, an additional feature which is not an essential part of the symbol. Potential ligatures are removed from the highest quality sets of substrokes, and the new sets are submitted for classification. If ligatures are removed from F' a penalty is imposed on $\mu_\alpha(F')$ according to the lengths of the ligatures removed and their locations in the symbol.

6. If a satisfactory result has still not been found, the result is the symbol α for which the highest membership value was produced at any stage.

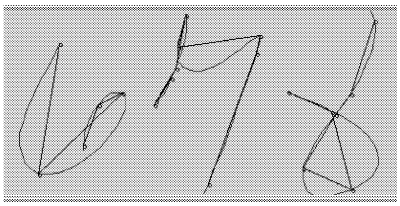

Figure 2. Examples of misclassified symbols

5. Experimental Results

A system has been implemented to recognise symbols using the approach described. Classification rules have been written for all digits and lowercase letters. Further rules will be added for uppercase letters, Greek symbols and mathematical notation. The system was tested with symbols from the UNIPEN [11] database. The files used were *aga*, *upb* and *val*, which contain a total 2465 symbols. 2324 of these were classified correctly, or **94.28%**. The recognition speed was approximately 30ms (Pentium 2.0GHz) for each symbol, which is certainly sufficient for online recognition.

For the symbols not recognised, some errors indicated that improvements to the feature extraction and classification rules were required. These rules are continually improved over time. Certain errors were due to ambiguity between similar symbols, for example certain instances of 1 and 7 were mistaken for each other. Other symbols were not recognised because they were written with an unorthodox set of features, and therefore none of the classification rules for the intended symbol matched the written version of the symbol.

To combat the latter problem, an alternative approach is being developed whereby a recurrent neural network developed by the group [12] is trained with feature sets for different symbols. Symbols drawn with highly unconventional feature sets may be recognised using this approach if similar symbols were encountered during training.

6. Conclusions

In this paper we have outlined a robust and innovative approach to recognising handwritten symbols which effectively combines feature extraction and fuzzy logic. The approach is dependent on the ability to accurately determine the likelihood that substrokes are *Lines*, *C-shapes* or *O-shapes*. This is achieved through proper selection of the properties required for each feature type, and the use of fuzzy rules which accurately assess the extent to which these properties are present.

Combining the rule-based classification approach with the recurrent neural network classifier will lead to greatly improved recognition rates. This symbol recogniser will be included as part of a larger system for the recognition of handwritten mathematical expressions.

References

[1] O. D. Trier, A. K. Jain and T. Taxt, "Feature extraction methods for character recognition - A survey," *Pattern Recognition 29, pp. 641-662*, 1996.

[2] L.A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *Man and Computer, pp. 130-165*, 1972.

[3] N.R. Gomes and Lee Luan Ling, "Feature extraction based on fuzzy set theory for handwriting recognition," *ICDAR'01, pp. 655-659*, 2001.

[4] A. Malaviya and L. Peters, "Fuzzy Feature Description of Handwriting Patterns", *Pattern Recognition, Pergamon Press, Vol. 30, No. 10, pp. 1591-1604*, 1997.

[5] I. P. Morns and S. S. Dlay, "The DSFPN: A New Neural Network and Circuit Simulation for Optical Character Recognition," *IEEE Transactions on Signal Processing, Vol. 51, No. 12, pp. 3198-3209*, 2003.

[6] I. Soraluze, C. Rodriguez, F. Boto, and A. Perez, "Multidimensional Multistage K-NN Classifiers for Handwritten Digit Recognition," *IWFHR'02, pp. 19-23*, 2002.

[7] Robert Powalka, "Algorithms for on-line cursive script recognition," *Report, The Nottingham Trent University*, 1993.

[8] L.A. Zadeh, "Calculus of Fuzzy Restrictions," *Fuzzy Sets and Their Applications to Cognitive and Decision Processes, Academic Press, NY, pp. 1-39*, 1975.

[9] A. Kaufmann, "Introduction to Theory of Fuzzy Subsets", *Academic Press, NY*, 1975.

[10] F. Ivancic, A. Malaviya and L. Peters, "An automatic rule base generation method for fuzzy pattern recognition with multiphased clustering", *KES'98, Vol. 3, pp. 66-75*, 1998.

[11] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet, "UNIPEN project of on-line data exchange and recognizer benchmarks," *12th ICPR, pp. 29-33*, 1994.

[12] B.Q. Huang, T. Rashid, and T. Kechadi, "A New Modified Network based on the Elman Network," *The IASTED International Conference on Artificial Intelligence and Applications (AIA 2004)*, Feb. 16-18, 2004.