A Fuzzy Based Feature Extraction Approach for Handwritten Characters

Mahmood K Jasim [1], Anwar M Al-Saleh [2] and Alaa Aljanaby [1]

[1] **Department of Mathematical & Physical Sciences, College of Arts & Sciences, University of Nizwa, Oman**

[2] **Computer Science Department, College of Sciences, Al-Mustansiriyah University, Iraq**

Abstract

This paper describes a technique that can be used to generate fuzzy rules to extract the features of handwritten characters. The feature extraction is a complicated problem as different people write the same character in different ways. The development of a technique that can generate the description of handwritten characters is still a challenge for the handwritten recognition systems. The fuzzy logic offers a good opportunity to build a rule-based feature extraction technique for handwritten characters with low computational cost.

Keywords: *Feature Extraction, Handwritten Characters, Fuzzy Logic.*

1. Introduction

The term feature extraction consists of two meanings; feature detection, and feature selection [1]. The purpose of feature detection is to obtain those features, which preserve the useful information about the image to the largest extent. The aim of feature selection is to determine those principal feature components depending on a certain classification task in order to achieve an effective classification [1, 2]. The above idea shows that the output of feature detector reflects the information of the image. Feature extraction is responsible for extracting all possible features that are expected to be effective in diagnosing all information of image, without concerning the disadvantages of excessive dimensionality [3].

The feature after selection may not contain enough information about the original image, but it must contain the information that is useful to distinguish different classes for image classification [2, 3]. Figure 1 shows a block diagram of feature extractor for classification system.

In the Handwritten Recognition systems, many tedious tasks can be made more efficient by automating the process of reading handwritten numerals. In such system an optical scanner converts each handwritten numeral to a digital image, and computer software classifies the image as one of the digits zero through nine. By reducing the need for human interaction, numeral-recognition systems can speed up jobs such as reading income tax returns, sorting inventory, and routing mail. Several steps are necessary to achieve this. A recognition system must first capture digital image of handwritten numerals. Before attempting to classify the numerals, some preprocessing image might be necessary. An algorithm must then classify each handwritten numeral as one of the ten decimal digits [4, 5].

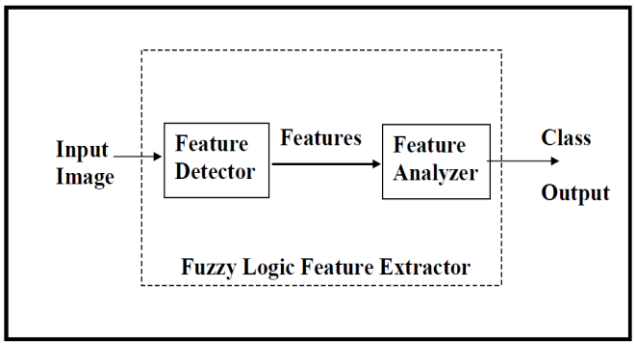

Fig. 1: Block Diagram of Feature Extractor

Although a qualitative description of this process is straight forward, it cannot be easily reduced to a few simple mathematical rules. The difficulty results from the natural variations in human handwritten. A useful recognition system must be robust to alterations in size, shape, orientation, thickness, etc. Closed-form mathematical models tend to be inadequate for such a task because of the many possible representations of the same image. The problem presents certain obstacles that make pattern matching on a pixel–by–pixel basis impractical. For instance, the edge of a character segment can show up in two or more data slices (all the pixels along one column), depending on where the slices overlap. Further, slight variations in printing cause character height and width to vary and misfeeding of the document can skew the imaged character.

Copyright (c) 2013 International Journal of Computer Science Issues. All Rights Reserved.

Fuzzy logic, which is inherently superior for processing imprecise data, is a natural for this application [1, 7]. However, a data preprocessor is necessary to simplify the problem so that it can be easily described in fuzzy rules. Feature extraction is the crucial phase in numeral identification as each numeral is unique in its own way, thus distinguishing itself from other numerals. Hence, it is very important to extract features in such a way that the recognition of different numerals becomes easier on the basis of the individual features of each numeral [8, 9].

In the present paper, the authors propose the transition calculation and sum of pixels of an image as a feature detector, fuzzy logic technique as a feature analyzer, and extraction the most useful information as outputs for classifier process. Handwritten numerals recognition system has been designed and implemented and a high degree of accuracy has been gained using fuzzy logic.

2. Feature Extraction

Feature extraction involves simplifying the amount of resources required to describe a large set of data accurately. Feature extraction is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy. The literature is replete with high accuracy recognition systems for separated handwritten numerals and characters [10]. However, research into the recognition of characters extracted from cursive and touching handwriting has not had the same measure of success [11]. One of the main problems faced when dealing with segmented, handwritten character recognition is the ambiguity and illegibility of the characters. Figure 2 illustrates the difficulties a programmer encounter when trying to match incoming patterns against an idealized pattern, or template. Each of the three sections of Figure 2 shows twenty data slices of typical read of the character 0.

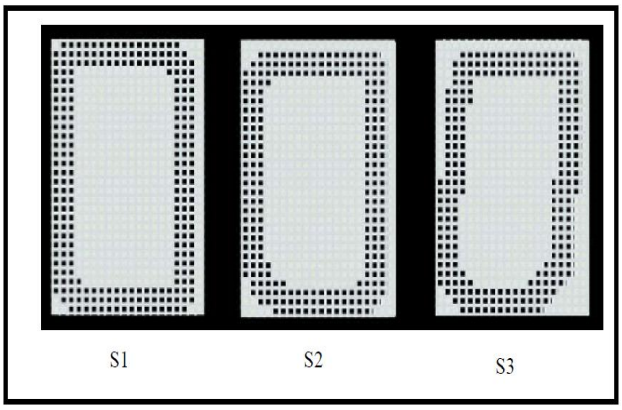

Fig. 2: Different Patterns of Character Zero

The leftmost portion of Figure 2 (S1) represents the pattern associated with an ideal read of a character 0. This portion of the figure can be considered to be a template for the read of a character 0. The center and the right portions of Figure 2 (S2, S3) show some patterns of character 0.

One approach to recognition would have a program compare scanned characters to templates on a pixel-by-pixel basis. Clearly, this procedure could often fail (in this case). For instance, the program would expect a 1 in slice 1, local 3 of a character 0 in pattern S1, and neither S2 nor S3 characters would satisfy the expectation.

Another approach would have the program sum all the pixels in each slice and compare the resulting slice totals to corresponding slice totals from templates. As shown in table 1 below, this approach also cannot produce a match in both S2 and S3 case.

Table 1: Slice Totals for the Three Readings of Fig. 1

Slice	S1	S2	S3
1	27	23	12
2	29	26	23
3	30	28	27
4	7	9	21
5	6	7	10
6	6	7	6
7	6	6	6
8	6	6	6
9	6	6	6
10	6	6	6
11	6	6	6
12	6	6	6
13	6	6	7
14	6	6	8
15	6	6	8
16	7	7	16
17	30	29	26
18	29	28	25
19	27	26	15
20	0	0	0

The data in Table 1 provides a useful insight. It is apparent in all three cases that the magnitude of the slice total increases to a high value of approximately 30, decreases to a low value of approximately 6, increases again to a high value of approximately 30, and then finally decreases to zero. It is possible to locates and quantifies these transitions, or changes of magnitude, in slice totals. Quantified transitions will form the input to the handwritten recognition fuzzy system. The fuzzy rules will look something like this: A very large positive transition, followed by a large negative transition, followed by a large positive transition, followed by a very large negative transition, indicates a character zero. A transition is defined as the difference between a current local maximum (or minimum) and the previous local minimum (or maximum). The data preprocessor takes a data slice,

Copyright (c) 2013 International Journal of Computer Science Issues. All Rights Reserved.

obtains its slice total, and compares the magnitude of the slice total to previous slice total to determine whether it constitutes a new local maximum or minimum.

3. The Transition Concept

Figure 3 below shows the block diagram of the preprocessor for Transition Calculation algorithm which takes in slice data and generates transition outputs. The variables that are updated during preprocessor operation are listed in Figure 3. Preprocessor outputs take the form of a transition number and an associated transition magnitude. For instance, T1=27 means transition number 1 has a magnitude of 27. The algorithm incorporates hysteresis in determining a direction change. In other words, a transition must be of three pixels or greater magnitude to be recognized. For instance, if a current reading produce a slice total of 6 and the previous reading left DIR as (–) and L-min as 4, the current reading would fail the test CR > PR+2, but would pass the test CR > PR. Since DIR-, none of the variables are changed. The preprocessor algorithm has no effect on system throughput because it can be run during the delay for integration time.

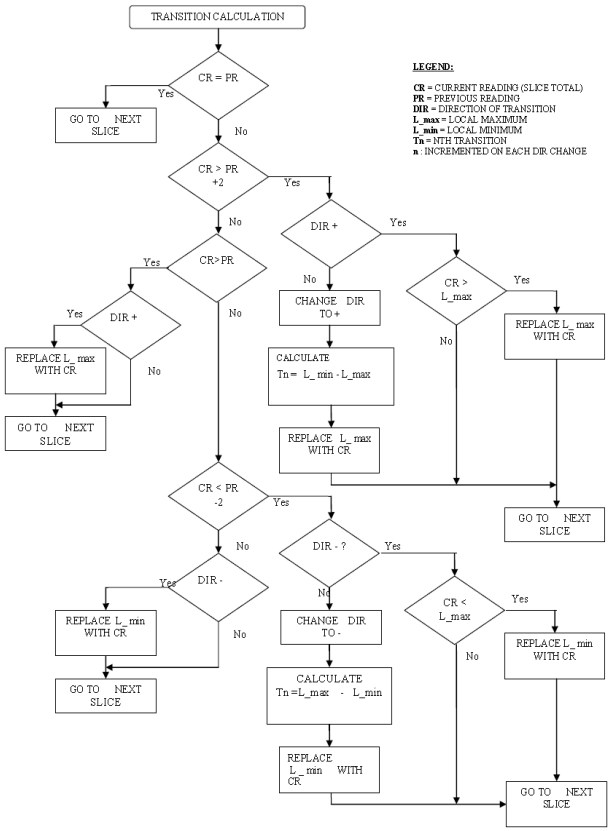

Fig. 3: The Block Diagram of the Preprocessor for Transition Calculation Algorithm

Table 2 shows how variables are updated after each slice. The slice data applied is from the S3 case shown in Figure 2. Prior to entering the routine Transition Calculation, variables are initialized to the values shown in the column labeled Init. Data slice #1 is defined as the first slice with a slice total greater than 2. A final transition number magnitude calculation is forced after the 22nd slice.

Table 2: Translation Calculation for S3 Character Scan

Slice	CR	PR	DIR	T	L_min	L_max	Tmag.
Init	0	0	+		0	0	0
1	12	0	+		0	12	0
2	23	12	+		0	23	0
3	27	23	+		0	27	0
4	21	27	-	T1	21	27	27
5	10	21	-		10	27	0
6	6	10	-		6	27	0
7	6	6	-		6	27	0
8	6	6	-		6	27	0
9	6	6	-		6	27	0
10	6	6	-		6	27	0
11	6	6	-		6	27	0
12	6	6	-		6	27	0
13	7	6	-		6	27	0
14	8	7	-		6	27	0
15	8	8	-		6	27	0
16	16	8	+	T2	6	16	-21
17	26	16	+		6	26	0
18	25	26	+		6	26	0
19	15	25	-	T3	15	26	20
20	0	15	-		0	26	0
21	0	0	-		0	26	0
22	0	0	-	T4	0	26	-26

The preprocessor found the following four transitions while reading this character. T1=27, T2=-21, T3=20, T4=-26. For Comparison, the preprocessor would return the following values for both S1 and S2 character 0 (respectively): (S1: T1=30, T2=-24, T3=24, T4=-30) and (S2: T1=28, T2=-22, T3=23, T4=-29). The visual representation of Table 2 can be represented by the following figure 4 below.

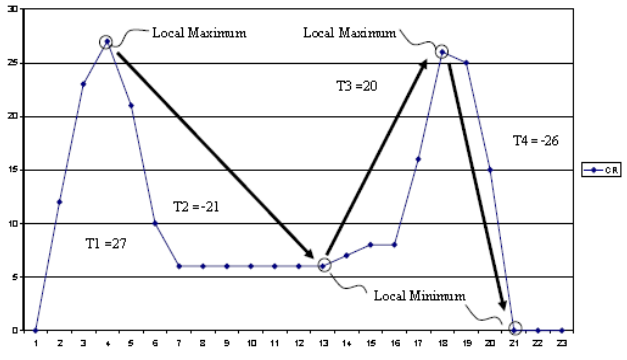

Fig. 4: Visual Representation of Table 2 Transition

Copyright (c) 2013 International Journal of Computer Science Issues. All Rights Reserved.

4. Fuzzifying Transition Inputs

The transition visualized in figure 3 above make it very easy to write a fuzzy rule that recognizes a character 0: *If T1 is Very_Large_Positive and T2 is Large_Negative and T3 is Large_ Positive and T4 is Very_ Large_ Negative, then character is 0.* A similar visualization of all ten characters is required to write the remaining rules. Table 3 below represents the range of transition magnitudes for all ten characters. This data was obtained by unconstrained images represented handwritten numeral. The number of transitions per character varies from two (characters 1, 3, and 7) to four (characters 2, 4, 5, 6, 8, 9, and 0).

The first step in fuzzifying this data is to establish a universe of discourse that defines the range of possible values for fuzzy input. Once the universe of discourse is defined, fuzzy sets can be created within it. In this case, T1, T2, T3, and T4 are fuzzy inputs. From Table 3 below, transition magnitudes, measured in pixels, vary from -28 to +28, since a slightly over sized character or stray marks on the document can cause more pixels to be counted, the universe of discourse is represented by the range of value from –30 to +30. There are actually two universes of discourse: a positive one associated with T1 and T3, and a negative one associated with T2 and T4. The positive universe of discourse is defined as the range of values from 5 to 28 pixels, and the negative universe of discourse is defined as the range of values from –3 to –28 pixels.

Table 3: Transition Magnitude Ranges for All Ten Characters

Char	T1 (LO,HI)		T2 (LO,HI)		T3 (LO,HI)		T4 (LO,HI)	
0	21	25	-14	-20	14	20	-21	-25
1	13	13	-13	-13	---	---	---	---
2	17	19	-6	-9	5	10	-17	-20
3	23	27	-23	-27	---	---	---	---
4	15	19	-15	-19	12	16	-15	-19
5	16	19	-6	-10	5	9	-14	-20
6	21	24	-12	-16	6	8	-12	-17
7	25	28	-25	-28	---	---	---	---
8	20	22	-7	-11	13	17	-23	-27
9	13	16	-3	-8	17	21	-23	-28

The distribution of transition values across the universe discourse label denotes each character and transition number, putting in graphical representation of transition range, from Table 3 above.

It is obvious from Figure 5 that there is some clumping of transition data could be used to create fuzzy sets that apply all four transition inputs. Based on the natural grouping of data, fuzzy sets are assigned and labeled. At this point, transition inputs and fuzzy sets are defined.

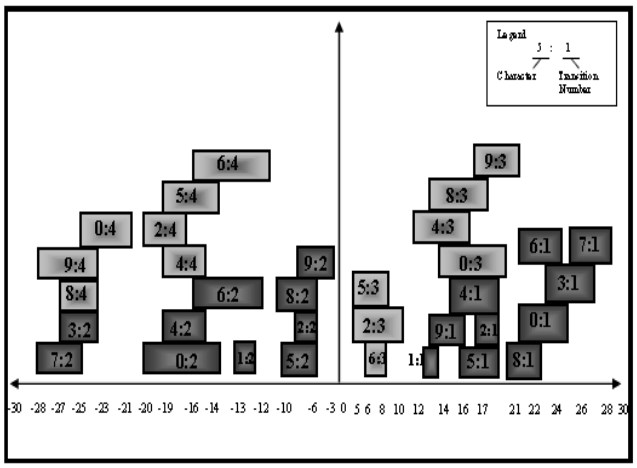

Fig. 5: Distribution of Transitions across the Universe of Discourse

Figures 6, 7, 8 and 9 show the fuzzy set distribution for T1, T2, T3 and T4. Unfortunately, transition inputs alone are not adequate to classify characters. Consider the characters 3 and 7. Each has only two transitions, T1 and T2. The rule for character 3 is: if T1 is Large and T2 is Small then character is 3. The rule for character 7 is: if T1 is large and T2 is Small then character is 7. Thus, from transition data, 3 and 7 are indistinguishable.

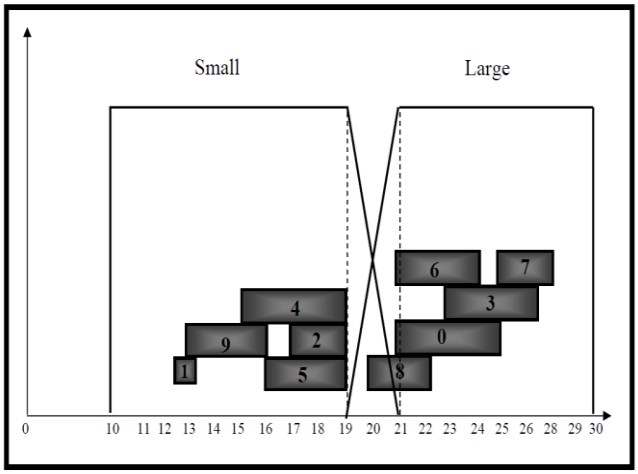

Fig. 6: T1 Fuzzy Set Distribution

Copyright (c) 2013 International Journal of Computer Science Issues. All Rights Reserved.

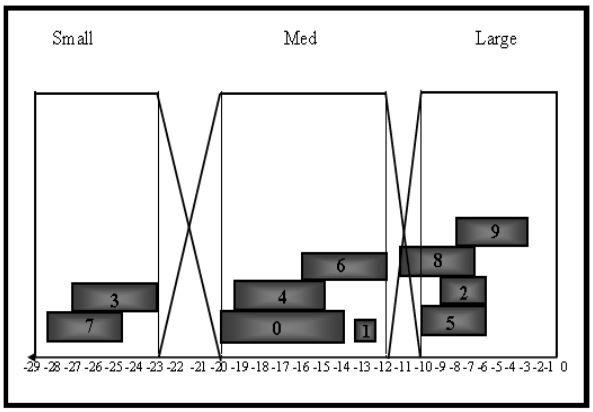

Fig. 7: T2 Fuzzy Set Distribution

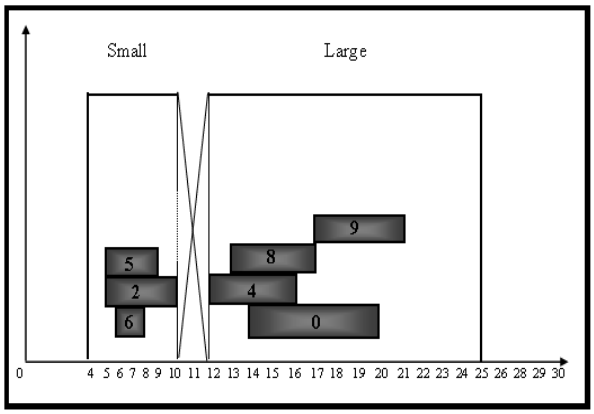

Fig. 8: T3 Fuzzy Set Distribution

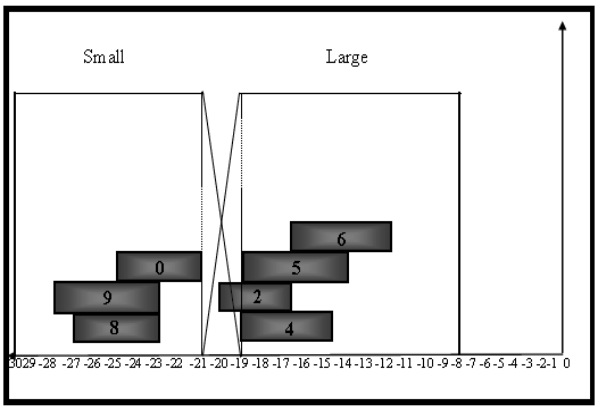

Fig. 9: T4 Fuzzy Set Distribution

Table 4 shows the fuzzy magnitude of each transition presented by character. The conflict column shows the other characters which present the same fuzzy magnitude. Several conflicts occur, so it is necessary to introduce input variables in addition to transition magnitude.

Table 4: Transition Magnitudes for Each Character

Char	T1	T2	T3	T4	Conflict
0	Large	Med	Large	Small	---
1	Small	Med	---	---	4
2	Small	Large	Small	Large	5
3	Large	Small	---	---	7
4	Small	Med	Large	Large	1
5	Small	Large	Small	Large	2
6	Large	Med	Small	Large	---
7	Large	Small	---	---	3
8	Large	Large	Large	Small	---
9	Small	Large	Large	Small	---

5. Sums Of Pixels (SOP)

Some of the conflicts shown in Table 4 can be resolved by considering the total dark area in each character. Total dark area is measured as a sum of pixels of the image, or SOP. The image is divided into four equal quarters and computes the sum of pixels for each quarter. The sub image 1 is the sub image in the top left of the original image (character image), and its size is (15×10), as shown in Figure 10 bellow.

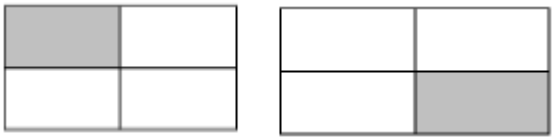

Fig. 10: SOP1 and SOP2

SOP1, represent sum of pixels of sub image 1, will be the fifth fuzz input. Table 5 shows the range of SOP1 values for each character.

Table 5: SOP1 Range for Each Character

Char.	0	1	2	3	4	5	6	7	8	9
SOP1 (low)	56	12	47	50	42	58	53	27	73	47
SOP1 (high)	61	12	65	56	50	84	68	30	80	70
SOP1 (avg.)	59	12	56	53	53	71	61	29	77	59

Figure 11 shows how the conflicts shown in Table 4 are resolved by the addition of the SOP1 input variable. Consider the conflict between characters 1 and 4. The value of SOP1 for the character 1 is always produces a degree of membership of 1 in the fuzzy set SOP1 Small and a degree of membership of 0 in the fuzzy set SOP1 Large, so that a 1 is never recognized as a 4. Conversely, as long as the possible range of SOP1 values for 4 always produces a higher degree of membership in the fuzzy set Large than in the fuzzy set Small, a 4 is recognized as a 4

Copyright (c) 2013 International Journal of Computer Science Issues. All Rights Reserved.

rather than as a 1. SOP1 for character 4 ranges from 42 to 50 (Table 5). Figure 11 shows that any value of SOP1 from 42 to 50 produces a degree of membership of 1 in the fuzzy set SOP1 Large. The conflict is completely resolved. Each character appears exclusively within a fuzzy set with no overlap.

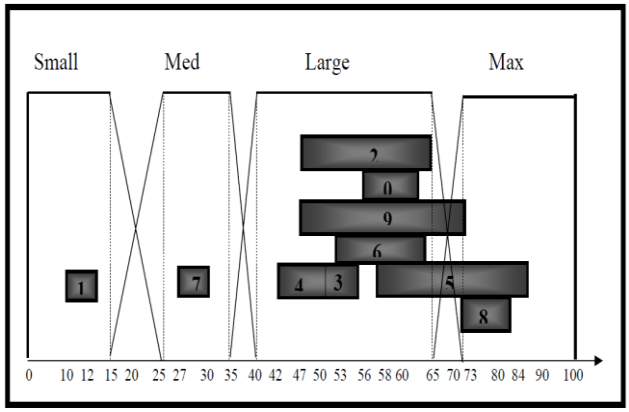

Fig. 11: SOP1 Fuzzy Set Distribution

The input variable SOP1 also resolved the conflict between the two characters 3 and 7. As long as the possible range of SOP1 values for 3 always produces a higher degree of membership in the fuzzy set Large than in the fuzzy set Med, a 3 is recognized as a 3 rather than as a 7. SOP1 for character 3 ranges from 50 to 56 (Table 5). Figure 11 shows that any value of SOP1 from 50 to 56 produces a degree of membership of 1 in the fuzzy set SOP1 Large. Values from 40 to 35 produce declining degrees of membership. Since the minimum value of SOP1 from character 3 is 50, the character 3 always produces some degree of membership in the fuzzy set SOP1 Large and none in the fuzzy set SOP1 Med. Therefore, a 3 is never recognized as a 7. Conversely, the range of SOP1 values for the character 7 is always produces a degree of membership of 1 in the fuzzy set SOP1 Med and a degree of membership of 0 in the fuzzy set SOP1 Large, so that a 7 is never recognized as a 3. The conflict is completely resolved between characters 3 and 7. Each character appears exclusively within a fuzzy set with no overlap.

Notice that the lowest value of SOP1 for 5 produces a higher degree of association with Large than with Max, preventing a resultant 5 for low values of SOP1. Then because of the interaction of its SOP1 ranges with multiple fuzzy sets, SOP1 should not be used as a qualifier for character 5. So the range of SOP1 values for character 5 is not adequately represented by either SOP1 Large or Max. In this case, it is best to leave the SOP1 input variable out of the fuzzy equation for character 5. Therefore an additional input variable is required. SOP1 is only resolving the conflict between characters 1 and 4 and between characters 3 and 7. However, it makes sense to assign the remaining characters to fuzzy sets in the SOP1 universe of discourse. The minimal additional code that required for these rule additions produces better-qualified results and a more robust classification system.

An additional characteristic that may resolve the remaining conflicts is by considering the sum of pixels in each character sub image 2, or SOP2, where sub image 2 is a sub image of character image with Size (15 x 10) as shown in Figure 6. Table 6 shows the range of SOP2 values for each character. Figure 7 shows the universe of discourse and fuzzy sets for SOP2.

Table 6: SOP2 Range for Each Character

Char	0	1	2	3	4	5	6	7	8	9
SOP2 (low)	59	14	61	18	45	32	78	0	72	30
SOP2 (high)	63	14	70	30	67	52	90	5	82	42
SOP2 (avg.)	61	14	66	24	56	42	84	3	77	36

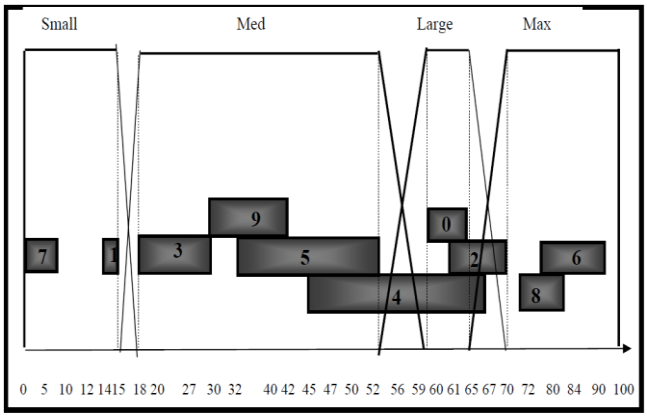

Fig. 12: SOP2 Fuzzy Set Distribution

Figure 12 shows how the conflict between characters 2 and 5 is resolved by the addition of the SOP2 input variable. As long as the possible range of SOP2 values for 2 always produces a degree of membership in the fuzzy set Large than in the fuzzy set Med, a 2 is recognized as a 2 rather than as a 5. Conversely, the range of SOP2 values for the character 5 is always produces a degree of membership of 1 in the fuzzy set SOP2 Med and a degree of membership of 0 in the fuzzy set SOP2 Large, so that a 5 is never recognized as a 2. The conflict is completely resolved

Copyright (c) 2013 International Journal of Computer Science Issues. All Rights Reserved.

between characters 2 and 5. Each character appears exclusively within a fuzzy set with no overlap. Fuzzy rules create associations between specific inputs and desired outputs. Here, the six input variables are T1, T2, T3, T4, SOP1 and SOP2. Ten valid output variables, one for each of the ten characters to be recognized, are defined. Table 7 shows the relationship between fuzzy input variables and fuzzy output variables.

Table 7: Fuzzy Set Association

Char	T1	T2	T3	T4	SOP1	SOP2
0	Large	Med	Large	Small	Large	Large
1	Small	Med	-------	-------	Small	Small
2	Small	Large	Small	Large	Large	Large
3	Large	Small	------	-------	Large	Med
4	Small	Med	Large	Large	Large	-------
5	Small	Large	Small	Large	-------	Med
6	Large	Med	Small	Large	Large	Max
7	Large	Small	------	-------	Med	Small
8	Large	Large	Large	Small	Max	Max
9	Small	Large	Large	Small	Large	Med

With the relationship information summarized in Table 7, the fuzzy rules can be written as:

Rule 1: if T1 is Large and T2 is Med and T3 is Large and T4 is Small and SOP1 is Large and SOP2 is Large **then** Character is *Zero*.

Rule 2: if T1 is Small and T2 is Med and SOP1 is Small and SOP2 is Small **then** Character is *One*.

Rule 3: if T1 is Small and T2 is Large and T3 is Small and T4 is Large and SOP1 is Large and SOP2 is Large **then** Character is *Two*.

Rule 4: if T1 is Large and T2 is Small SOP1 is Large SOP2 is Med **then** Character is *Three*.

Rule 5: if T1 is Small and T2 is Med and T3 is Large and T4 is Large SOP1 is Large **then** Character is *Four*.

Rule 6: if T1 is Small and T2 is Large and T3 is Small and T4 is Large and SOP2 is Med **then** Character is Five.

Rule 7: if T1 is Large and T2 is Med and T3 is Small and T4 is Large and SOP1 is Large and SOP2 is Max **then** Character is Six.

Rule 8: if T1 is Large and T2 is Smell and SOP1 is Med and SOP2 is Small **then** Character is Seven.

Rule 9: if T1 is Large and T2 is Large and T3 is Large and T4 is Small and SOP1 is Max and SOP2 is Max **then** Character is Eight.

Rule 10: if T1 is Small and T2 is Large and T3 is Large and T4 is Small and SOP1 is Large and SOP2 is Med **then** Character is Nine.

6. Conclusion

The method presented in this paper is a reliable and relatively simple method for generating the fuzzy rule-based description of handwritten characters. Even though this method is not deemed to be the ultimate solution for the recognition of handwritten characters but it is a solution which is extremely simple to implement and use. Testing results will be considered in a forthcoming paper incorporating the presented technique in a complete online handwritten recognition system.

References

[1] Bandara G.E, Pathirana S.D., Ranawana R. M. "Use of Fuzzy Feature Descriptions to Recognize Handwritten Alphanumeric Characters", 1st Conference on Fuzzy Systems and Knowledge Discovery, Singapore, November 2002.

[2] Belal K. Elfarra and Ibrahim S. I. Abuhaiba. "New Feature Extraction Method for Mammogram Computer Aided Diagnosis", International Journal of Signal Processing, Image Processing and Pattern Recognition, Vol. 6, No. 1, February, 2013.

[3] Rajashekararadhya, S. and Ranjan, P. "Zone based feature extraction algorithm for handwritten numeral recognition of kannada script", In Advance Computing Conference, 2009. IACC 2009. IEEE International, pages 525–528, 2009.

[4] L. M. Lorigo and V. Govindaraju, "Offline Arabic handwriting recognition: a survey", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 5, pp. 712 - 724, 2006.

[5] Bandara G.E., Pathirana S.D., Ranawana R. M., "A Short Method for On-line Alphanumeric Character Recognition", NAFIPS – FLINT 2002, New Orleans, USA, June 2002.

[6] Pal, U., Sharma, N., Wakabayashi, T., and Kimura, F. "Handwritten character recognition of popular south indian scripts", In Doermann, D. and Jaeger, S., editors, Arabic and Chinese Handwriting Recognition, volume 4768 of Lecture Notes in Computer Science, pages 251–264, Springer Berlin / Heidelberg, 2008.

Copyright (c) 2013 International Journal of Computer Science Issues. All Rights Reserved.

[7] Jan J. "Tutorial on fuzzy logic", Technical University of Denmark of Automation, Report No. 98-E868, 19 Aug. 1998.

[8] Yadana T. and San S. Y. "High Accuracy Myanmar Handwritten Character Recognition using Hybrid approach through MICR and Neural Network", International Journal of Computer Science Issues (IJCSI), Vol. 7, Issue 6, November 2010.

[9] Muthumani .I , Uma Kumari C.R. "Online Character Recognition of Handwritten Cursive Script", International Journal of Computer Science Issues (IJCSI), Vol. 9, Issue 3, No 2, May 2012.

[10] Lauer, F., Suen, C. Y., and Bloch, G. "A trainable feature extractor for handwritten digit recognition", Pattern Recognition, 40(6):1816–1824, 2007.

[11] Pal, U., Sharma, N., Wakabayashi, T., and Kimura, F. "Handwritten character recognition of popular south indian scripts", In Doermann, D. and Jaeger, S., editors, Arabic and Chinese Handwriting Recognition, volume 4768 of Lecture Notes in Computer Science, pages 251–264, Springer Berlin / Heidelberg, 2008.

Mahmood K Jasim is an Associate Professor and head of the Department of Mathematical and Physical Sciences, College of Arts and Sciences, University of Nizwa, Oman. His research interests are Mathematical Modeling, Mathematical Physics, Differential Equations and its application, Numerical Analysis, Artificial Neural Networks and Fuzzy logic.

Anwar M Al-Saleh is a Lecturer at Computer Science Department, College of Sciences, Al-Mustansiriyah University, Baghdad, IRAQ. Her research interests are Fuzzy Logic and Image processing.

Alaa Aljanaby is an IEEE member, Assistant Professor and head of the Computer Science Section, Department of Mathematical and Physical Sciences, College of Arts and Sciences, University of Nizwa, Oman. His research interests are soft computing, bio-inspired optimization algorithms, combinatorial optimization problems, persuasive technology, opinion mining and image processing.

Copyright (c) 2013 International Journal of Computer Science Issues. All Rights Reserved.