Multifonts Numeral Recognition using Hybrid Technique

Mamoun Suleiman Al Rababaa
Computer Science Department,
Information Technology Faculty
Al al-Bayt University, Jordan

Mohammad Krayyem Al.bakkar
Computer Science Department,
Information Technology Faculty
Al al-Bayt University, Jordan

ABSTRACT
This paper presents two methods for enhancing recognition rate for Arabic typewritten digits. The first is node method that computes number of terminal nodes, and the second is right side method that studies the shape from the right side. These methods can recognize multi-font digits using two stages; each method produces specific results and then compares these results to obtain the final output.

The recognition of multi-font typewritten is essential. It is a bit complicated to process, due to the difference in shape and size of the same digit. Therefore, the researcher used two methods for recognizing multi-fonts. The recognition system contains several steps, image preprocessing, which includes converting into binary, cropping the digit in single image with resize to 32 x 42 pixel, and thinning the shape of the digit to get the skeleton of the digit. Feature extraction includes number of terminal nodes from nodes method and two characters to specify the curve of right side of the shape. The recognition includes logical comparison between two vectors, one from each method.

The proposed technique was implemented and tested The experimental results showed that the proposed technique is efficient for recognizing typewritten digits. The results proved that the techniques work properly and are able to give recognition rate for 11 fonts up to 100%, or less for other fonts due to irregularity for some fonts or failing for one of two methods. The dataset contains multi-size and multi-fonts for the digits from 0 to 9.

Keywords
Numeral Recognition, typewritten digits, nodes method, right side method

1. INTRODUCTION
For time immemorial, the magic numbers have enchanted researchers and scientists across the globe. The applications of numbers are so varied that there is no point in discussing them further [1]. The recognition of machine-printed numerals has been the subject of much attention in pattern recognition, because the number of applications, such as devices, use English text recognition, such as scanner, hand-held devices, smart devices, mobiles and useful for textual reader programs; these programs that convert English text into sound, and useful for chatting and text messages.

Character recognition is the process of converting the language written in a spatial form into computer understandable form (Unicode) [2]. Digits recognition has become very popular and useful in many fields [3]. Many techniques of recognition for both machine-printed and hand-printed characters are available [4, 5, 6].

The difficulty of recognition-typing digits is that various fonts existing today, which make the feature extraction relatively hard, as if more font options are considered. In this study, the dataset for testing contains more than 176 different fonts. The performance of character/digit recognition mainly depends on the feature extraction methods and the classifier used for labeling the digits [7] [8].

This paper aims to enhance recognition digit methods, using two proposed methods, nodes method and right side method. However, the form of fonts for typewritten numbers is different and has various size, shape and fonts. So the objectives of this study are: Build a software system for identification of the typewritten digits that converts the digits (from 0 to 9) from spatial shape to Unicode form, enhance recognition rates for large dataset of digits, enhance recognition rates for large dataset of digits, use a proposition way to perform feature extraction from the shape of digits, recognize many of fonts without complex and lengthy computation and recognize some fonts of several sizes.

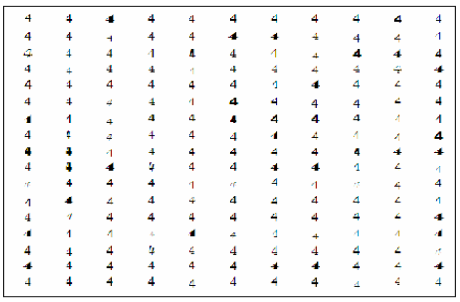
2. RELATED WORKS
Several researches have been done on digits recognition. The most variation between researches is in the way of feature extraction, because it's a critical stage in any recognition system.

M. Hanmandlu M.Hafizuddin and V.K. Madasu [1], Binary image of character is partitioned into fixed number of sub-images called boxes. The features consist of normalized vector distance and angle from each box. SOM algorithms were used in the learning phase to produce prototype, which together with variances, are used to determine fuzzy regions and membership functions. Liying Zheng [7], Four edges of an Arabic character image are adopted for character recognition, edge extraction for left, upper, right and lower edge, respectively. Sequence of computation to edge representation and feature extraction were adopted, the features extracted from edges are four types; number of lines in each edge, number of straight lines in lower edge, ratio of number of straight lines in lower edge (and right edge) and the character width (and height), and the length of longest straight line in upper edge. H. Ebrahimnezhad, GH. A. Montazer and N. Jafari [8], Proposed a combined method for recognition of multi-font Persian numeral characters. Binary image of character is devised into fixed number of sub-images called boxes. The average vector distance and angle of each box are computed as features. The features have some variations in different fonts of any character. Therefore, employed the fuzzy sets to face with recognition problem. Used exponential fuzzification involving two extra parameters, which take account of the variations in the fuzzy sets. The parameters are obtained by minimizing the entropy of fuzzy membership function. Zheru Chi H.Yan [9], Described the shape of numerals through a set of junction points, tips (end points),

and segments between two points. Twelve types of segments, used to describe the digits, four features for were used to describe a segment, they are type, the normalized length, the normalized horizontal and vertical coordinates of the segments center, then produce fuzzy rules based on ID3 approach and to optimize defuzzification parameters by using two layer perceptron. Pyeoung Kee Kim [10], Grouping confusing numerals into confusion groups and build fuzzy functions applying human knowledge. Confusing features are represented and used for classification using fuzzy logic, and representing numerals as sequence of primitive strokes and features points was very efficient to absorb many variations. Structural and statistical methods are combined to compensate other weaknesses. N. Belkhamza, A. Akkouche [11], Proposed a methodology in designing character recognition system using fuzzy logic technique. Used vertical slice for each digit with distribution of the bits number to produce input variables (called transitions); the transitions are important input to fuzzy logic system. The membership functions are derived from the statistical study, and there are 14 rules for recognition. B.V.Dhandra, V.S.Malemath, H.Mallikarjun, and H.Mallikarjun [12], proposed a novel method for Multi-font numeral recognition which is a thinning free approach. The minimum rectangle Bounding box is fitted over isolated numeral image and image is cropped. The outer densities of pixels for each of the direction are computed in four directions viz. bottom, top, left and right. The ratios of these densities are taken with the total area of the cropped numeral image and are stored in a feature vector. Decision tree based minimum distance nearest neighbor classifier is used to classify the numerals, and N.S.Arjun, G.Navaneetha, G.V.Preethi, T.K.Babu [13], Proposed a method which recognizes 17 multi- fonts of different sizes varying from size 8 to 72. The method requires less computation time for recognizing a numeral while maintaining the high amount of accuracy. This method used Euler number of a numeral, to initially characterize the numbers into different groups. Then used individual distinct features of each numeral for recognizing it.

3. PROPOSED TECHNIQUE

In this section, introduced the methodology of typewritten digits recognition technique, which will be used two proposed methods which lie in feature extraction step. The digits recognition is described in three steps, each step is essential and inevitable. The diagram and flowchart of this methodology are shown in Figure.1 and Figure.2, respectively



Preprocessing:
1. **RGB to Binary**
2. **Edge detection**
3. **Dilation & Fill**
4. **Blobs analysis**
5. **Cropping & Resize**

Feature Extraction:

1. **Nodes method**
2. **right side method**

Classification And Recognition

Fig (1): Digits recognition steps.

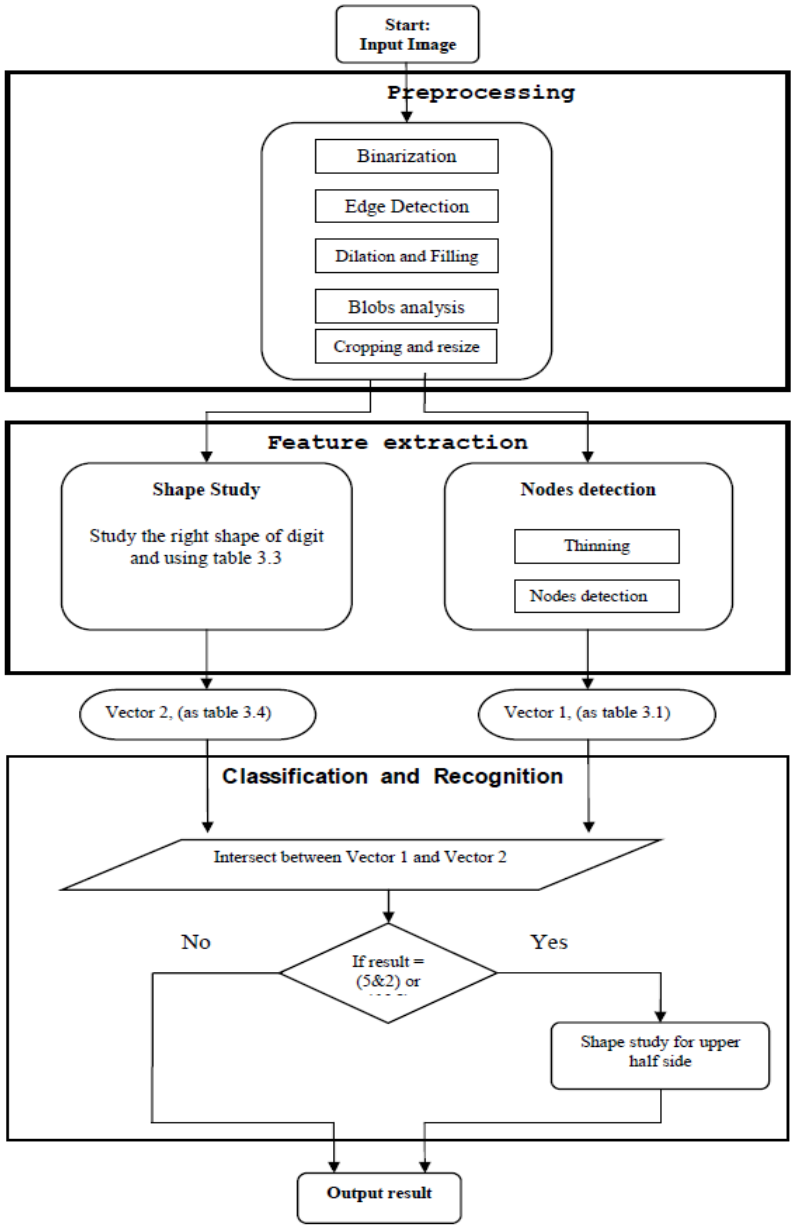


Fig (2): Flowchart for the methodology of recognition digits.

For examination the capability for recognition in this study, studied 176 types of fonts for all digits from 0 to 9, such as, each image have size 420x320 from type

.bmp, and contains 176 digits, each digit describes specific different font type from 176 fonts.

3.1 Image Preprocessing [15, 16, 17]

After acquiring the image according to set of characteristic: size 420x320, type .bmp, contain set of digits as columns and rows, the space between each row is one line, and the digits written in multi fonts until 176 font types. Each digit in the image represents one font type, in order to determine the font types that can be recognized from this study. This manner is follow for numbers from 0 to 9. Notable, the image must be void from noise, where if found image that have noise must be filtering and then use it in this system.

After that, will be apply a set of operations on the image that aims to enhance the shapes of digits, and to extract each digit in an independent image, such as each image contains isolated digit. There are a set of morphological operations that will be applied on the input image to produce the final expected images:- Binarization, Edge detection, Dilation and Filling the pixels, Blobs Analysis and Cropping and Resizing.

3.2 Feature extraction

Feature extraction is the important phase in numeral identification as each numeral is unique in its own way, thus distinguishing itself from other numerals. Hence, it's very important to extract features in such a way that the recognition of different numerals becomes easier on the basis of the individual feature of each numeral. Feature extraction stage is considered with two proposed methods, each method produces a set of results (suggested digits) the correct output digit is one of it. Then compare these two vectors of results to get the correct digit, where the intersection between two vectors occur in one element only, except two duplicated states between 2,5 and 6,8. The duplicated states will treat with other technique. The advantages of these methods stem from flexibility to recognize several sizes of fonts, recognize many fonts with well recognition rate, and not needed to complex operation and calculations.

3.2.1 Nodes method

This method will compute the number of nodes for each digit. Node (or tip) is a pixel with one or at most two neighbors in 8-connected neighborhood. Through the experiments we reached to recognize or analyze the digits by using the number of nodes for digits. We will use the number of nodes in the next method to recognize the input image. The input image is sub-image with size 42x32. Before studying the shape to get the nodes, we have to thin the shape of input digit. Thinning is widely used technique in the pre-processing stage of pattern recognition system to compress data and to enhance feature extraction in the subsequent stage, it reduces digitized pattern to a skeleton, so that all resulting branches are 1 pixel thick. The thinning algorithm used in this study [14] working as follows:

1. Divide the image into two distinct subfields in a checkerboard pattern.
2. In the first subiteration, delete pixel p from the first subfield, if and only if, the conditions G1, G2, and G3 are all satisfied.
3. In the second subiteration, delete pixel p from the second subfield, if and only if, the conditions G1, G2, and G3' are all satisfied.

Condition G1:

$$X_H(p) = 1$$

Where

$$X_H(p) = \sum_{i=1}^{4} b_i$$

$$b_i = \begin{cases} 1, \text{ if } x_{2i-1} = 0 \text{ and } (x_{2i} = 1 \text{ or } x_{2i+1} = 1) \\ 0, \text{otherwise} \end{cases}$$

$x1, x2, ..., x8$ are the values of the eight neighbors of p, starting with the east neighbor and numbered in counter-clockwise order.

Condition G2:

$$2 \le \min\{n_1(p), n_2(p)\} \le 3$$

Where

$$n_1(p) = \sum_{k=1}^{4} x_{2k-1} \vee x_{2k}$$

$$n_2(p) = \sum_{k=1}^{4} x_{2k} \vee x_{2k+1}$$

Condition G3:

$$(x_2 \vee x_3 \vee \bar{x}_8) \wedge x_1 = 0$$

Condition G3':

$$(x_6 \vee x_7 \vee \bar{x}_4) \wedge x_5 = 0$$

After thinning the digit, the number of nodes will be computed, whereas, each shape has a specific number of nodes, this number of nodes is repeated at some cases, since we use a large dataset containing several shapes for the same digit. For example, the number four rises in several shapes, each shape generates different number of nodes. To compute the number of nodes, we follow up these steps:

- Start search from the upper left corner to right and from the upper to down.
- Nodes detection: determination of the end point (or tips) of shape, points that have one neighbor only in 8-connected neighboring.
- Compute the number of all nodes in the shape, and then compare the total of nodes with the table.1 for producing the output vector, consisting of expected digits. The table.1 is prepared by experiential tests on dataset containing 176 font type.

Table.(1): results from node method.

Number of nodes	Elements of vector (possible output)
0	{0,8}
1	{ 4, 6, 9}
2	{1, 2, 3, 4, 5, 7}
3	{1, 2, 3, 4, 5, 7}
4	{4, 7}
5	{4}
Over 5	excluded from study

By contemplation in the table.1, can observe 6 cases of nodes, which are concerned with five cases, and the last case (over 5 nodes) is excluded from this study, where those numbers which have over 5 nodes are irregular shapes, and will be needed for orientation with this study.

From table.1 observed the number of nodes for some digits is changeful, for example the number '4' has range from 1 node to 5 nodes. The interpretation for this state: due to the variety of shapes for the same number, in the state of number '4' has 5 types, because we are interested in a large number of type fonts, (Figure .3).

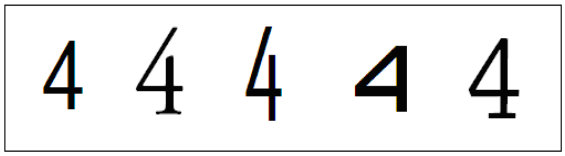


Fig (3): possible shapes for number 4.

3.3.2 Right side method

Each digit is written in a different shape that distinguishes it from others. If we view to drawing of any digit and look to cardinal points (upper, down, right, and left) of digit's drawing, we can expected the identification of digit from the right side of the shape. Therefore, this part of the study will study the right side of the shape of any digit and then produce the possible digits as elements in output vector. To study the shape of any digit, we have to scan the image of shape from right to left and from up to down, according to the following steps:

- Start from upper right corner to the left and from up to down, during that will face one of two cases:
  - Reading all the pixels that lie in this row without any black pixel.
  - Through reading the row of pixels, will find black pixel. Thus, store the (j) index of this pixel in one dimension sorted array. Then, stop reading in this row and shift to next lower row to start new reading with it.
- Start from right side to left side and repeat this for the next rows. Accordingly, will create an array which contains the indexes from first pixel to final pixel for the right side of the shape. Notice the length of array is 42, since the length of input image is 42 (rows) and the content of the array's value is j (columns) for each pixel in the right side of the shape.

After studying the right side of the input shape, as shown in figure 3.10, will form a new shape that represents only right side from the original shape.

The reason for studying the shape from right side only not the shape from other direction (left, upper, down) is specified by experiment, and noticed through studying the shapes of numerals:

- Similarity of the results of studying upper side and down side between the shapes of digits
- The proportion of similarity in left side of the shape is more than proportion of similarity in the right side. And the number of apertures in the left side of the shapes of digits is more than in the right side, as in numerals 2, 3, 5, 9.

The protrusions in the end of limits shape, appears in the left side more than in the right side. (Figure. 4).

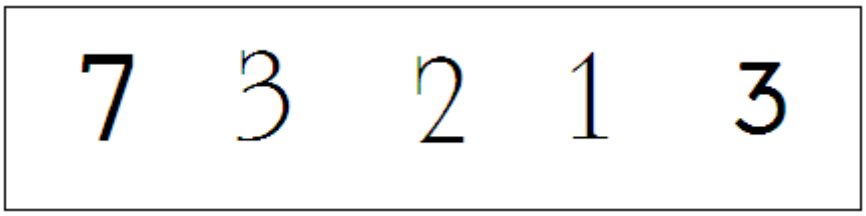


Fig (4): limits of left side of shape.

Table.(2): Cases of right side shape.

	Case 1	Case 2	Case 3	Case 4	Case 5
A	**L** m	**L**	m	m	
		m	**L**	**L**	m
	M		m		**M** m
	m	**R** m		**R** m	
B	**R**		**R**		**R L**

Now, will determine the first element in array and describe it as A, and describe the last element in array as B. Thus, A is higher pixel in the shape (right side shape) and B is lower pixel in the shape (right side shape). Then will determine the pixel that lies in the rightmost side of the shape with reliance on index j, and describe as R. Then, will determine the pixel that lies in the leftmost side of the shape with reliance on index j, and describe as L.

Consequently, there are a set of states for the (L) and (R) presence within the interval [A, B], these states for L and R are shown in the table.2. Details as follows:

- **If (A=L) and (B=R):**

In this case, must find the midpoint between two terminal points A and B, and name it (M). In this manner, will be constituting the two intervals (A, M), (M, B). And for each interval, we must find small (m), where the meaning of (m) is the midpoint in new sub-intervals. The reason to compute small (m); each interval must consist three elements in order to be able to specify the character for each interval, thus, each digit has two characters to describe, one for each interval, as shown in table. 3 and figure 3.5.

- **If (A=L) and (B≠R):**

In this case must find a small (m) between (A, R) and (R, B). See case 2 in the table.2.

- **If (A≠L) and (B=R):**

In this case must find a small (m) between (A, L) and (L, B). See case 3 in the table.2.

- **If(A≠L) and (B≠R):**

In this case must find a small (m) only between (A, L) and (L, R), because, each shape is needed for two characters only.

- **If (B=R=L):**

In this case must find M in (A, B), then find a small (m) between two intervals (A, M), (M, B).

To study the right side of number 3 and explain the way for extracting its features, follow these steps:

- First, will determine the A, B, L, R points from the extracted right side as shown in figure 3-a.
- Then, apply the case which materialize on this shape, in this example on numeral 3, the (A=L) and

(B≠R) correspond to Case 2. Hence, find a small m in [A, R] and a small m in [R, B] as shown in figure 3-b.

- Then, considering the points' places as viewed in figure 3-c and from table.3 and figure. 6, we can specify one character for each interval (or 3 points). The one interval is (1, 2, 3) and upon comparing with table 3.3 we observe it corresponds to character 'C', and the next interval is (3, 2, 1) corresponds to character '/'. The figure 3-d shows the character for the final step.

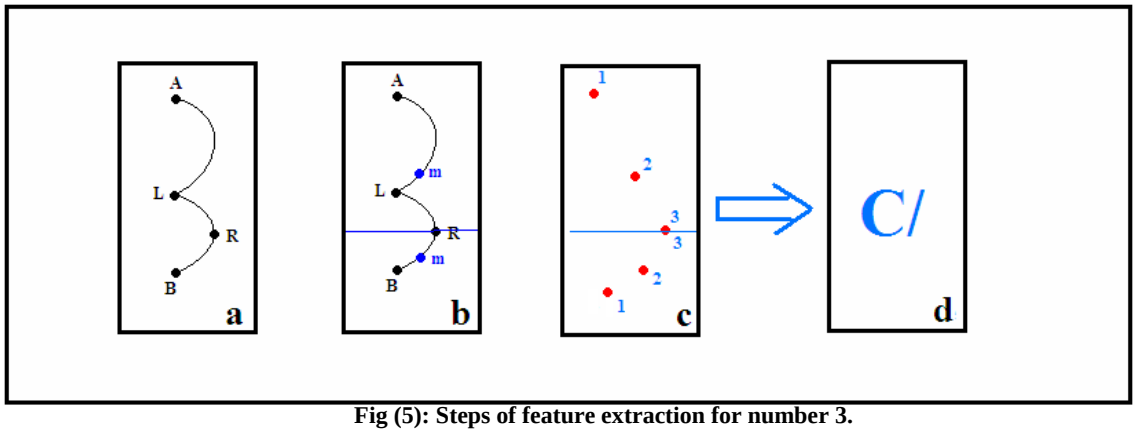


Fig (5): Steps of feature extraction for number 3.

Table .(3): Techniques to determine characters.

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9
A	1	1	3	2	1	2	1	2	1
M	1	2	2	2	1	1	2	1	2
B	1	3	1	1	2	2	1	1	2
decision	**I**	**C**	**/**	**O**	**C**	**C**	**O**	**C**	**O**

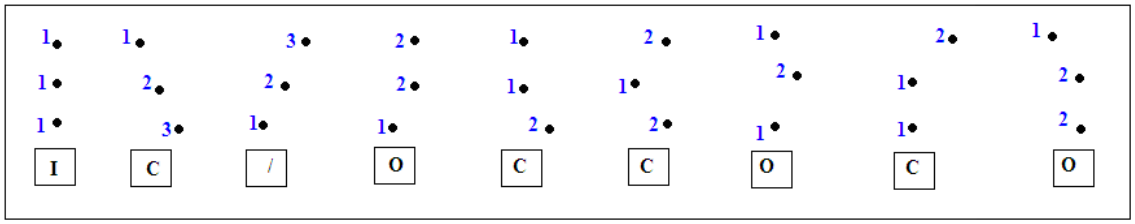


Fig (6): Character Definition.

After that, each shape will be specified by two characters, as follows in the table .4. Finally, the output vector from this method contains the elements which represent the probable digits.

Table .(4): Expected Output Vectors of shape.

Characters of Shape	Output Vectors
OC	**{2, 5, 6, 8}**
CO	**{0, 3, 9}**
II	**{1}**
IC	**{1}**
C/	**{3, 6, 8}**
CC	**{4}**
OO	**{5, 8}**
/C	**{3, 6, 8}**
/O	**{7,8}**

3.4 Classification and Recognition

In this stage of recognition system, will use all features extracted from the shape of input digit to recognize and expect the correct number, where the strength of output for this stage depends on the quality, those features and the methods used in the recognition system. The output of the previous two methods consists of two vectors, each vector contains a set of elements. Thus, in the classification stage, we will match those elements according to table 3.5 which shows all the expected results for all possible states. The dataset used in testing and experiment consists of 128 font type, the names of those fonts are already listed in appendix.

In order to determine and obtain the desired output, we will compare between two vectors by using (if-statement rules). Accordingly, the results might be correct, incorrect, or confusing. All steps of recognition and classification as follows:

1. Find the number of nodes for the shape in image (I), thus, the number of nodes will describe the expected elements in vector of nodes method, and we deal with 6 cases as shown in table 3.1.

2. Studying the shape of digit in the image (I) from the right side, we get two characters that describe the shape of the digit in image (I), the two characters may be anyone from {OC, CO, II, IC, C/, CC, OO, /C and /O} or in some cases it fails. Table 3.4 shows the elements of vector obtained from this method.

3. Now will compare the two vectors and find the intersection elements between the two vectors, as shown in table 3.5. The white area contains one correct result i.e., finality result. If the return value is (-2) this means error in the node and if the return value is (-1) this means error in the shape, and if the return value contains two elements. just as in red area, we continue to step 4.

Table .(5): The results for matching the two vectors.

		Number Of Nodes						
		0	**1**	**2**	**3**	**4**	**5**	**others**
Character of Shape	**Vector elements**	**{0,8}**	**{4,6,8,9}**	**{1,2,3,4,5,7}**	**{1,2,3,4,5,7}**	**{4,7}**	**{4}**	**-2**
OC	**{2,5,6,8}**	**8**	**6,8**	**2,5**	**2,5**			
CO	**{0,3,9}**	**0**	**9**	**3**	**3**			
II	**{1}**			**1**	**1**			
IC	**{1}**			**1**	**1**			
C/	**{3, 6, 8}**	**8**	**6,8**	**3**	**3**			
CC	**{4}**		**4**	**4**	**4**	**4**	**4**	
OO	**{5, 8}**	**8**	**8**	**5**	**5**			
/C	**{3, 6, 8}**	**8**	**6,8**	**3**	**3**			
/O	**{7,8}**	**8**	**8**	**7**	**7**	**7**		
Others	**All**	**0,8**	**4,6,8,9**	**1,2,3,4,5,7**	**1,2,3,4,5,7**	**4,7**	**4**	**-1**

4. From previous steps, we observe the final or desired output mostly determined, but some confusion states need to be treated. The confusion state occurred between (5, 2) and (6, 8). So, we will use extra operation to distinguish between (5,2) and (6,8). In order to correct confusion cases, which presented measures in table 5 to be adopted:

Studying the upper half of the shape from the right side in the same manner to study the all shape, and ignore the lower half of the shape. (Figure.7).

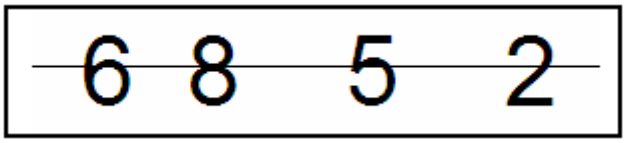


Fig (7): Duplicated case.

- In case (6, 8): if the return value is ['C/' or 'CC'] then the number is 6, else, the number is 8.
- In case (2, 5): if the return value is ['CC' or 'CO'] then the number is 2, else, the number is 5.

4. EXPERIMENTAL RESULTS

The dataset used to evaluate the performance of two proposed methods is made up of sets of images with bitmaps (.bmp Files) and any size for images, where the important thing here is the arrangement of digits inside the image. The digits must be arranged as lines (or rows), and the image may contain one line or more, and each line may contain one digit or more. But in any case, there must be at least one space between each two digits in the same line. And all lines must contain the same number of digits. See the figures (8 a-f), that shows some of the correct cases of image for this study. In this study we assume the image does not include any noise from scanning or digital imaging. Thus, to obtain the same results demonstrated in this paper must observe the condition on dataset.

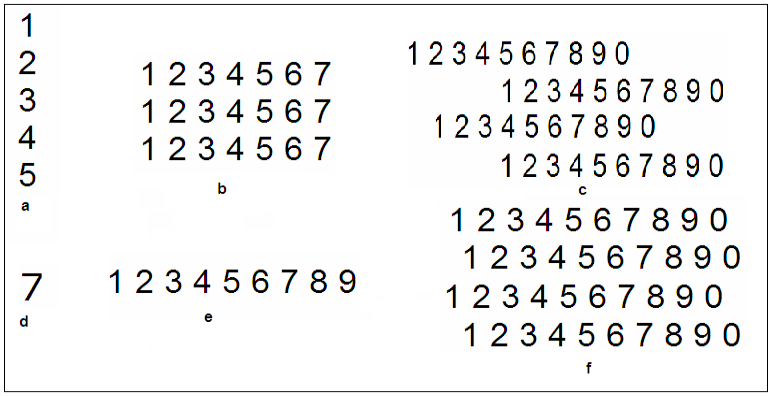


Fig (8): some of samples of data image: (a) 5 lines, 1 column. (b) 3 lines, 7 columns. (c) 4 lines, 10 columns. (d) 1 line, 1 column. (e) 1 line, 10 columns. (f) 4 lines, 10 columns.

In the experiment stage, executed a lot of tests to describe and determine the range of ability for the proposed method in recognition of digits. Some of the results were well, other results were moderate and some results failed. However, all the results of the proposed methods are considered useful for other researchers. In view of good or bad result, it will be used for completion of research in this field. The bad result is useful for other researchers to avoid.

Table 6. describes the recognition rate and explains number of recognition vs. number of non-recognition for 176 numerals. The observation from table 4.1, is that the number of recognition states that it's mostly larger than the number of non-recognition states, except the state of number 8. Nevertheless, 68 times is not a bad result, the errors of recognition results due to error either in shape method or nodes method or both. The node error occur due to cutting of shape which increases number of nodes or changed in number of nodes due to irregular shape. The deformed shapes originally are caused by error reading for right side of the shape.

Table (6): Results for recognize 176 fonts.

description / digits	Recognition Rate	Number of recognized	Number of non-recognize
1	67.6%	119	57
2	74.4%	131	45
3	55.7%	98	78
4	68.8%	121	55
5	61.4%	108	68
6	65.9%	116	60
7	71.0%	125	51
8	38.6%	68	108
9	69.9%	123	53
0	69.9%	123	53

The recognition rate computed in table 4.1, achieved by dividing number of recognition states by number all states (176). Where, each digit from 0 to 9 has an image containing 176 fonts and the size for this image is 320 x 420 pixels. The font size is 12 for all 176 fonts. The figure 9 abridges the relation between recognized states and non-recognized states.

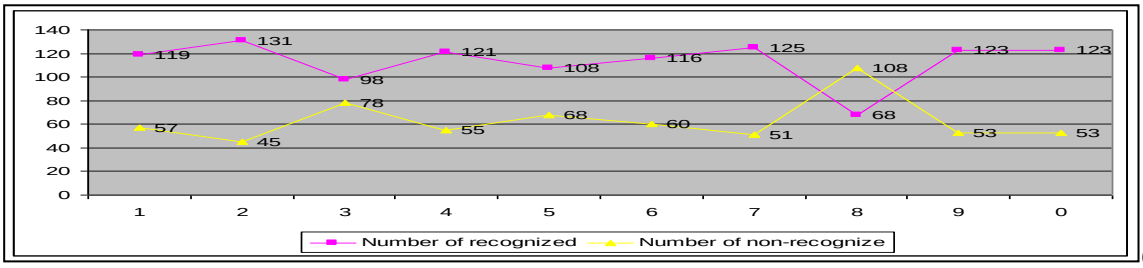


Chart **Fig (9):**

Recognition rate for 176 fonts.

Anyway, attempting to recognize 176 fonts and to reach 100% recognition rate, is surely considered as a complicated task, because, the variation of shapes for typed fonts is massive and similar to variation of shapes for handwritten fonts. If we are trying recognition of the irregular shapes, we must enhance and increase the operations in preprocessing, then use the second proposed method to study the shape from left side and right side. This case for irregular shapes is not included in our study and it's considered as future works.

From the results shown in table 4.1, extracted some of fonts having high recognition rate. The fonts recognized from 0 to 9, where the recognition rate reaches to 100%. Table 7. describes these fonts. The recognition rate for table 7. is (number of recognized digits/10).

Table (7): Details for recognize 10 fonts from 176 fonts.

Description / Fonts	Size of fonts	Recognition Rate
Arial	12	100%
AvantGrade BK	14	100%
Bell Gothis Std Light	14	100%
Eurostar Regular	14	100%
Eurostile	14	100%
Krone	12	100%
Futura Md BT	14	100%
Kalinga	14	100%
Futuist	14	100%
Gulim	14	100%
Zurich Ex BT	12	100%
Overall	**100%**	

The recognition rates for some fonts are lower than 60% and failed for some fonts, i.e., extract the fonts which have the best recognition rates. And found some fonts having low recognition rate or there are digits not recognized. All of these results are customary because the original work did not have 100% recognition rate for all things. Furthermore, the difference between types of fonts is massive. Where, reached to the same difference between handwritten fonts. This is one truth of reasons for not recognizing large number of fonts, dealing with this problem will be one of the issues that will be treated in futures works.

Table 7. shows the sizes for the eleven fonts either 12 or 14. Where, the change in size or font style may be affecting the rate of recognition for some fonts. For example, the font "Arial" with size 12 has 100% recognition rate, but if the style of the font is "Bold", then the recognition rate will drop to 90%. But in some fonts, the change in size or style of fonts does not affect any change on the recognition rate, because the shapes of these fonts don't change through editing its size or style.

The results for multi-sizes fonts:

The experiment on several sizes of fonts was taken into account. Some cases of the results was good and not affected voluminously when the size changed, but in other cases, it's affected in different rates. Type of font and digit itself play a main role when changing the size of the font. For example, the digit "1" is not affected by size, where the results prove the recognition rate was 100% for sizes (12, 18, 24, 36, 54, and 72). On the other hand, the font type "Verdana" has 97% recognition rate for sizes (12, 18, 24, 36, 54, and 72) where they were recognized as follow (1:100%, 2:100%, 3:90%, 4:100%, 5:100%, 6:100%, 7:100%, 8:80%, 9:100%, 0:100%). Results for 9 fonts are described in table 4.3 below. The sizes of tested fonts are (12, 18, 24, 36, 54, and 72).

Table (8): Recognition rate for multi-size

Types of fonts	Recognition rate
Arial	94%
Times	91%
Verdana	97%
SimHei	93%
Serifa	90%
Impact	87%
Gothic	86%
Arial Narrow	93%
Arial Black	92%

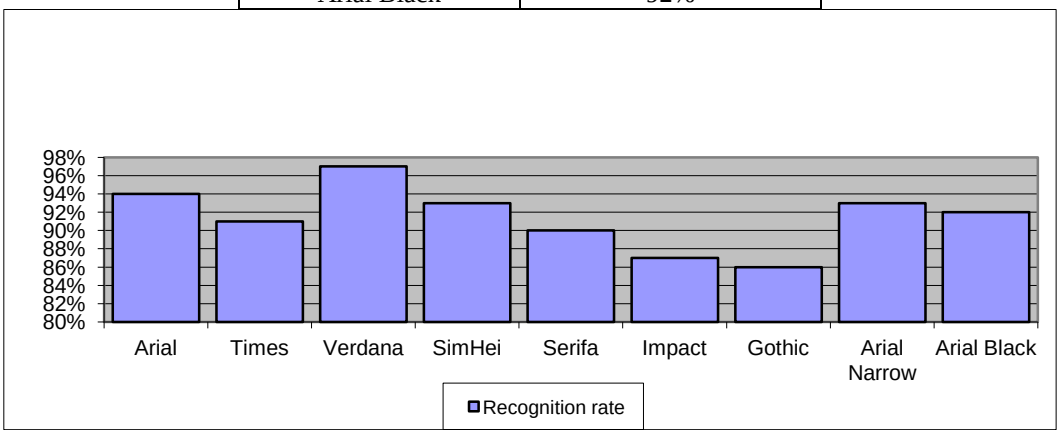


Fig (10): Recognition rate for multi-sizes.

The figure 10. describes the recognition rate for 9 fonts. Where, the rate for some fonts reached to 97% or less than 87%, and other fonts the recognition rate was low or failed. The low and failed results in some cases, werea result of error in studying the shape or error in number of nodes These errors are a result of irregular shapes and disconnected edge of shapes that occur because the original shape of fonts itself is thin or deformed.

After browsing the results in table 8, conclude, that the results obtained from our technique on various fonts sizes proved the flexibility for recognizing the multi-sizes fonts. In principle for the two methods (node method and right side method) have ability to recognize a set of sizes, and we would attribute this ability to two reasons:

I. The node method is detecting the number of nodes in tips (end points) of shape. Consequently, the tips for any digit mostly do not change if we change the size of the shape. Figure (11-b) shows several sizes for number five, all shapes were recognized with recognition rate 100%.

II. The right side shape is almost steady for any size of font. Therefore, the recognition rate when using the two methods together, we must expect good percentage for recognition rate. For example, the figure (11-a) contains seven shapes for number 2 and we realized 100% recognition rate for all shapes in this figure. This proves the flexibility for the two methods in this study.

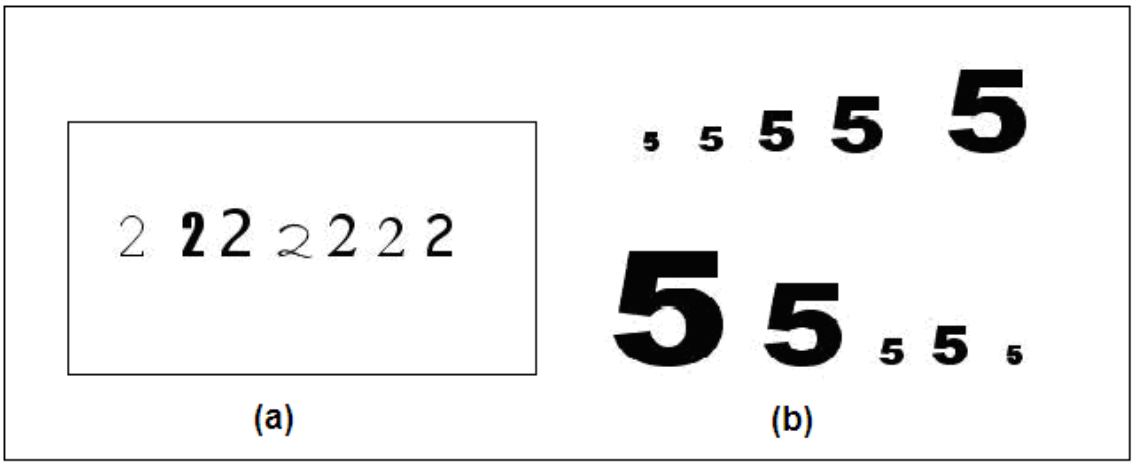


Fig (11): Several shapes for '2' and '5'.

5. CONCLUSION

The proposed technique for digits recognition shows the capability to recognize the digits, in addition to recognition of multi-font numerals. The node method and right side method were applied on large number of fonts.

First, the input image converted to gray scale then converted into binary image with two colors (black and white). Then the binary image passed through series of morphology processes to be more suitable for next processing. The next step includes thinning the shape, then compute the terminal nodes (tips) and then produce one vector containing set of elements. This is called node method. The next step includes the study of shape from the right side, then determine two characters for each shape {OC, CO, II, IC, C/, CC, OO, /C and /O}. And then produce one vector containing a set of elements. This is called the right side method.

The last phase includes dealing with two vectors, each vector contains a set of elements and must compare between two vectors by applying logical comparison operation to obtain the desired output. If the output consist one of two confusion states (6, 8) or (2, 5), we must use the technique to solve this duplicated state, where studying the upper half of the shape from the right side as we did in the previous method.

Through analyzing the experimental results gained, we conclude that the proposed technique is proper, able to give good results, able to recognize 11 fonts in 100% recognition rate, and recognize 9 multi- fonts of different sizes varying from size 8 to 72. The shape study was applied through a set of operations with low complexity, and the classification stage relies on logical comparison operation. If this method is used from four sides (left, right, upper, down) that will give better results and may used in recognizing handwritten digits.

The technique is flexible for a change in size and type of fonts, and the technique is simple and doesn't need complex computations, where, we used (if statements rules) to decide the desired result, thus, we can use it in the devices with small memory and low processor, and ability to recognize irregular shapes for some of fonts.

6. REFERENCES

[1] M. Hanmandlu, M. Hafizuddin and V.K. Madasu, "Fuzzy based approach to the recognition of multi-font numerals", IEEE Trans. On fuzzy systems, vol. 4, no., 24-52, 2003.

[2] M. Razzak, S.A. Hussain, A. Belaid and M. Sher, "Multi-font numerals recognition for Urdu script based language", international Journal of recent trends in engineering, Vol. 2, No. 3, November 2009.

[3] A. Malaviya and R. Klette, "A Fuzzy Syntactic Method for On-line Hand writing Recognition", advances in structural and syntactic pattern recognition, SSPR'96, pp. 381-392, 1996.

[4] V.K. Govindan and A.P. Shivaprasad, "Character recognition – A review", Pattern Recognition, vol. 23, no. 7, pp.671-683,1990.

[5] C.Y. Suen, C. Nadal, R. Legault, T.A. Mai and L. Lam, "Computer recognition of unconstrined handwritten numerals", Proc. Of IEEE, vol. 80, no. 7, pp. 1162-1180, 1992.

[6] R. Plamondon, N. Srihari, "On-line and Off-line Hand writing Recognition: A Comperhensive Survey", IEEE Trans. On PAMI, vol. 22, no.1, pp. 63-84, January 2000.

[7] Liying Zheng, "Recognition for Arabic Character Based on Edge and BPNN", Proceedings of the world congress on Engineering and computer since, San Francisco, USA, October 2008.

[8] H. Ebrahimnezhad, GH. A. Montazer and N. Jafari, "Recognition of Persian numeral fonts by combining the entropy minimized fuzzifer and grammar", proceedings of the 6th WSEAS int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, February 16-9, 2007.

[9] Z.Chi, H.Yan,"ID3-Derived Fuzzy Rules and Optimized Defuzzification for Handwritten Numeral Recognition",IEEE transiction on fuzzy system,VOL, 4, February 1996.

[10] P Pyeoung Kee Kim," Improving Handwritten Recognition Using Fuzzy Logic", Pusan Women's University,Korea, 2004.

[11] N. Belkhamza, A. Akkouche, "Fuzzy Logic Character Recognition", Madinah College of Technology, Saudi Arabia, 2004.

[12] B.V.Dhandra, V.S.Malemath, H.Mallikarjun, and H. Mallikarjun, "Multi-font Numeral Recognition without Thinning based on Directional Density of Pixels", Gulbarga Univ., Digital Information Management, 1st IEEE International Conference, 2006.

[13] N.S.Arjun, G.Navaneetha, G.V.Preethi and T.K.Babu, "An approach to multi-font numeral recognition", TENCON 2007 - 2007 IEEE Region 10 Conference , Taipei , 2007 .

[14] L.Lam, Seong-Whan Lee, and Y. Suen Ching, "Thinning Methodologies-A Comprehensive Survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 14, No. 9, pp. 869-885, September 1992.

[15] Frank Y.Shih, "Image Processing and Pattern Recognition Fundamentals and techniques", John Whily and Sons, Directory of IEEE Book and Information services, pp.3-12, 2010.

[16] Frank Y.Shih, "Image processing and Mathematical morphology fundamentals and techniques", Wiley-IEEE, CRC Press, United State of America, pp. 4-17, 2009.

[17] R. C .Gonzalez, and Woods, "Digital image processing", second edition, New Jersey: Prentice Hall, pp. 34-69, 2002.