RECOGNITION OF AMBIGUOUS PAIRS OF TOTALLY UNCONSTRAINED HANDWRITTEN NUMERALS

YU TAN

A MAJOR REPORT
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE
CONCORDIA UNIVERSITY
MONTREAL, QUEBEC, CANADA

JANUARY 2002

0-612-68480-6

Abstract

Recognition of Ambiguous Pairs of

Totally Unconstrained Handwritten Numerals

Yu Tan

In order to improve the recognition rate of totally unconstrained handwritten numerals, a verification stage is added after classification to distinguish the ambiguous numerals between two or more classes. Two recognition methods, structural method and statistical method, are used to overcome the limitations of a single method and deliver a much more reliable recognition system. The recognition rate is 71.32% using structural method without verification. The verification stage improves the recognition rate from 71.32% to 87.50%. After combining the structural method with statistical method, the final recognition rate reached up to 95.59%.

To my family

Acknowledgement

I would like to thank my supervisor Professor Ching Y. Suen, the founder of the International Workshop on Frontiers in Handwriting Recognition, for his guidance and encouragement throughout. His solid knowledge and experience attracted me to this field. I also would like to take this chance to thank Dr. T. Kasvand for his suggestions and comments.

Contents

List of Figures

1. Introduction

Handwriting recognition still remains as one of the most difficult problems in the frontier of pattern recognition research. Although there have been revolutionary changes in both software and hardware during the last 20 years, there are still many technical issues to be overcome in order to make handwriting practical for the human interface with computers [1, 2].

There are three major kinds of methods used to try to capture the distinctive features of handwritten numerals: structural method [3], statistical method [4] and artificial neural networks [5]. But each method has its own weaknesses, and none of them can achieve a perfect recognition rate. However, the combination of different methods can overcome the limitations of a single method and deliver a much more reliable classification result [6].

In this project, the structural method is used at first to classify the input pattern. This is accomplished by the following procedures: preprocessing, feature extraction, classification and verification. The preprocessing includes filling, thinning and skeletonization. The feature extraction includes decomposition, creating chain codes and extracting features. Base on the features extracted, the input pattern will be classified into one or more classes with confidence levels. If the confidence levels are lower than the threshold, or the two confidence levels of two classes are too close to each other, the verifier will be invoked. Verification looks closely at certain parts of the numeral images to verify its identity [9, 14]. Then the statistical method will be used to validate the identity of the input image. The final recognition will be based on the results of both the structural method and the statistical method.

2. Structural Methods

In the structural method, efforts are aimed at capturing the essential shape features of characters, generally from their skeletons or contours. Such features include loops, endpoints, junctions, arcs, concavities, convexities and strokes. Most often, a syntactical classification approach is used with structural features [7, 8, 11].

The structural method used in this project comprises the following four major steps:

- The first step is preprocessing that includes filling, thinning and skeletonization to obtain the skeleton of the input image and prepare for feature extraction.

- The second step is feature extraction. It includes decomposition, creating chain codes, and extracting features. The features comprise end points, junctions, loops, cups, caps, left open concavities and right open concavities.

- The third step is classification. Totally 10 models are used to evaluate the input image. There is one model for each numeral. Each model will return a confidence level for the input numeral image. If two confidence levels are too close to each other, or all confidence levels are lower than the threshold, the verification step will be used.

- The last step is verification, which takes a closer look at certain parts of the ambiguous numeral image to verify its identity. For example, in order to prove the identity of an input numeral image is '4', the system checks the curvature changes at the top part and the left part of the loop extracted from the input image, to verify its identity is '4' or '9'.

2.1 Preprocessing

Preprocessing is to get rid of the useless information, remove noise and change the data into a format that is suitable for the feature extraction. It includes filling, thinning and skeletonization.

The input data used in this project come from NIST database. All images are already digitized and binarized. The region of the numerals has pixel values of 1's and the rest of the 'empty' regions are filled with 0's.

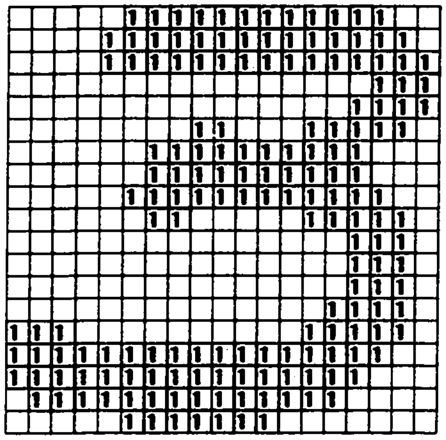


Figure 1. Digitized And Binarized Image

2.1.1 Filling

Filling is to add pixels to the image to reduce random noise.

a	b	c
h	p	d
g	f	e

Figure 2. A 3x3 Window For Filling

A 3x3 size window is used to scan all pixels of the input image. If the value of the pixel p equals 1, the system will do nothing for filling this pixel. If the value of the pixel p equals 0, and the values of its neighbors, pixel h, a, b, c, d, all are 1, the system will set p to 1. Rotating the window, we can get this formula:

```
if (p == 0)
{
if (a*b*c*d*h + b*c*d*e*f +d*e*f*g*h + f*g*h*a*b > 0)
        p = 1;
else
        //do nothing
}
```

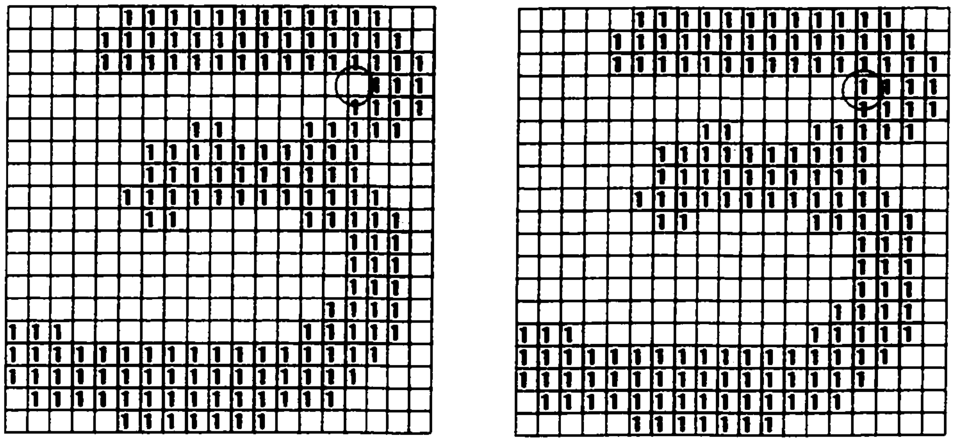


Figure 3. Filling

2.1.2 Thinning

Thinning is to remove pixels in the images to reduce random noise and bumps.

a	b	c
h	p	d
g	f	e

a	b	c
h	p	d
g	f	e

Figure 4. A 3x3 Window For Thinning

A 3 by 3 window is used to scan all pixels. If the value of the pixel p equals 1, the system will check its 8 neighbours. Otherwise, the system will move the window to check the next pixel. The algorithm is as follows:

- If p is 0, the system will do nothing for thinning.
- If p is 1, and eight neighbors are 0, the system will remove this random noise to set p to 0.
- If p is 1, and a, b, c are all 1, but d, e, f, g, h are all 0, the system will convert p to 0 to remove a bump.

Rotate window, we can get this formula:

```
if(  (a*b*c !=0 && (d+e+f+g+h)==0)
   || (c*d*e !=0 && (f+g+h+a+b)==0)
   || (e*f*g !=0 && (h+a+b+c+d)==0)
   || (g*h*a !=0 && (b+c+d+e+f)==0)
   || (a+b+c+d+e+f+g+h == 0)   )
{
        p=0;
}
```

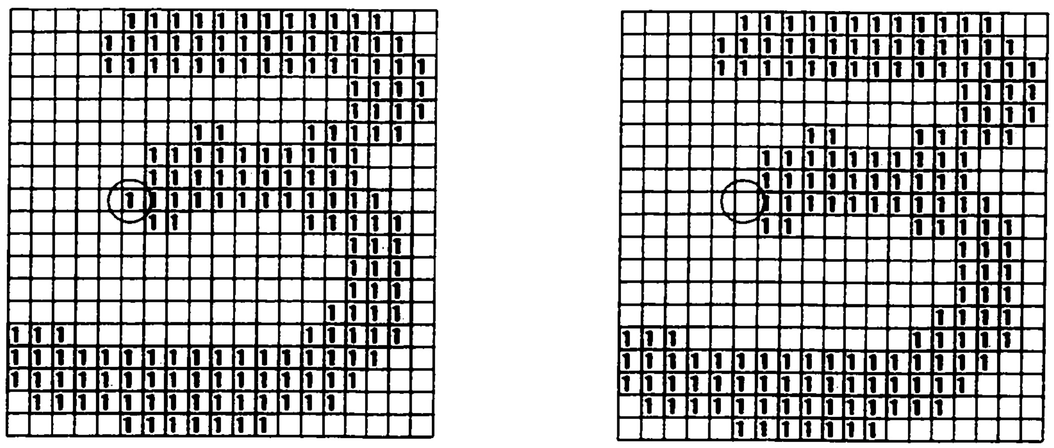


Figure 5. Thinning

2.1.3 Skeletonization

In this project, Zhang-Suen Skeletonizing method [10] is used to create a skeleton of the input image. Zhang-Suen Skeletonizing repeatedly removes the boundary points from a region in a binary image until an irreducible skeleton remains.

The Zhang-Suen Skeletonizing transform is a parallel algorithm that reduces regions of a Boolean image to an 8-connected skeletons of unit thickness. Using 3x3 window, the system checks whether the point p can be deleted without corrupting the ideal skeleton.

p8	p1	p2
p7	p	p3
p6	p5	p4

Figure 6. A 3x3 Window For Skeletonization

If p is a contour point and its 8-neighbors satisfy the following four conditions listed below, then p will be removed in the first sub-iteration.

The conditions for boundary pixel removal are:

a) $2 \leq Bp \leq 6$

b) $Ap = 1$

c) $p1 * p3 * p5 = 0$

d) $p3 * p5 * p7 = 0$

Bp: the number of nonzero 8 neighbors of p

Ap: the number of zero-to-one transitions in the ordered sequence p1, p2, .. p8, p1.

Condition (a) insures that endpoints are preserved. Condition (b) prevents the deletion of points of the skeleton that lies between endpoints. Conditions (c) and

(d) select southeast boundary points and northwest corner points for the first sub-iteration.

A contour point will be subject to deletion during the second sub-iteration provided its 8-neighbors satisfy conditions (a) and (b) above, and conditions (c′) and (d′) are given by

c′) $p1 * p3 * p7 = 0$

d′) $p1 * p5 * p7 = 0$

Iteration continues until either sub-iteration produces no change in the image.

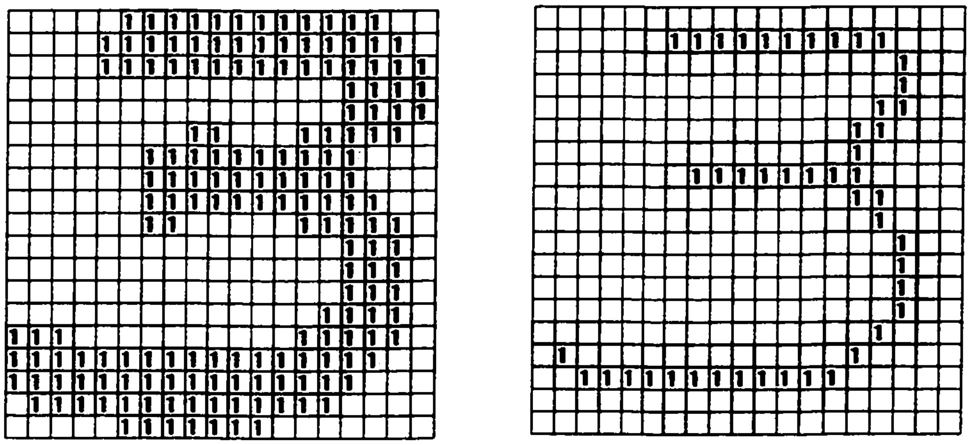


Figure 7. Skeletonizing

2.2 *Feature Extraction*

Feature extraction derives the high level information from individual patterns and tries to reduce the redundancy. It first decomposes the skeleton into branches, then creates a chain code for each branch, and finally extracts features from the chains.

2.2.1 Decomposition

While scanning through and storing the skeleton in matrix form, end-points and junctions are detected. These points are used to determine the branches that form the pattern [6]. The end points are marked by letter 'E' and the junctions are marked by letter 'J'.

But sometimes, even a point has more than 2 neighbors, it still may not be a junction. Figure 8 shows that a point has 3 neighbors, but it is not a junction. If the junction point is detected by checking whether it has 3 or more neighbors, an inflection point cannot be distinguished.

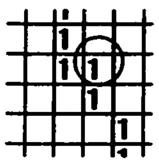


Figure 8. Curved Point, Not A Junction

Also if one pixel is already marked as a junction, at the same junction area, we should not mark its neighbors again. This brings a lot of convenience to the feature extraction.

The method used to check end points and junctions is as follows, the positions of the eight neighbors a, b, c, d, e, f, g and h refer to Figure 4.

```
// check end point
if(      a+b+c+d+e+f+g+h == 1)
{
         m_cell[i][j] = 'E' ;
}
//one neighbor is Junction, do not mark junction again.
else if (a=='J' || b=='J' || c=='J' || d=='J' || e=='J' || f=='J' || g=='J' || h=='J')
         continue;
//find a Junction
```

```
else if( a*c*(e+f+g) + c*e*(g+h+a) + e*g*(a+b+c) + g*a*(e+d+c)
        + h*b*d + b*d*f + d*f*h + f*h*b
        + a*d*f + c*f*h + e*h*b + g*b*d > 0)
{
        m_cell[i][j] = 'J' ;
}
```

Starting from a J-point, one of its neighbors is selected. By traveling through the skeleton from neighbor to neighbor until another J-point or an E-point is found, A branch of the pattern is determined. Similarly, all the branches are defined and examined. Due to skeletonization, sometimes extra small branches appear on the skeletons of some patterns written with heavier strokes. To correct and simplify such skeletons, a branch found to have a length less than 5% of the total number of pixels in the skeleton is deleted.

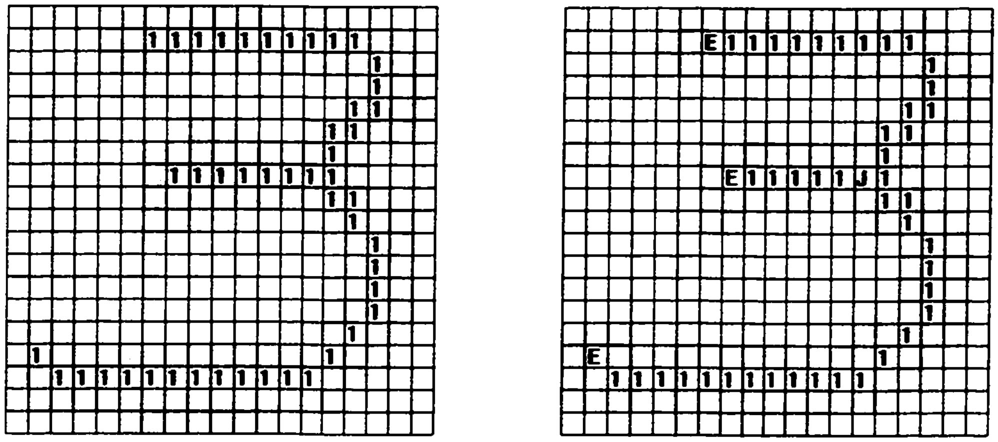


Figure 9. Decomposition

2.2.2 Create Chain Codes

After the decomposition, the skeleton is separated into branches. Each branch is a chain. The system will create a chain code for each branch in order to facilitate the extraction of features.

Each chain has a start point, an end point, integer length, and a string to store the chain code.

```
struct CHAINS
{
        char chainCode[200];
        int m_length;
        int m_StartPointX;
        int m_StartPointY;
        int m_EndPointX;
        int m_EndPointY;
};
```

The chain code is determined by the direction from the current point to the next point.

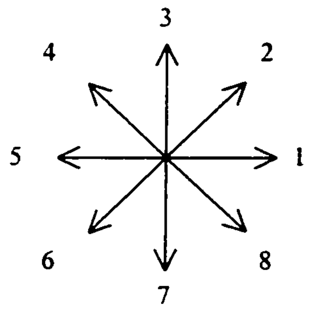
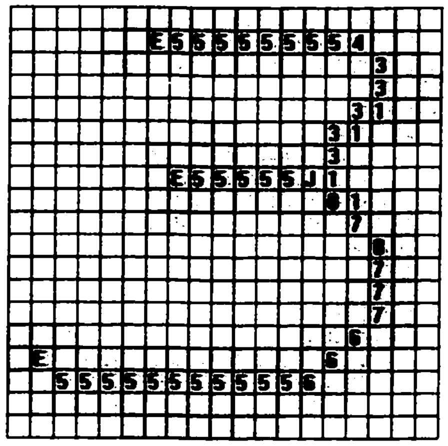


Figure 10. Chain Code

To create a chain code, we start from a junction or an endpoint, sort them by their coordinates (x + y). If no junction and endpoint is found, we start from any point where its value is 1. We set the current position to this point, following a specified order, from direct neighbor to indirect neighbor, from top to down, from left to right, to find its neighbors. If a neighbor is found, we save this chain code, set a flag to avoid re-calculating this point and falling into an infinite loop. Then we move the current position to the neighbor just found, repeat the above process until all points are checked.

Figure 11 shows the chain codes for the input numeral image '3'.



```
Chain1- 5 5 5 5 5
     StartPoint: [7, 13]; EndPoint: [7, 7];
Chain2- 1 3 3 1 3 1 3 3 4 5 5 5 5 5 5 5 5
     StartPoint: [7, 13]; EndPoint: [1, 6];
Chain3- 8 1 7 8 7 7 7 6 6 6 5 5 5 5 5 5 5 5 5 5 5
     StartPoint: [7, 13]; EndPoint: [15, 1];
```

Figure 11. Create Chain Codes

2.2.3 Extract Features

When all branches have been determined, the following features are extracted from the chain codes:

- Endpoints: number of endpoints and their positions
- Junctions: number of junctions and their positions
- Loops: number of loops
- Cup: number of cups
- Cap: number of caps
- Left-open-concavity: number of left-open-concavities
- Right-open-concavity: number of right-open-concavities

All features are extracted by reading the chain codes. A function is created to read a chain code, and to check whether the chain code satisfies a special feature.

```
BOOL ReadChainCode(CHAIN aChain, double confidenceLevel,
                   char direction1, double maxPercentage1,
                   char direction2, double maxPercentage2,
                   char direction3, double maxPercentage3);
```

CHAIN aChain: a specified chain code to be read.

Double confidencLevel: the threshold to decide true or false.

Char direction1, double maxPercentage1: one specified direction and the max contribution for the final decision.

For example, to find the top part of the right-open-concavity the essential chain code feature is "2-1-8".

ReadChainCode(aChain, 0.80, 2, 0.50, 1, 1.00, 3, 0.50);

If this function returns TRUE, which means that the input chain *aChain* satisfies the feature of the top part of the right open concavity.

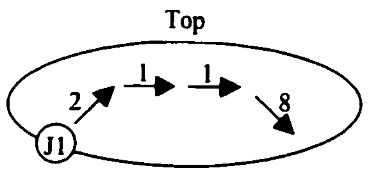


Figure 12. Read Chain Code

Figure 13 shows a right-open-concavity as an example to explain how to extract the features in this project.

The right-open-concavity can be divided into three parts: top, left and bottom. Each part has a typical feature. For example, the chain code of the top part feature is 2-1-1-8. The chain code can also be in a reverse direction 4-5-5-6 for the top part feature.

The three parts of the left-open-concavity can have at most two junctions J1, J2. They may also just have one junction J1, the left part and bottom part combined together, or one junction J2, top part and left part combined together. Also they may not have any junction at all.

In the two-junction case, first we need to find the left part, that is a chain code that satisfies the feature 4-3-2 or 6-7-8, then check whether there exists a chain which satisfies the top part. The start point of the top part should be equal to the end point of the left part and satisfy the feature 2-1-8. We do the same for the bottom part. If all of the left part, the top part and the bottom part are found, then we can say that a right-open-concavity is found.

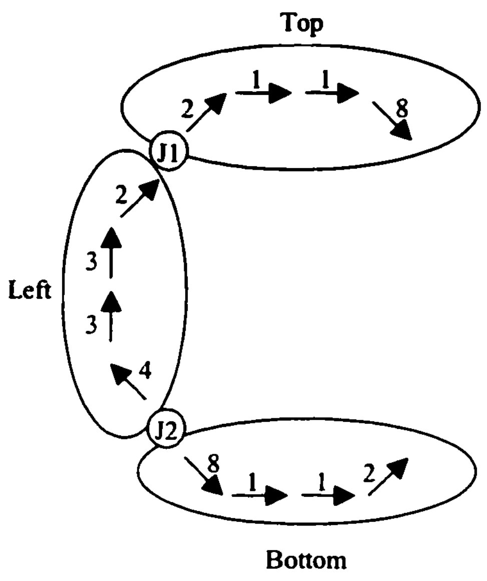
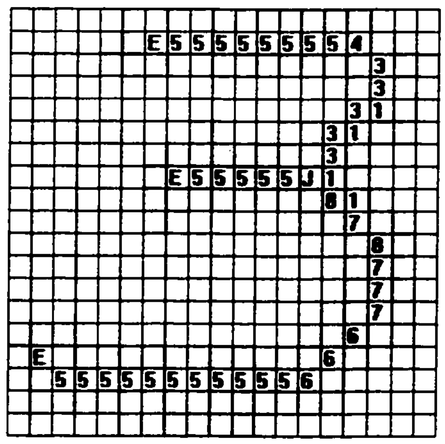


Figure 13. Detect Right Open Concavity By Chain Code

In this way the system reads all chain codes, and extracts all features.

Figure 14 shows the features extracted from the chain code of numeral '3'.



```
There are 3 end points:
    [1, 6],   [7, 7],   [15, 1],
There are 1 Junctions:
    [7, 13],
There are 0 Loop[s]
There are 0 Cup[s]
There are 0 Cap[s]
There are 2 Left-Open-Concavity[s]
There are 0 Right-Open-Concavity[s]
```

Figure 14. Feature Extraction

2.3 *Classification*

Based on the configuration of primitives, patterns with simpler structures are classified by decision trees according to the primitives and their extracted features.

To determine the identity of the target numeral image, 10 independent modules (module0 to module9) are used. Each module has several criteria for determining a number. For example moduleX determines whether the numeral image is the number X or not, and it determines independently from the other modules. A numeral image is inputted into all of the 10 modules and moduleX determines whether the input image is a number X. There can be none or more than one module that responds positively to the input numeral image. The conflict of multiple acknowledgments of the numeral modules will be resolved by verification stage or statistical method.

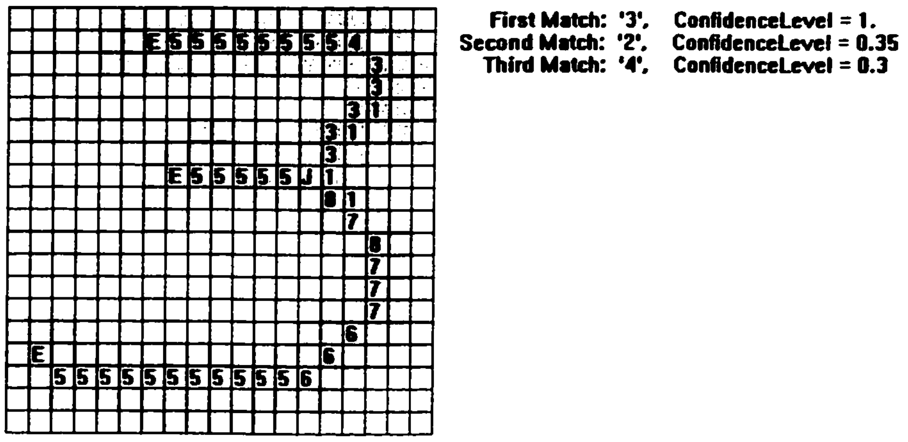
The confidence level of numeral i is the sum of n features that the input image satisfied. The range of the confidence level is 0 to 1.

$$ConfidenceLevel(i) = \sum_{k=0}^{n} F(i,k);$$

Each module will consider part or all of these factors:

- The number of endpoints and their positions
- The number of junctions and their positions
- The number of loops
- The number of cups
- The number of caps
- The number of left-open-concavity
- The number of right-open-concavity

Figure 15 shows that the input image is classified as numeral '3' with a confidence level of 1, classified to '2' with a confidence level of 0.35 and classified to '4' with a confidence level of 0.3 based on the features extracted, 3 endpoints, 1 junction, 2 left-open-concavities.



```
 First Match: '3',  ConfidenceLevel = 1.
Second Match: '2',  ConfidenceLevel = 0.35
 Third Match: '4',  ConfidenceLevel = 0.3
```

Figure 15. Classification

2.4 *Verification*

Verification is to look closely at certain parts of the characters to verify its identity.

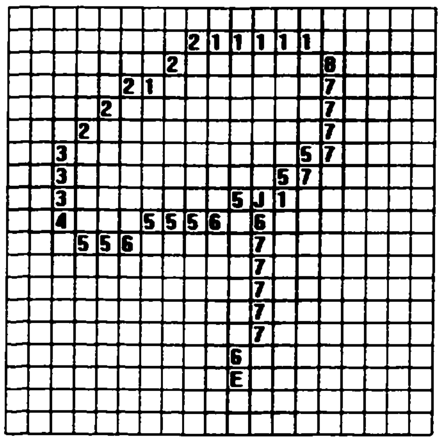
2.4.1 Verify '4' And '9'

The first two ambiguous classes are '4' and '9'. The system cannot distinguish them from the features extracted in the previous classification stage because they have the similar features.

Figure 16 shows how similar numerals '4' and '9' are. For the first pair, both of them have 1 loop, 1 junction and 1 endpoint. The position of loop, endpoint and junction are similar. For the second pair, both of them have 1 cup, 1 junction and 3 endpoints. The position of cup, junction and endpoints are similar.

Figure 16. Verify '4'-'9'

The classification stage cannot determine their identities. Figure 17 shows the classification results of input numeral '9'. The system cannot determine the identity of input image because they have the same confidence level for numerals 4 and 9.



```
   First Match: '4',  ConfidenceLevel = 0.9
Second Match: '9',  ConfidenceLevel = 0.9
   Third Match: '0',  ConfidenceLevel = 0.8
```

Figure 17. Confusion Between '4' And '9'.

To verify '4' and '9', the system needs to look closely at the certain parts of the input image. The following features are extracted and verified by the system in the verification step in order to distinguish between numerals '4' and '9':

- The curvature change of the right-top part of the loop. If the curvature changes more sharply, the input image is closer to numeral '4'.
- The curvature change of the left-bottom part of the loop. If the curvature changes more sharply, the input image is closer to numeral '4'.

- The angle between the loop and the vertical stroke. If it is a right angle, the input image would be numeral '4'.
- The distance between the two top endpoints. If there is no loop, and the distance between the two top endpoints is smaller, the input image is closer to numeral '4', otherwise the system can not determine their identities.
- The length of the vertical tail. It the tail is longer, the input image will be closer to numeral '4', otherwise it cannot be determined.

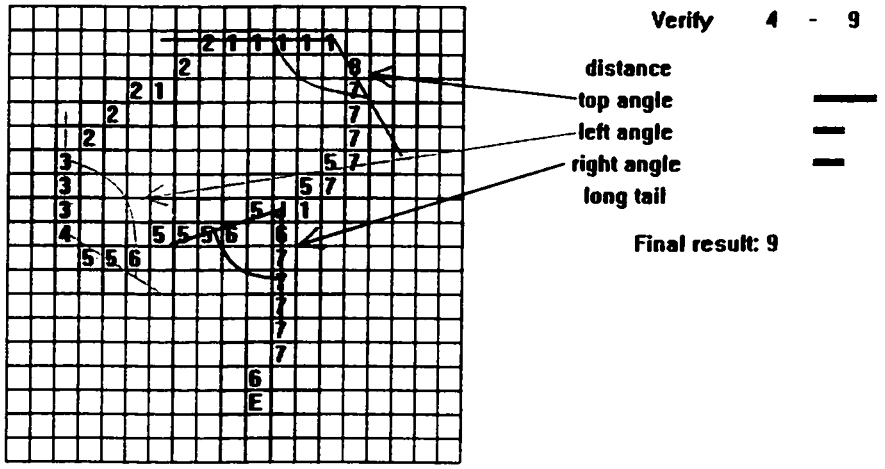


Figure 18. Verify '4' - '9' (1)

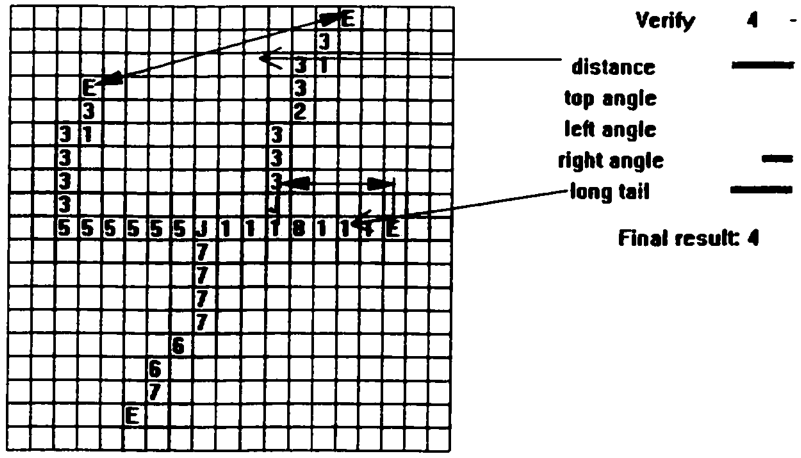


Figure 19. Verify '4' - '9' (2)

2.4.2 Verify '0' And '6'

The other two ambiguous classes are '0' and '6'. It is also very difficult to distinguish between them from the features extracted in their previous stage.

Figure 20 shows how similar numerals '0' and '6' are. For the first pair, they both have 1 junction, 1 endpoint and 1 loop. The endpoints are at similar positions. For the second pair, they both have 2 endpoints, no loop and no junction. Cup, left-open-concavity and right-open-concavity may be detected.

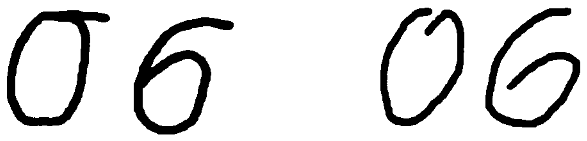
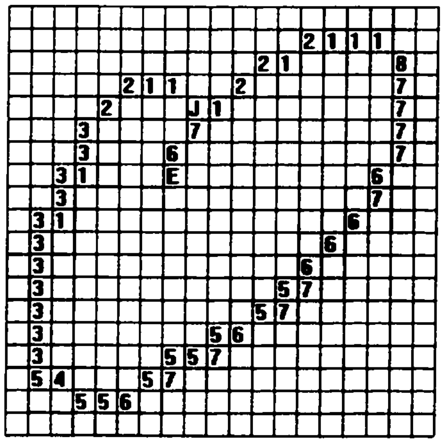


Figure 20. Verify '0' - '6'

They cannot be identified by the above classification procedure. Figure 21 shows the classification results of the input numeral '0'. The classification can not determine the identity of the input image because they have the same value of confidence level for both numerals '0' and '6'.



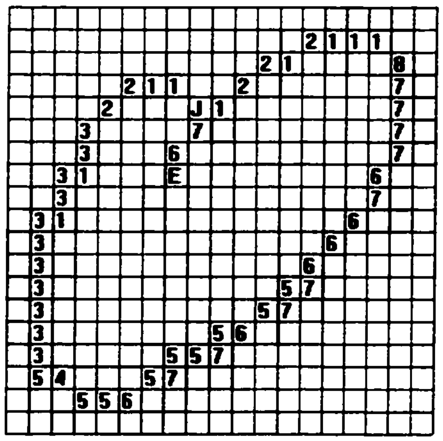
```
 First Match: '0',  ConfidenceLevel = 0.8
Second Match: '6',  ConfidenceLevel = 0.8
 Third Match: '-1',  ConfidenceLevel = 0.
```

Figure 21. Confusion Between '0' And '6'

To verify 4 and 9, the system needs look closely at certain parts and extract more features. The following features are extracted and verified in the verification step in order to distinguish numerals '0' and '6':

- The distance from the endpoint to the arc. If there is no loop, and no junction, we have to check the distance from the top endpoints to the top of the arc that forms the loop.
- The endpoint is inside the loop. In the case of 1 endpoint and 1 loop, if there is 1 endpoint and 1 loop, and the endpoint is inside the loop, the input image must be '0'.
- The position of the endpoint. In the case of 1 endpoint and 1 loop, if the position of the endpoint is at the left side of the input image, the input image must be '0'.
- The distance from endpoint to loop. In the case of 1 endpoint and 1 loop, the shorter the distance from the endpoint to the loop, the closer the input image to '0'.

Figures 22 and 23 show the verification results of input images that can not be identified in the classification step.



```
                 Verify    0  -  6

            Distance to Arc
    End Points Inside Loop  ——
    Position of 1 End Point
          Distance to Loop

          Final result: 0
```

Figure 22. Verify '0' - '6' (1)

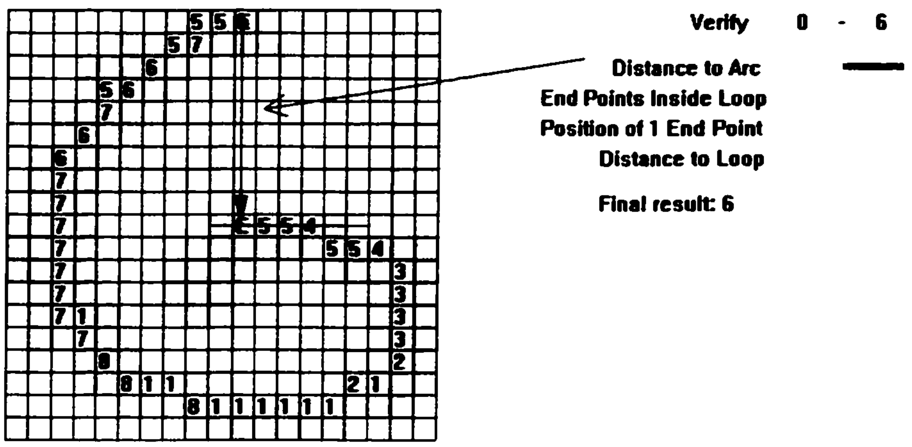


Figure 23. Verify '0' - '6' (2)

3. Statistical Methods

First, the system builds up templates for each numeral using the training data, then matches the input numeral images against the templates and calculates the distances between the input image and each template. The identity of the input image is then classified according to a minimum distance criterion.

In this project, all input numeral images are size normalized before they are inputted into the system. Tanimoto Similarity distance metric is used to calculate the distances between the input image and a set of size-normalized templates. The 3 templates that match the input numeral image most closely are found with their similarities.

3.1 *Normalization*

Normalizing the size of the images of the numerals is the first step since we cannot compare the images in two different sizes using the Tanimoto Similarity distance metric. All input images are normalized into 19 by 19.

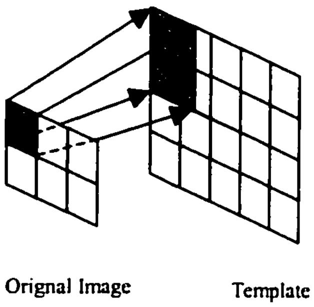


Figure 24. Size Normalization.

Figure 24 shows the algorithm which projects an input image into a window of a different size.

```
fLeftBound = j* widthRatio;
fRightBound = (j+1)*widthRatio;
fTopBound = i* heightRatio;
fBottomBound = (i+1)*heightRatio;
```

Figure 25 shows the original image of numeral '2' in size 40X33 is normalized into size 19X19.

```
w = 40 h = 33 Truth = 2
0000000000000001000000000000000000000000
0000000001111111100000000000000000000000
0000001111111111111100000000000000000000
0000011111111111111111000000000000000000
0000011111111111111111100000000000000000
0000111111111111111111110000000000000000
0001111111110000000111111100000000000000
0001111111000000000111111110000000000000
0001111100000000000011111100000000000000
0001110000000000000001111110000000000000|
0000000000000000000001111110000000000000
0000000000000000000001111110000000000000
0000000000000000000001111110000000000000
0000000000000000000001111110000000000000
0000000000000000000001111111000000000000
0000000000000000000000111111000000000000
0000000000000000000000111111000000000000
0000000000000000000000111110000000000000
0000000000000000000000111110000000000000
0000011111111110000000111110000000000000
0001111111111111110001111110000000000000
0011111111111111111111111110000000000000
1111111111111111111111111100000000000000
1111111111111111111111111100000000000000
1111111000000000111111111100000000000000
1111110000000000011111111111000000000000
1111110000000000111111111111110000000000
1111110000000001111111111111111111000000
1111111111111111111111111111111111111110
0111111111111111111111000001111111111111
0111111111111111111110000000001111111110
0001111111111111110000000000001111111100
0000111111111111000000000000000000000000
```

```
w = 19 h = 19 Truth = 2
0000011100000000000|
0001111111000000000
0011111111100000000
0111111111110000000
0111000001111000000
0110000000111000000
0000000000111000000
0000000000111000000
0000000000111000000
0000000000111000000
0000000000111000000
0011111100111000000
1111111111111000000
1111111111111000000
1111000111111000000
1110000111111111000
1111111111111111111
1111111110000111111
0011111110000001110
```

Figure 25. Size Normalized Image

3.2 *Calculate Distance*

A training set of 500 training images is inputted into the system to build up the template classes. Then the size-normalized input numeral image is compared by a set of size-normalized templates using the Tanimoto Similarity distance metric. The 3 templates that most closely match the input character are then determined.

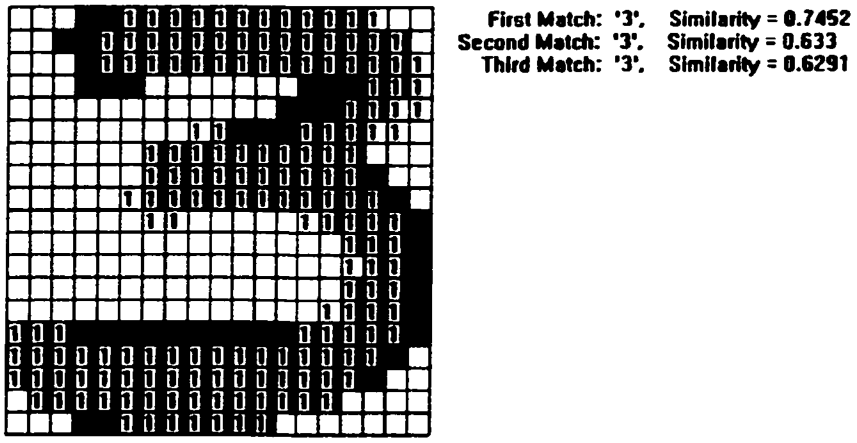
Tanimoto Similarity:

Similarity $T=P\cap Q/(P\cup Q-P\cap Q)$

$P\cap Q$ *is common features*

$P\cup Q-P\cap Q$ *is total number of features*

There are 50 templates for each class. Figure 26 shows the best three matched templates with their similarities. The dark color background is the best matching template in the training set.



```
 First Match: '3',  Similarity = 0.7452
Second Match: '3',  Similarity = 0.633
 Third Match: '3',  Similarity = 0.6291
```

Figure 26. Classification By Statistical Method

The final recognition result is obtained by comparing the results from both structural method and statistical method. Figure 27 shows the final result combining the structural method and statistical method for the input image of numeral '3'.

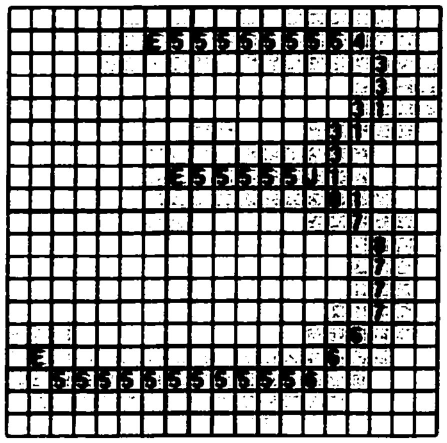


Figure 27. The Final Result Of Recognition

When we know the identity of the input image we can append this image into the training template sets in order to better classify it next time. We can dynamically train the system. Figure 28 shows the input image is appended into the templates.

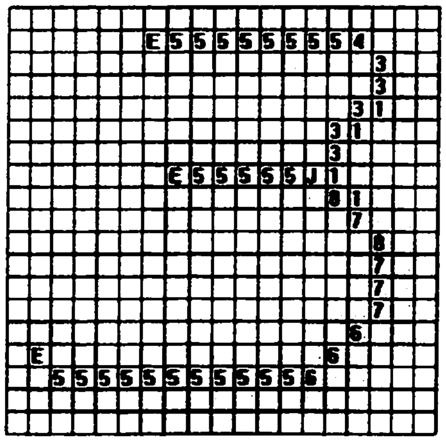
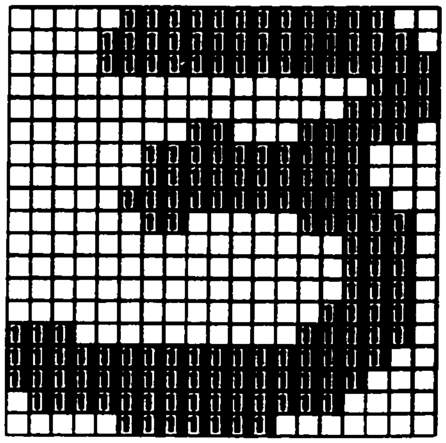


Figure 28. Dynamically Train System

Figure 29 shows that the system returns a better matching result after appended the input numeral image '3' into the templates.



```
 First Match: '3',  Similarity = 1.
Second Match: '3',  Similarity = 0.7452
 Third Match: '3',  Similarity = 0.633
```

Figure 29. Better Match After Training

4. Implementation And Conclusion

Totally 1000 numeral images from NIST database are used in this project. They are already digitized and binarized. The region of the numerals has pixel values of 1's and the rest of the 'empty' regions are filled with 0's. The size of the images is different, the width varies from 6 to 95 and height from 18 to 79.

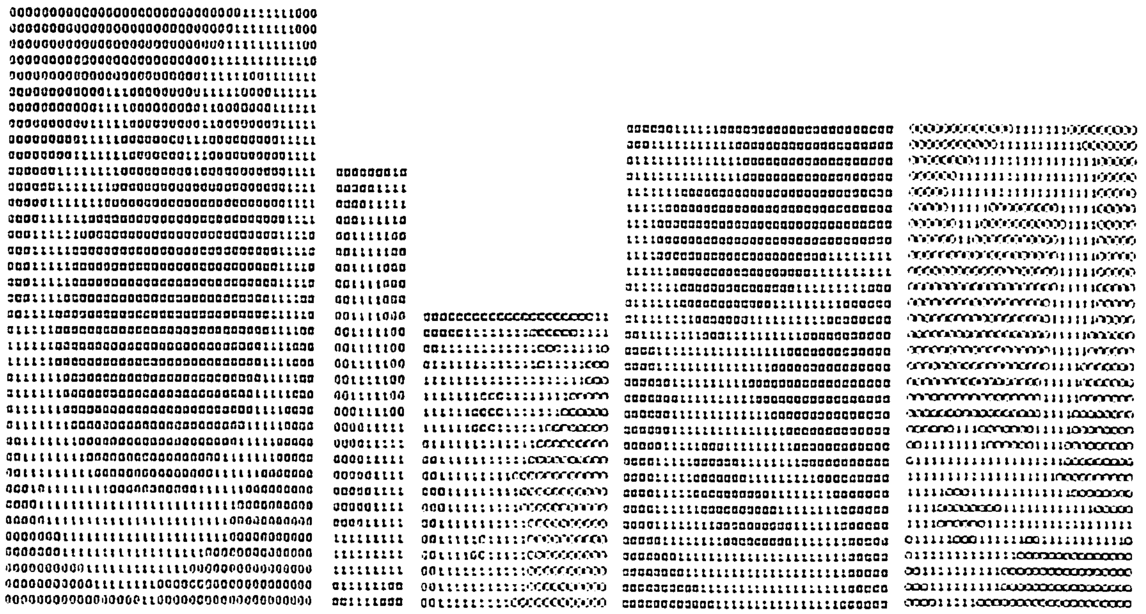


Figure 30. Original Images

First, all images are normalized into the same size 19 X 19. Then these 1000 images are separated into two sets, training data and testing data, each set contains 500 images. The training data, containing 50 images for each numeral, total 500 images are used to create the templates for the statistical method. The testing data, other 500 testing images are used to test the system.

Figure 31 shows the images after size normalization. They are numerals '3', '1', '8' and '9' in size 19X19.

```
0111111111110000000 0011111111111110000 0000000000000000011 0000000001111110000
1111111111111110000 0011111111111111100 0000111111100000111 0000001111111111100
0111111111111111000 0011111111111111100 0111111111110011110 0000111111100111100
0111110001011111000 1111111111111111100 0111111111111111100 0111111000000001100
0000000000011111000 1111111111111111100 1111111111111111100 1111100000000001000
0000000011111111000 1111111111111111100 1111110011111110000 1111000001111111100
0000000111111110000 1111111111111111100 1111100011111100000 1111111111111111110
0000000111110000000 1111111111111111100 1111100111111000000 1111111111000001111
0000000011111000000 1111111111111111100 0111111111110000000 0000100000000001111
0000000000111110000 1111111111111111100 0111111111100000000 0000000000000001111
0000000000001111100 1111111111111110000 0111111110000000000 0000000000000001110
0000000000000001110 1111111111111110000 0011111111000000000 0000000000000011110
0000000000000001110 1111111111111110000 0011111111000000000 0000000000000111110
0000000000000001111 1111111111111110000 0111111111100000000 0000000000000111100
0000000000000011111 1111111111111111100 0111111111100000000 0000000000000111100
0000000000111111110 1111111111111111100 0111101111100000000 0000000000000111100
0000011111111111100 0011111111111111100 0111111111100000000 0000000000000111100
1111111111111100000 0011111111111111111 0111111111100000000 0000000000000111100
0011111000000000000 0000111111111111100 0111111110000000000 0000000000000111100
```

Figure 31. Normalized Into Size 19X19

4.1 *Recognition Without Verification*

In this project, 136 test images are tested by the system. Table 1 lists the test results using structural method without verification. The total recognition rate is 71.32% with a rejection rate of 25.00% and an error rate of 3.68%.

	0	1	2	3	4	5	6	7	8	9	Reject	Total	Recognition Rate	Rejection Rate	Error Rate
0	21										1	22	95.45%	4.55%	0.00%
1		13			2						4	19	68.42%	21.05%	10.53%
2			11								1	12	91.67%	8.33%	0.00%
3				14							0	14	100.00%	0.00%	0.00%
4					11	1					2	14	78.57%	14.29%	7.14%
5						14					0	14	100.00%	0.00%	0.00%
6					1		3				9	13	23.08%	69.23%	7.69%
7								4			1	5	80.00%	20.00%	0.00%
8									4	1	3	8	50.00%	37.50%	12.50%
9										2	13	15	13.33%	86.67%	0.00%
SUM											34	136	71.32%	25.00%	3.68%

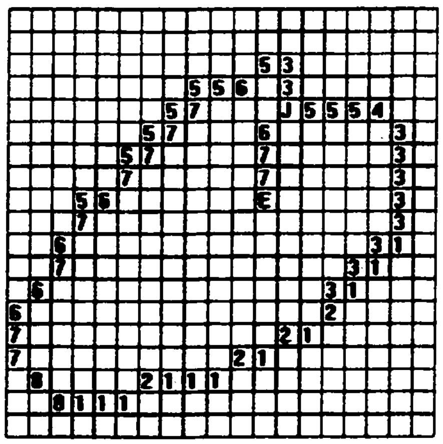
Table 1. Substitution & Recognition Results Using the Structural Method Without Verification

From Table 1 we can find out that numerals '8' and '1' have high error rates, 12.50% and 10.53%. The recognition rate is 50.00% for numeral '8' and 68.42% for numeral '1'. The major reason causing this is that the preprocessing (normalization, filling, and skeletonization) created some noise and spoiled some features. This will be discussed in greater detail in Section 5, Further Improvement.

Another reason causing the low recognition rate is the rejection. For numerals '9'and '6', they have high rejection rates, 86.67% and 69.23%, because of the confusion between the pairs '0' and '6', '4' and '9'. Each pair has some similar features, so the system could not determine the input image's identity when the input image is classified to '0' and '6' with the same confidence levels or classified to '4' and '9' with the same confidence levels. Figure 17 shows the confusion between '4' and '9', and Figure 21 shows the confusion between '0' and '6'. This problem will be solved at the verification stage.

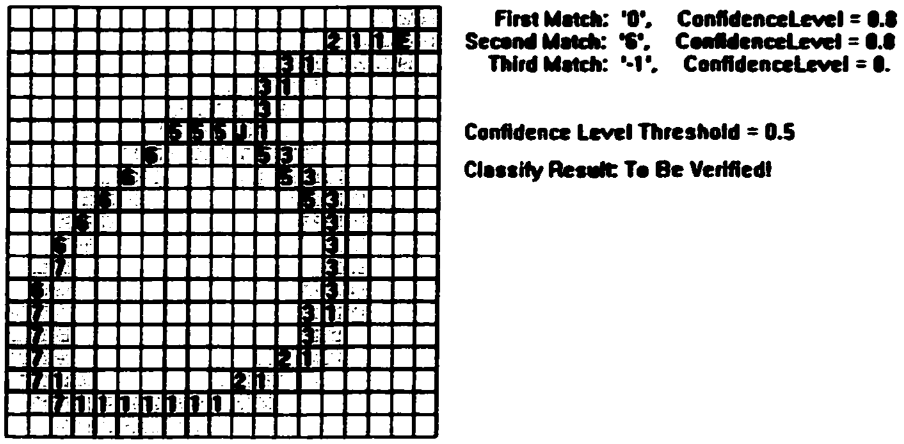
4.1.1 Images Rejected Due To The Confusion Between '0' And '6'

There are total 10 numeral images are rejected due to the confusion between images '0' and '6'. All of them have the similar features: 1 loop, 1 junction, 1 endpoint and the position of the endpoint is on the top part of the image. These images will be identified at the verification stage.



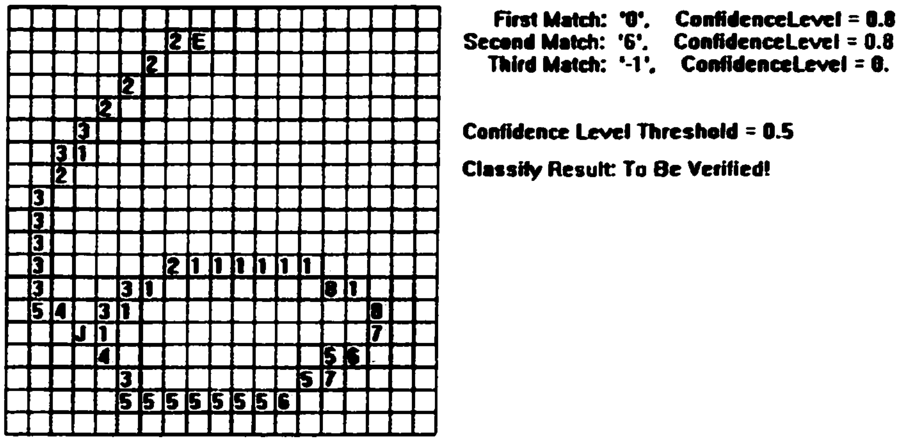
```
   First Match: '0'.  ConfidenceLevel = 0.8
Second Match: '6'.  ConfidenceLevel = 0.8
   Third Match: '-1'.  ConfidenceLevel = 0.

Confidence Level Threshold = 0.5
Classify Result: To Be Verified!
```

Figure 32. Image Rejected Due To The Confusion Between '0' And '6' (1)



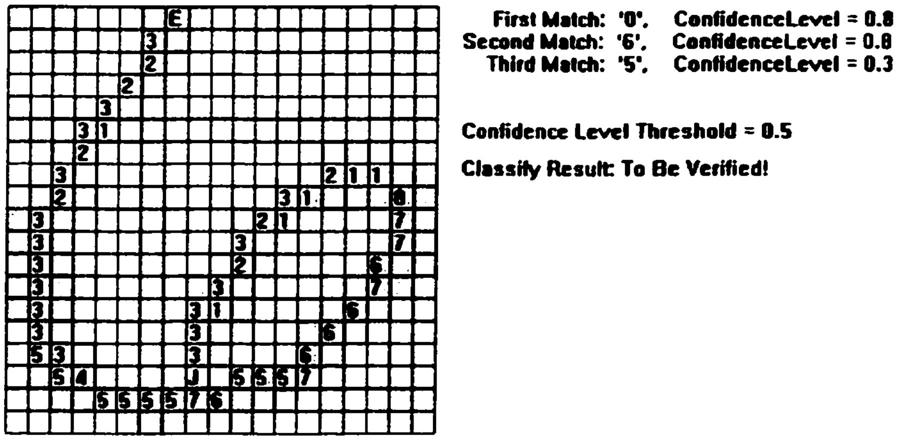
```
First Match: '0'.   ConfidenceLevel = 0.8
Second Match: '6'.   ConfidenceLevel = 0.8
Third Match: '-1'.   ConfidenceLevel = 0.

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!
```

Figure 33. Image Rejected Due To The Confusion Between '0' And '6' (2)



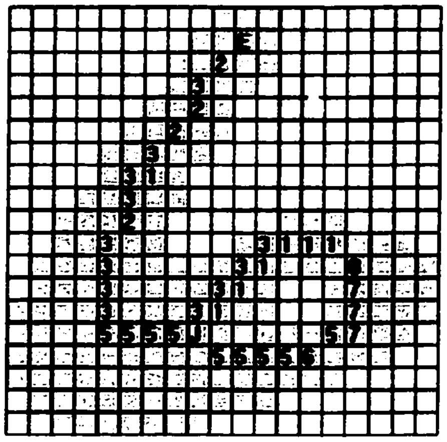
```
First Match: '0'.   ConfidenceLevel = 0.8
Second Match: '6'.   ConfidenceLevel = 0.8
Third Match: '-1'.   ConfidenceLevel = 0.

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!
```

Figure 34. Image Rejected Due To The Confusion Between '0' And '6' (3)



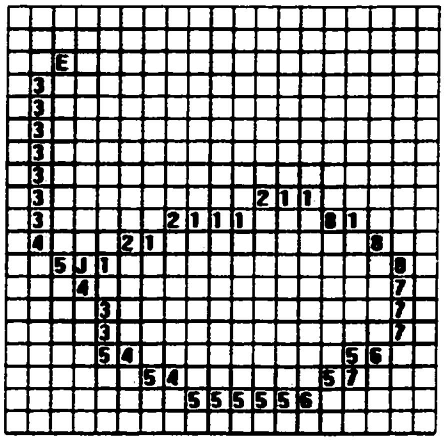
```
First Match: '0'.   ConfidenceLevel = 0.8
Second Match: '6'.   ConfidenceLevel = 0.8
Third Match: '5'.   ConfidenceLevel = 0.3

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!
```

Figure 35. Image Rejected Due To The Confusion Between '0' And '6' (4)



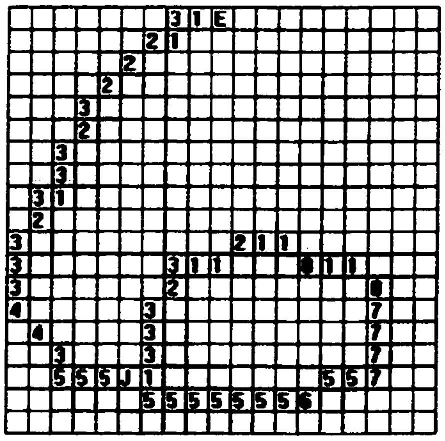
```
  First Match: '0',  ConfidenceLevel = 0.8
Second Match: '6',  ConfidenceLevel = 0.8
  Third Match: '5',  ConfidenceLevel = 0.3

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!
```

Figure 36. Image Rejected Due To The Confusion Between '0' And '6' (5)



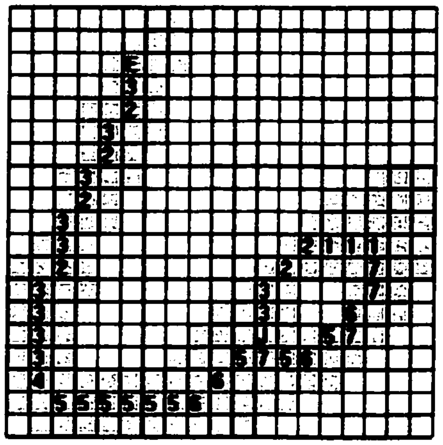
```
  First Match: '0',  ConfidenceLevel = 0.8
Second Match: '6',  ConfidenceLevel = 0.8
  Third Match: '-1',  ConfidenceLevel = 0.

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!
```

Figure 37. Image Rejected Due To The Confusion Between '0' And '6' (6)



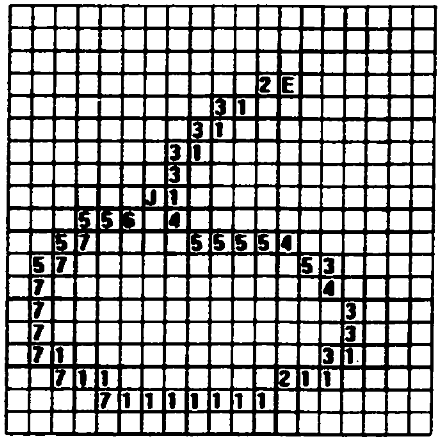
```
  First Match: '0',  ConfidenceLevel = 0.8
Second Match: '6',  ConfidenceLevel = 0.8
  Third Match: '5',  ConfidenceLevel = 0.3

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!
```

Figure 38. Image Rejected Due To The Confusion Between '0' And '6' (7)



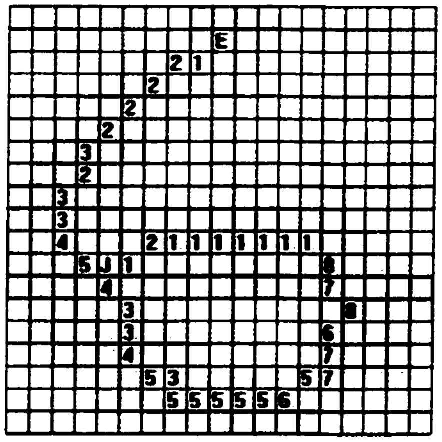
```
 First Match:  '0',  ConfidenceLevel = 0.8
Second Match:  '6',  ConfidenceLevel = 0.8
 Third Match:  '5',  ConfidenceLevel = 0.3

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!
```

Figure 39. Image Rejected Due To The Confusion Between '0' And '6' (8)



```
 First Match:  '0',  ConfidenceLevel = 0.8
Second Match:  '6',  ConfidenceLevel = 0.8
 Third Match:  '-1',  ConfidenceLevel = 0.

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!
```

Figure 40. Image Rejected Due To The Confusion Between '0' And '6' (9)

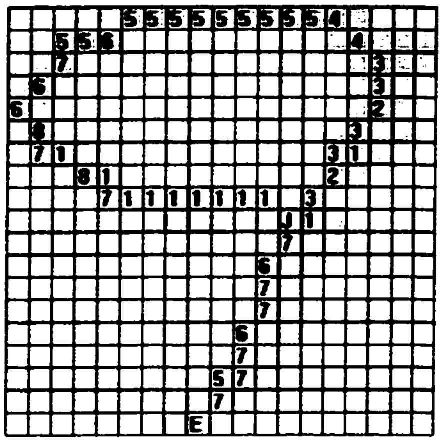


```
 First Match:  '0',  ConfidenceLevel = 0.8
Second Match:  '6',  ConfidenceLevel = 0.8
 Third Match:  '-1',  ConfidenceLevel = 0.

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!
```

Figure 41. Image Rejected Due To The Confusion Between '0' And '6' (10)

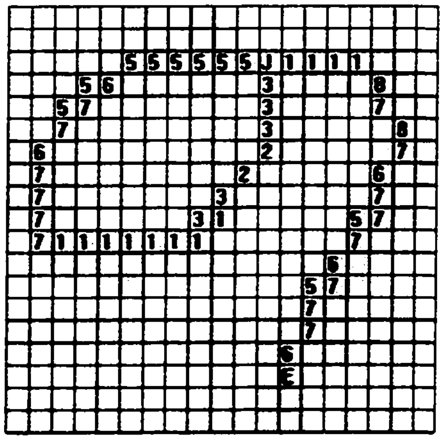
4.1.2 Images Rejected Due To The Confusion Between '4' And '9'

There are total 12 images are rejected due to the confusion between numerals '4' and '9'. All of them have similar features: 1 loop, 1 junction, 1 endpoint and the position of the endpoint is on the bottom part of the image. These images will be identified at the verification stage.



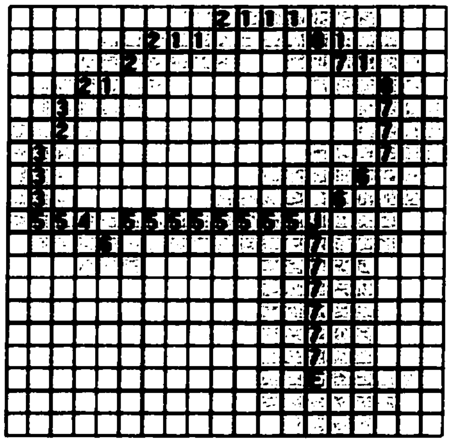
```
 First Match: '4',  ConfidenceLevel = 0.9
Second Match: '9',  ConfidenceLevel = 0.9
 Third Match: '0',  ConfidenceLevel = 0.8

Confidence Level Threshold = 0.5
Classify Result: To Be Verified!
```

Figure 42. Image Rejected Due To The Confusion Between '4' And '9' (1)



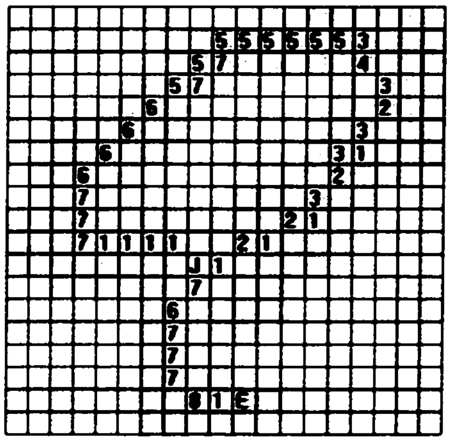
```
 First Match: '4',  ConfidenceLevel = 0.9
Second Match: '9',  ConfidenceLevel = 0.9
 Third Match: '0',  ConfidenceLevel = 0.8

Confidence Level Threshold = 0.5
Classify Result: To Be Verified!
```

Figure 43. Image Rejected Due To The Confusion Between '4' And '9' (2)



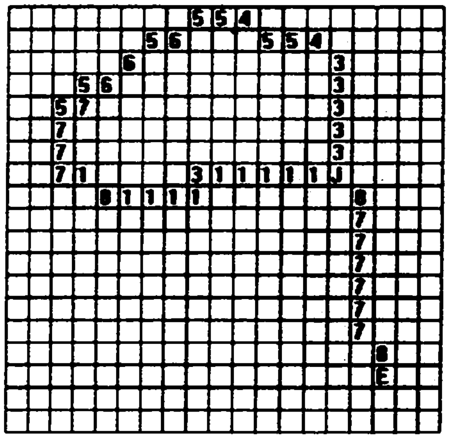
```
  First Match: '4'.   ConfidenceLevel = 0.9
Second Match: '9'.   ConfidenceLevel = 0.9
  Third Match: '0'.   ConfidenceLevel = 0.8

Confidence Level Threshold = 0.5
Classify Result: To Be Verified!
```

Figure 44. Image Rejected Due To The Confusion Between '4' And '9' (3)



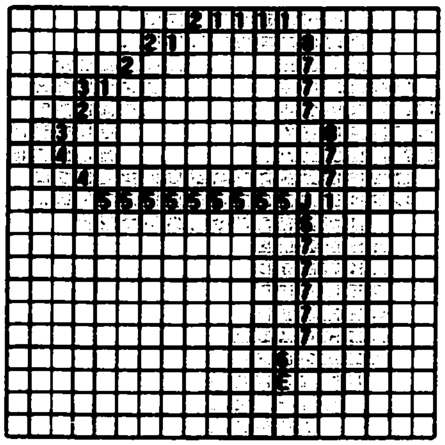
```
  First Match: '4'.   ConfidenceLevel = 0.9
Second Match: '9'.   ConfidenceLevel = 0.9
  Third Match: '0'.   ConfidenceLevel = 0.8

Confidence Level Threshold = 0.5
Classify Result: To Be Verified!
```

Figure 45. Image Rejected Due To The Confusion Between '4' And '9' (4)



```
  First Match: '4'.   ConfidenceLevel = 0.9
Second Match: '9'.   ConfidenceLevel = 0.9
  Third Match: '0'.   ConfidenceLevel = 0.8

Confidence Level Threshold = 0.5
Classify Result: To Be Verified!
```

Figure 46. Image Rejected Due To The Confusion Between '4' And '9' (5)

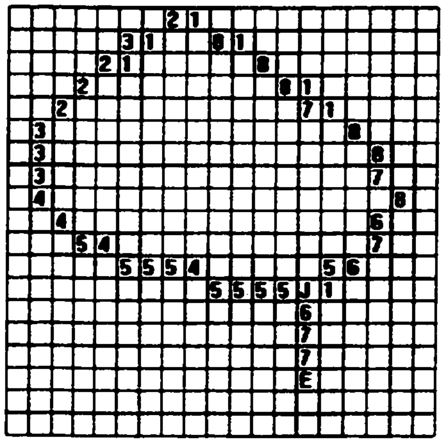


First Match: '4', ConfidenceLevel = 0.9
Second Match: '9', ConfidenceLevel = 0.9
Third Match: '0', ConfidenceLevel = 0.8

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!

Figure 47. Image Rejected Due To The Confusion Between '4' And '9' (6)

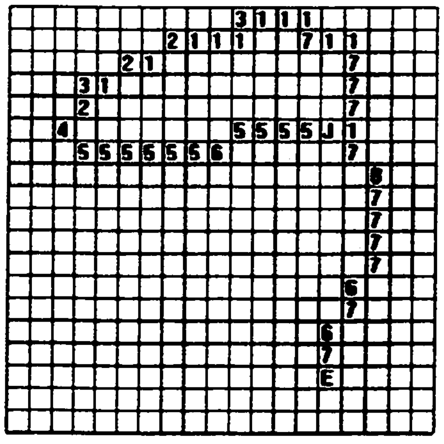


First Match: '4', ConfidenceLevel = 0.9
Second Match: '9', ConfidenceLevel = 0.9
Third Match: '0', ConfidenceLevel = 0.8

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!

Figure 48. Image Rejected Due To The Confusion Between '4' And '9' (7)

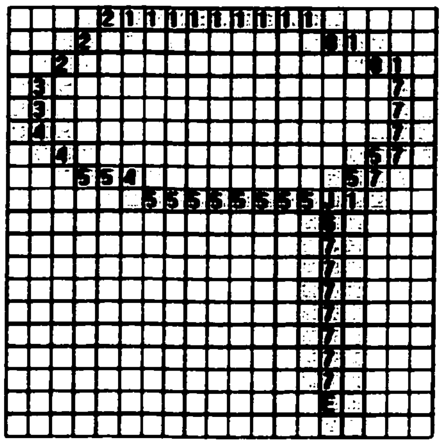


First Match: '4', ConfidenceLevel = 0.9
Second Match: '9', ConfidenceLevel = 0.9
Third Match: '0', ConfidenceLevel = 0.8

Confidence Level Threshold = 0.5

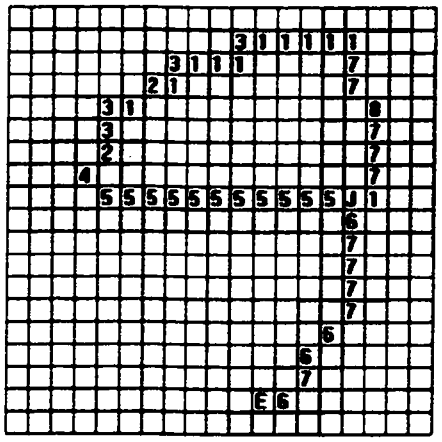
Classify Result: To Be Verified!

Figure 49. Image Rejected Due To The Confusion Between '4' And '9' (8)



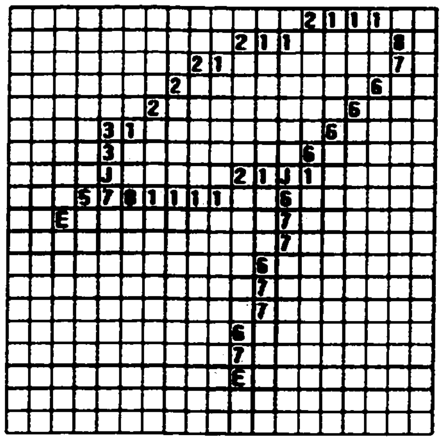
```
  First Match: '4',  ConfidenceLevel = 0.9
 Second Match: '9',  ConfidenceLevel = 0.9
  Third Match: '0',  ConfidenceLevel = 0.8

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!
```

Figure 50. Image Rejected Due To The Confusion Between '4' And '9' (9)



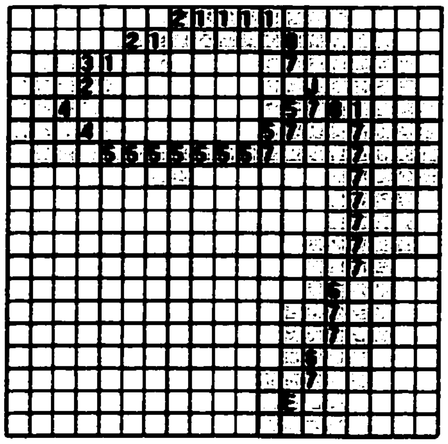
```
  First Match: '4',  ConfidenceLevel = 0.9
 Second Match: '9',  ConfidenceLevel = 0.9
  Third Match: '8',  ConfidenceLevel = 0.8

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!
```

Figure 51. Image Rejected Due To The Confusion Between '4' And '9' (10)



```
  First Match: '4',  ConfidenceLevel = 0.6
 Second Match: '9',  ConfidenceLevel = 0.6
  Third Match: '3',  ConfidenceLevel = 0.5

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!
```

Figure 52. Image Rejected Due To The Confusion Between '4' And '9' (11)

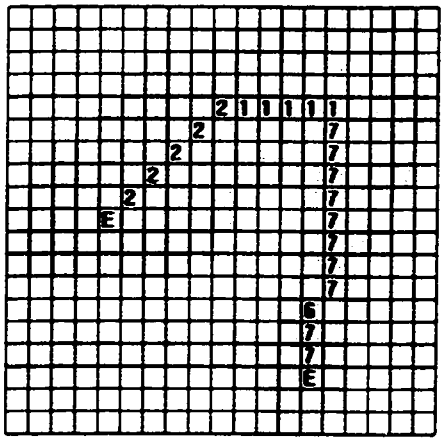


```
    First Match:  '4',   ConfidenceLevel = 0.9
  Second Match:  '9',   ConfidenceLevel = 0.9
    Third Match:  '8',   ConfidenceLevel = 0.8

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!
```

Figure 53. Image Rejected Due To The Confusion Between '4' And '9' (12)

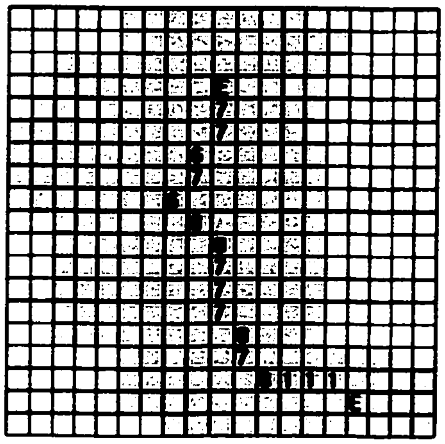
4.1.3 Images Rejected When The Confidence Levels Are Lower Than The Threshold

There are total 9 images are rejected when their confidence levels are lower than the threshold. The reason is that the preprocessing corrupted the original image, for example, a loop was filled in Figure 54. The classifier cannot properly classify the input image based on the features extracted from the corrupted image. These rejected images will be further classified by combining the statistical method.



```
    First Match:  '2',   ConfidenceLevel = 0.4
  Second Match:  '7',   ConfidenceLevel = 0.4
    Third Match:  '4',   ConfidenceLevel = 0.3

Confidence Level Threshold = 0.5

Classify Result: Rejected!
```

Figure 54. Confidence Levels Are Lower Than The Threshold (1)

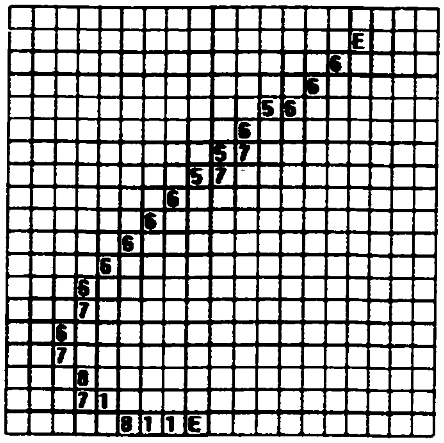


First Match: '2', ConfidenceLevel = 0.4
Second Match: '4', ConfidenceLevel = 0.3
Third Match: '5', ConfidenceLevel = 0.3

Confidence Level Threshold = 0.5

Classify Result: Rejected!

Figure 55. Confidence Levels Are Lower Than The Threshold (2)

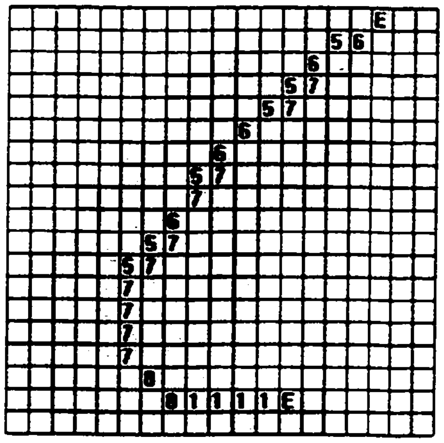


First Match: '2', ConfidenceLevel = 0.4
Second Match: '4', ConfidenceLevel = 0.3
Third Match: '5', ConfidenceLevel = 0.3

Confidence Level Threshold = 0.5

Classify Result: Rejected!

Figure 56. Confidence Levels Are Lower Than The Threshold (3)

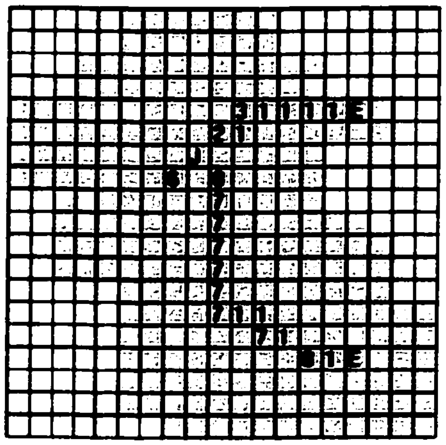


First Match: '2', ConfidenceLevel = 0.4
Second Match: '4', ConfidenceLevel = 0.3
Third Match: '5', ConfidenceLevel = 0.3

Confidence Level Threshold = 0.5

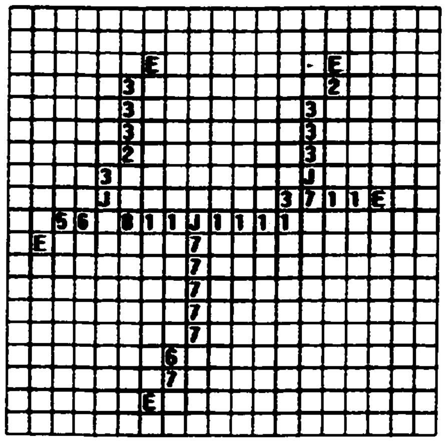
Classify Result: Rejected!

Figure 57. Confidence Levels Are Lower Than The Threshold (4)



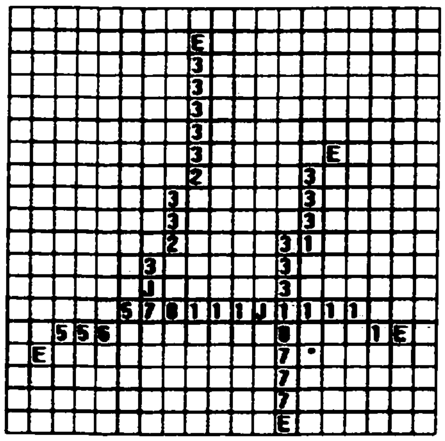
```
   First Match: '2',  ConfidenceLevel = 0.4
  Second Match: '4',  ConfidenceLevel = 0.3
   Third Match: '5',  ConfidenceLevel = 0.3

Confidence Level Threshold = 0.5
Classify Result: Rejected!
```

Figure 58. Confidence Levels Are Lower Than The Threshold (5)



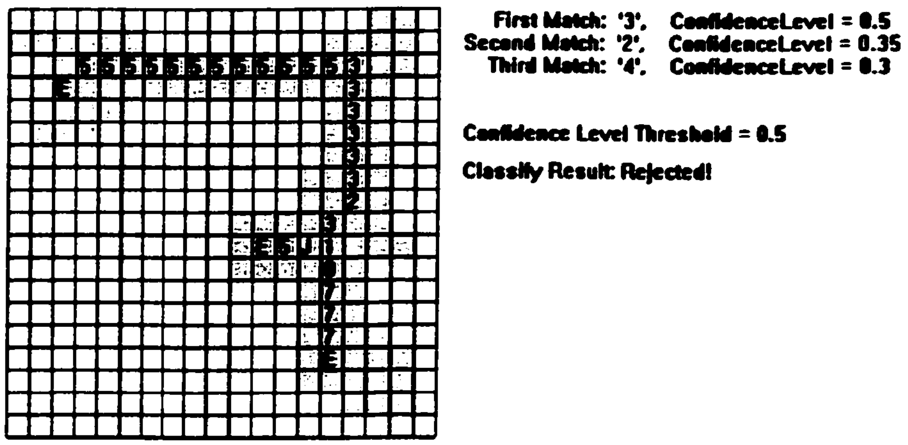
```
   First Match: '-1',  ConfidenceLevel = 0.
  Second Match: '-1',  ConfidenceLevel = 0.
   Third Match: '-1',  ConfidenceLevel = 0.

Confidence Level Threshold = 0.5
Classify Result: Rejected!
```

Figure 59. Confidence Levels Are Lower Than The Threshold (6)



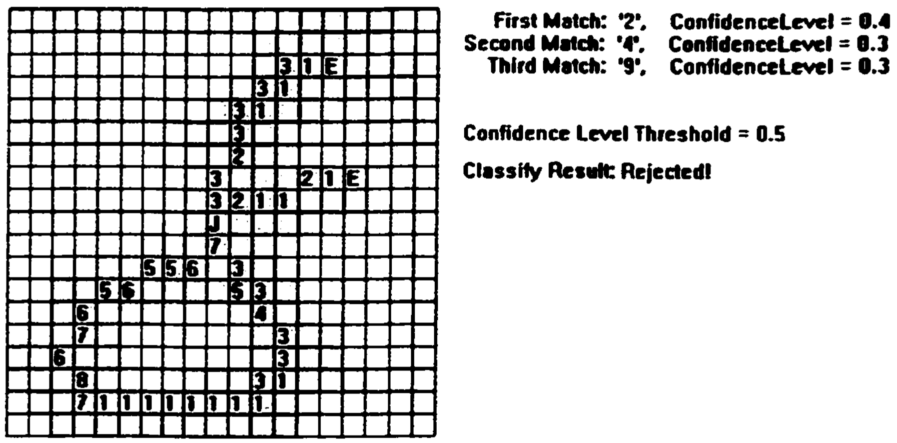
```
   First Match: '-1',  ConfidenceLevel = 0.
  Second Match: '-1',  ConfidenceLevel = 0.
   Third Match: '-1',  ConfidenceLevel = 0.

Confidence Level Threshold = 0.5
Classify Result: Rejected!
```

Figure 60. Confidence Levels Are Lower Than The Threshold (7)



```
   First Match: '3',  ConfidenceLevel = 0.5
  Second Match: '2',  ConfidenceLevel = 0.35
   Third Match: '4',  ConfidenceLevel = 0.3

Confidence Level Threshold = 0.5
Classify Result: Rejected!
```

Figure 61. Confidence Levels Are Lower Than The Threshold (8)

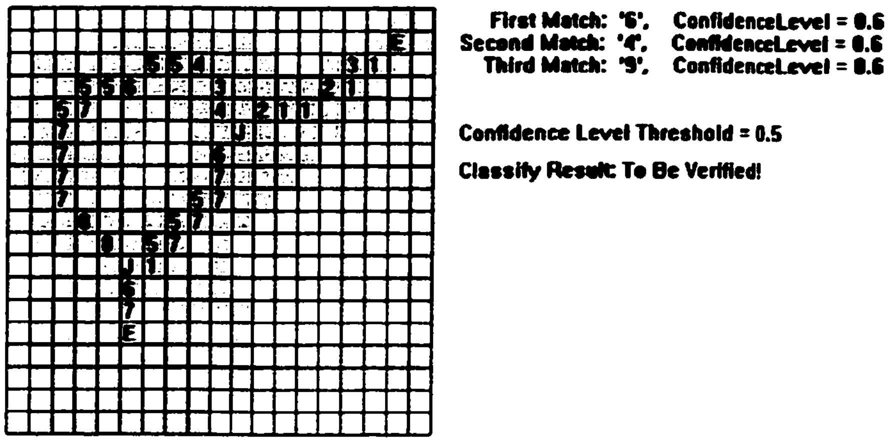


```
   First Match: '2',  ConfidenceLevel = 0.4
  Second Match: '4',  ConfidenceLevel = 0.3
   Third Match: '9',  ConfidenceLevel = 0.3

Confidence Level Threshold = 0.5
Classify Result: Rejected!
```

Figure 62. Confidence Levels Are Lower Than The Threshold (9)

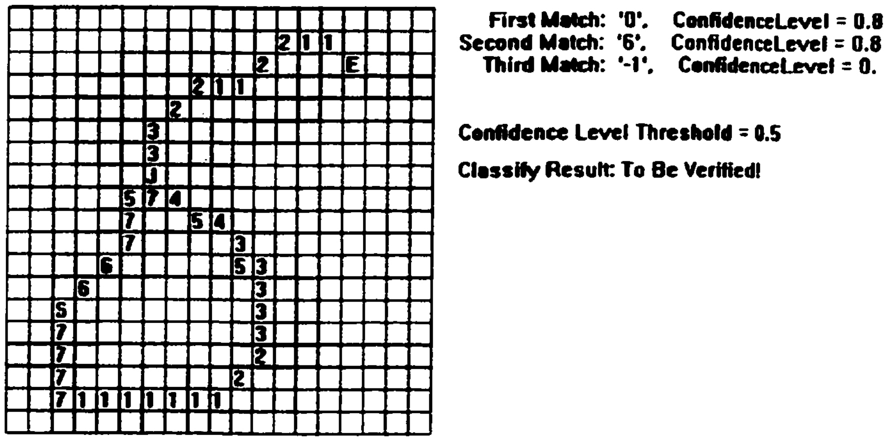
4.1.4 Images Rejected When Two Confidence Levels Are Too Close

There are total 3 numeral images are rejected when the confidence levels of the best two matched numerals are too close. One reason is that the preprocessing corrupted the original image. A loop was filled in Figures 63 and 64. Another reason is that the input image has some defect. There is not a tail in the input numeral '2'. The classifier cannot properly classify these input images based on the features extracted from previous stage. The images in Figure 63 will be further classified by combining the statistical method. The images in Figures 64 and 65 will be further identified in verification stage.



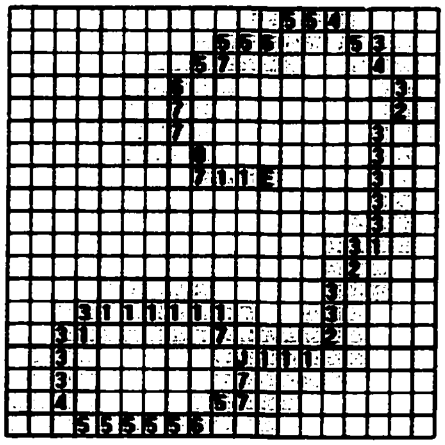
```
   First Match:  '6'.   ConfidenceLevel = 0.6
Second Match:  '4'.   ConfidenceLevel = 0.6
   Third Match:  '9'.   ConfidenceLevel = 0.6

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!
```

Figure 63. Image Rejected When Confidence Levels Are Too Close (1)



```
   First Match:  '0'.   ConfidenceLevel = 0.8
Second Match:  '6'.   ConfidenceLevel = 0.8
   Third Match:  '-1'.   ConfidenceLevel = 0.

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!
```

Figure 64. Image Rejected When Confidence Levels Are Too Close (2)

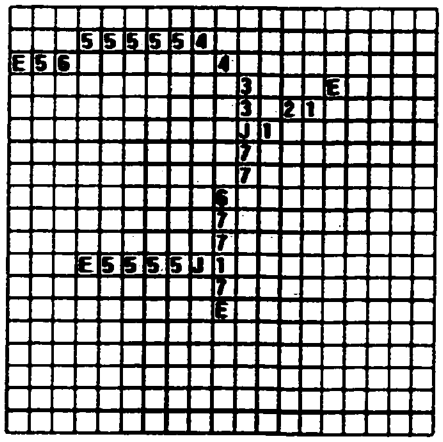


```
  First Match:  '8',   ConfidenceLevel = 0.8
Second Match:  '6',   ConfidenceLevel = 0.8
  Third Match:  '-1',   ConfidenceLevel = 0.

Confidence Level Threshold = 0.5

Classify Result: To Be Verified!
```

Figure 65. Image Rejected When Confidence Levels Are Too Close (3)

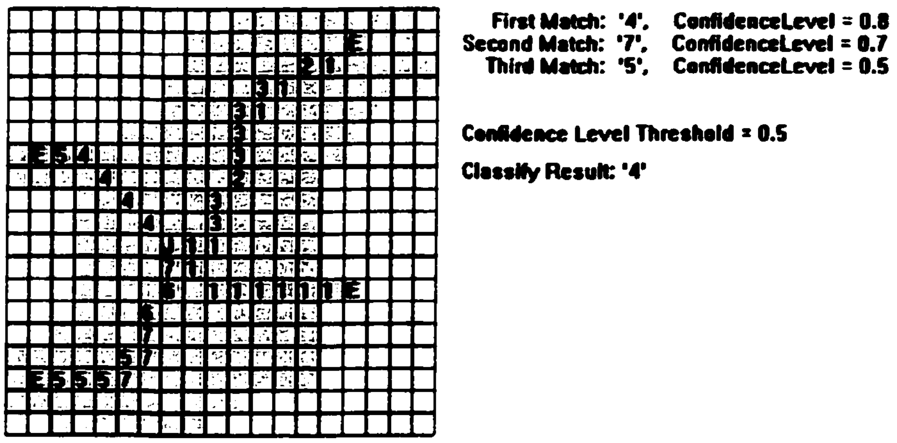
4.1.5 Images Incorrectly Recognized

There are total 5 numeral images that are incorrectly recognized. Figures 66 and 67 show that the input numeral images '1' are recognized to '4' by mistakes. One reason is that preprocessing corrupted the features of original input images. The skeletons do not present the original images anymore. This will be discussed in greater detail in Section 5.2 Improve Normalization. Another reason is that the classifier of numeral '4' should be further improved. It should reject these input images instead of making wrong decision.



```
  First Match:  '7',   ConfidenceLevel = 0.9
Second Match:  '4',   ConfidenceLevel = 0.6
  Third Match:  '5',   ConfidenceLevel = 0.5

Confidence Level Threshold = 0.5

Classify Result: '7'
```

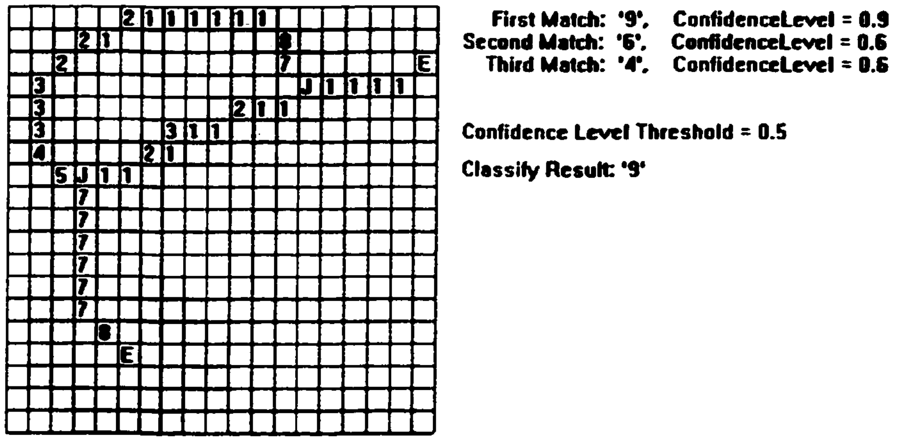
Figure 66. Image Incorrectly Recognized (1)



```
 First Match: '4',  ConfidenceLevel = 0.8
Second Match: '7',  ConfidenceLevel = 0.7
 Third Match: '5',  ConfidenceLevel = 0.5

Confidence Level Threshold = 0.5
Classify Result: '4'
```

Figure 67. Image Incorrectly Recognized (2)

Figures 68 and 69 show that the input numerals '8' and '6' are classified to '9' and '4' by mistakes. That is because that the loop of the input images is filled by the preprocessing. This will be discussed in greater detail in Section 5.2 Improve Normalization.



```
 First Match: '9',  ConfidenceLevel = 0.9
Second Match: '6',  ConfidenceLevel = 0.6
 Third Match: '4',  ConfidenceLevel = 0.6

Confidence Level Threshold = 0.5
Classify Result: '9'
```

Figure 68. Image Incorrectly Recognized (3)

Figure 70 shows that the input numeral image '4' is classified to '5' by mistake. The classifier should be improved and more features should be extracted in the feature extraction stage in order to improve the accuracy of the classification.

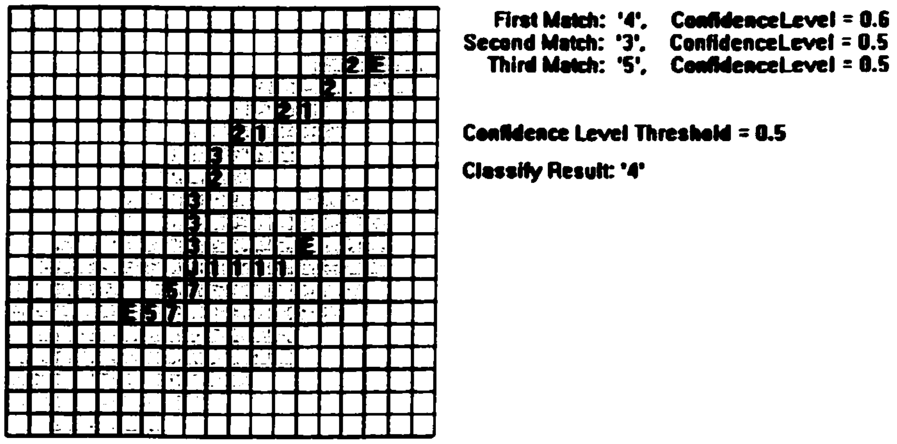

First Match: '4', ConfidenceLevel = 0.6
Second Match: '3', ConfidenceLevel = 0.5
Third Match: '5', ConfidenceLevel = 0.5
Confidence Level Threshold = 0.5
Classify Result: '4'

Figure 69. Image Incorrectly Recognized (4)

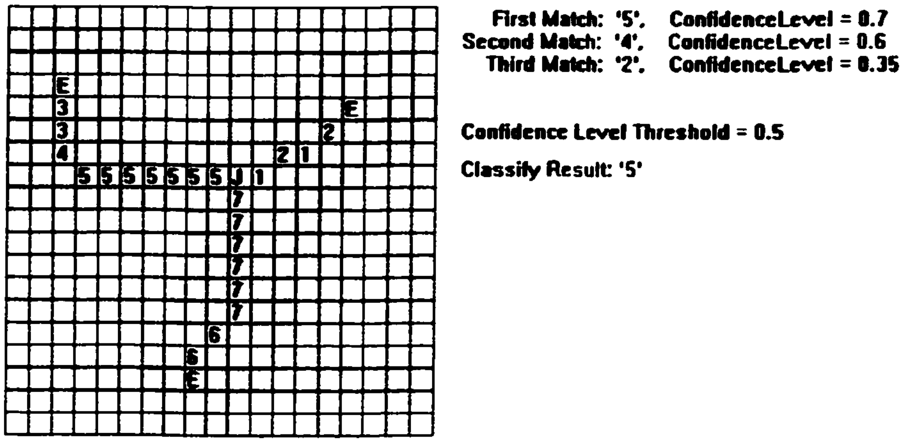

First Match: '5', ConfidenceLevel = 0.7
Second Match: '4', ConfidenceLevel = 0.6
Third Match: '2', ConfidenceLevel = 0.35
Confidence Level Threshold = 0.5
Classify Result: '5'

Figure 70. Image Incorrectly Recognized (5)

4.2 Recognition With Verification

The verification stage is added when we get the same confidence levels for the pairs of '0' - '6' and '4' - '9' to reduce the rejection rate. The results are recorded in Table 2. From this table, we can see that the recognition rate is improved from 13.33% to 93.33% for numeral '9', from 20.08 to 92.31% for numeral '6'. The overall recognition rate is improved from 71.32% to 87.50% with a rejection rate of 7.35% and an error rate of 5.15%.

	0	1	2	3	4	5	6	7	8	9	Reject	Total	Recognition Rate	Rejection Rate	Error Rate
0	22										0	22	100.00%	0.00%	0.00%
1		13			2						4	19	68.42%	21.05%	10.53%
2			11				1				0	12	91.67%	0.00%	8.33%
3				14							0	14	100.00%	0.00%	0.00%
4					11	1					2	14	78.57%	14.29%	7.14%
5						14					0	14	100.00%	0.00%	0.00%
6					1		12				0	13	92.31%	0.00%	7.69%
7								4			1	5	80.00%	20.00%	0.00%
8							1		4	1	2	8	50.00%	25.00%	25.00%
9										14	1	15	93.33%	6.67%	0.00%
SUM											10	136	87.50%	7.35%	5.15%

Table 2. Substitution & Recognition Results Using the Structural Method With Verification

But the low recognition rate for numerals '1' and '8' still cannot be solved in the verification stage. Combining the statistical method with the structural method will improve the recognition rate for these numerals.

4.2.1 Images Rejected Due To The Confusion Between '0' And '6'

All of the 10 input numeral images rejected due to the confusion between '0' and '6' are recognized properly by the verification. From Figure 71 to Figure 80, they show the verification results.

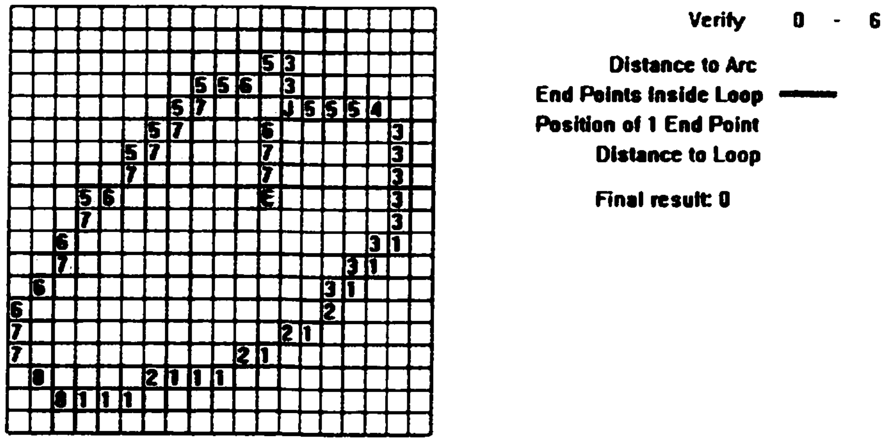


Figure 71. Rejection Solved By Verification 0-6 (1)

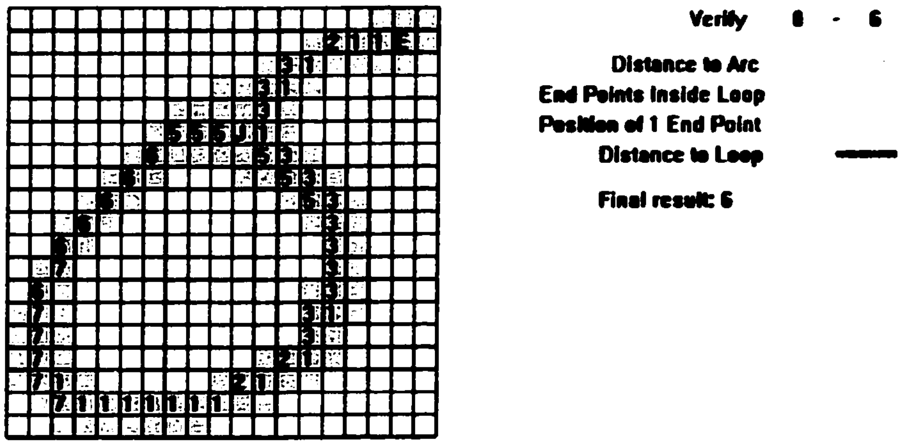


Figure 72. Rejection Solved By Verification 0-6 (2)

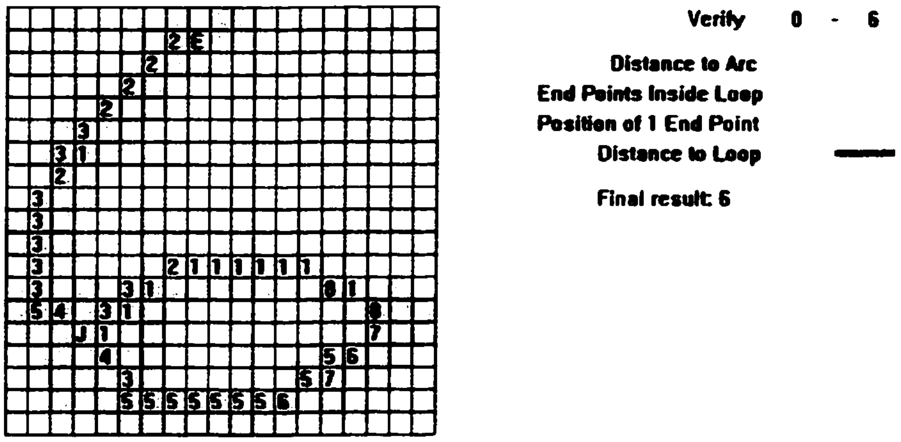


Figure 73. Rejection Solved By Verification 0-6 (3)

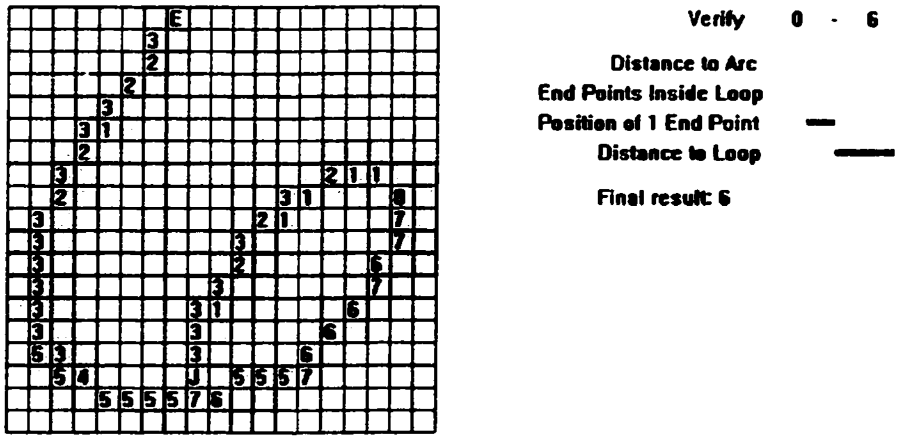


Figure 74. Rejection Solved By Verification 0-6 (4)

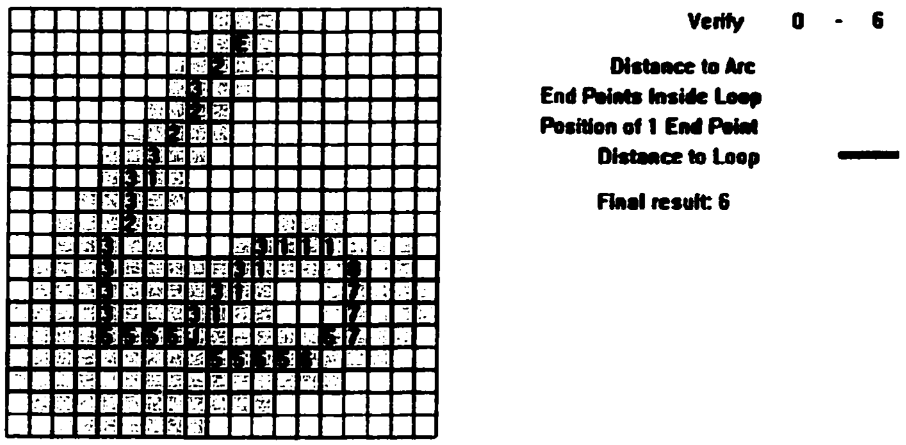


Figure 75. Rejection Solved By Verification 0-6 (5)

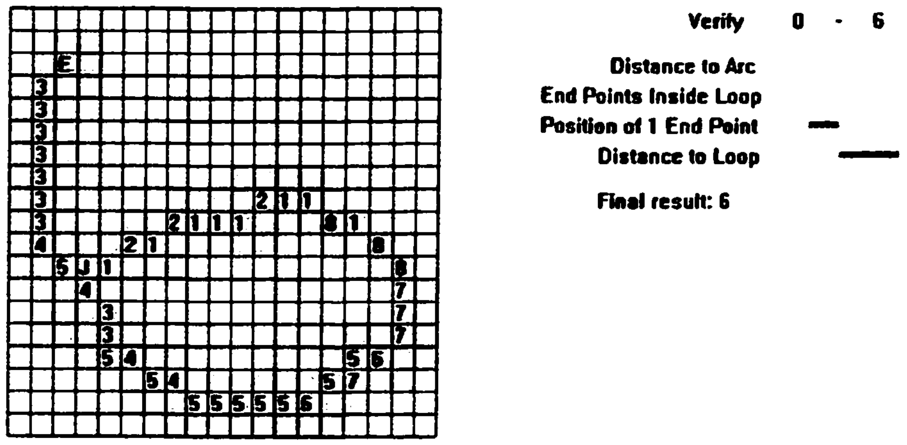


Figure 76. Rejection Solved By Verification 0-6 (6)

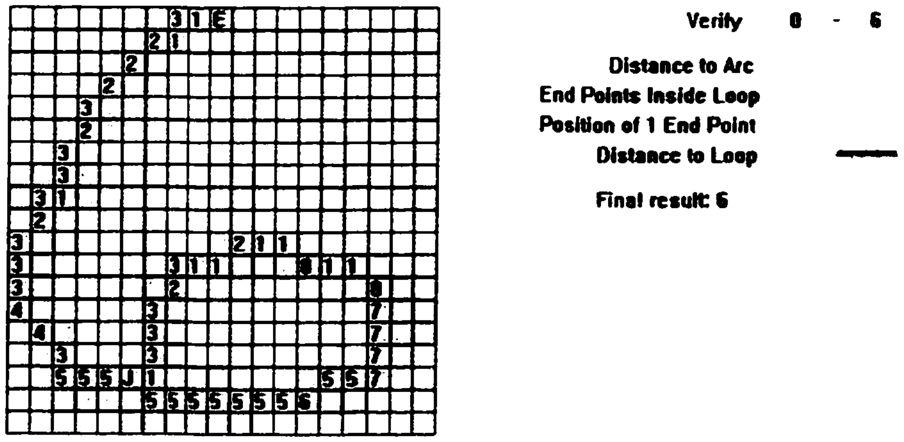


Figure 77. Rejection Solved By Verification 0-6 (7)

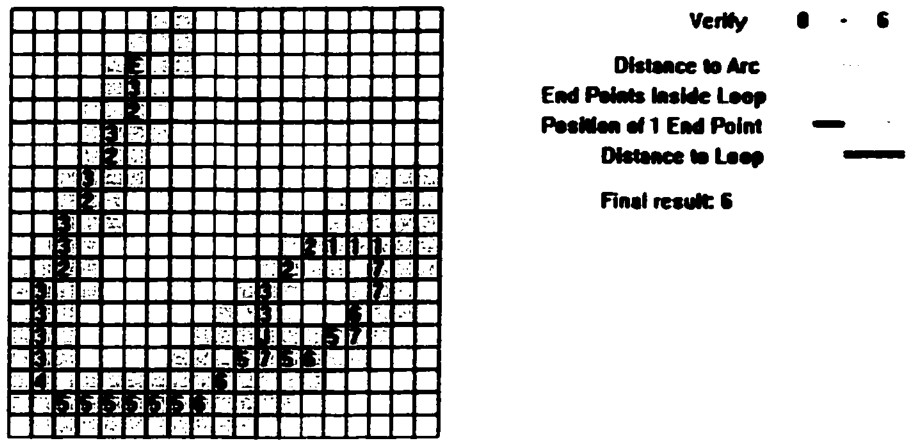


Figure 78. Rejection Solved By Verification 0-6 (8)

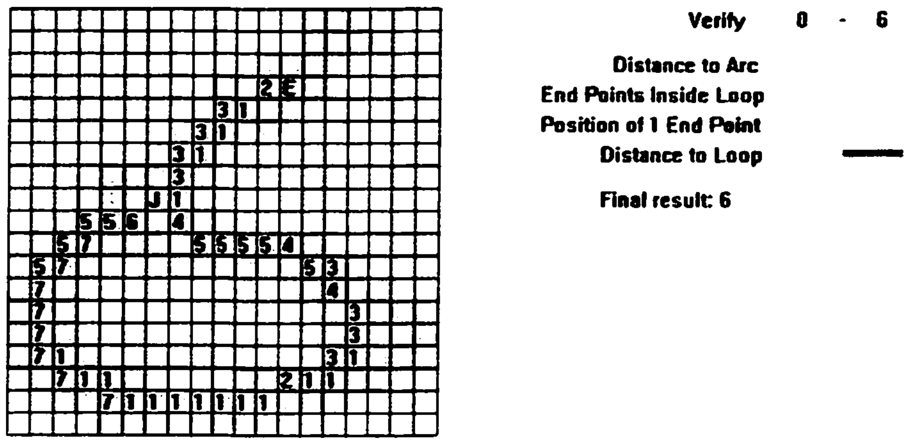


Figure 79. Rejection Solved By Verification 0-6 (9)

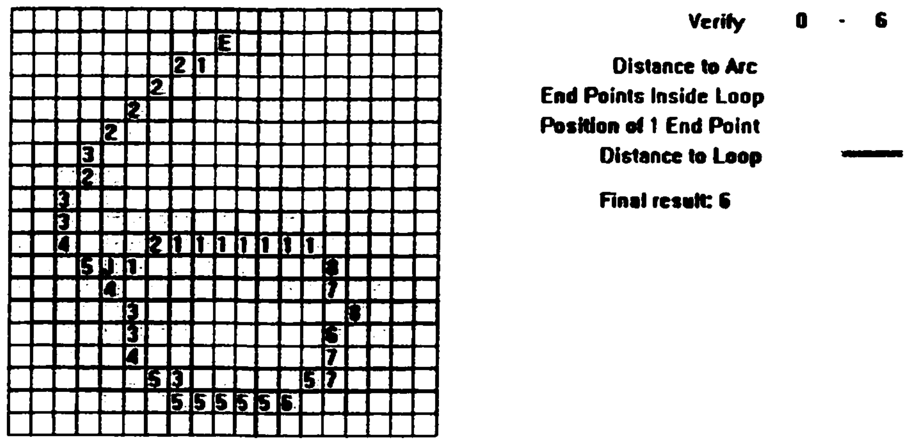
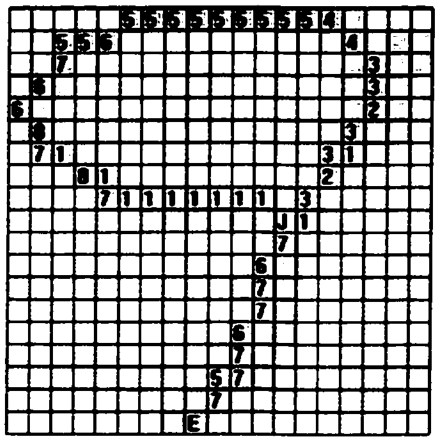


Figure 80. Rejection Solved By Verification 0-6 (10)

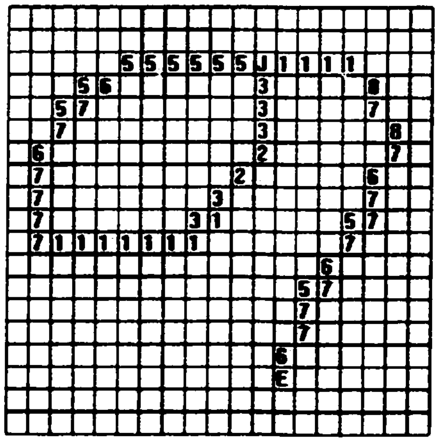
4.2.2 Images Rejected Due To The Confusion Between '4' And '9'

All of the 12 input numeral images rejected due to the confusion between '4' and '9' are recognized properly by the verification. From Figure 81 to Figure 92, they show the verification results.



```
        Verify      4  -  9
distance
top angle                   ———
left angle
right angle                 —
 long tail
      Final result: 9
```

Figure 81. Rejection Solved By Verification 4-9 (1)



```
        Verify      4  -  9
distance
top angle                   ———
left angle
right angle                 —
 long tail
      Final result: 9
```

Figure 82. Rejection Solved By Verification 4-9 (2)

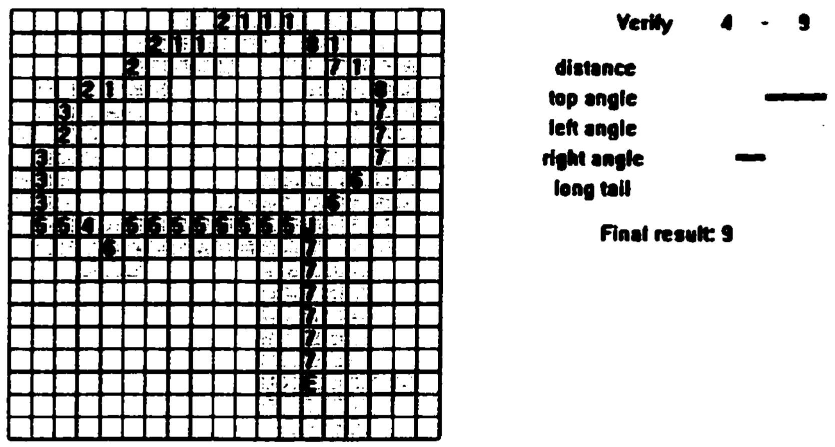


Figure 83. Rejection Solved By Verification 4-9 (3)

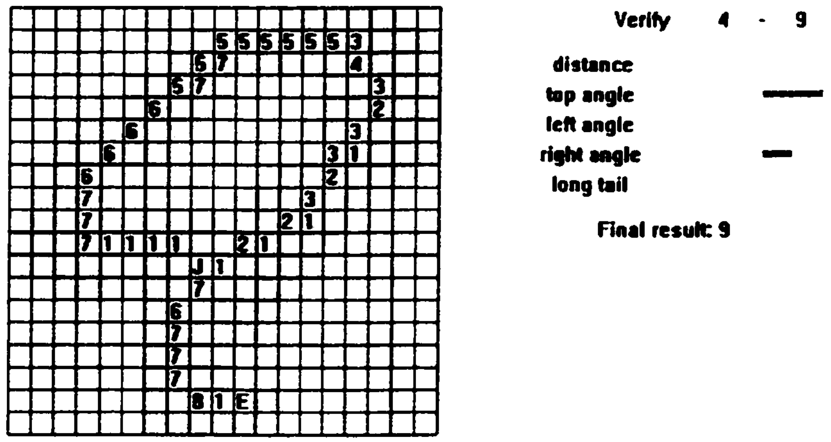


Figure 84. Rejection Solved By Verification 4-9 (4)

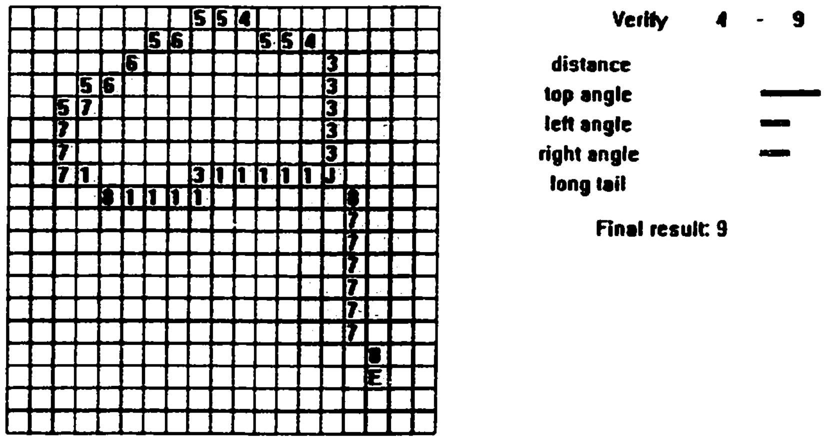


Figure 85. Rejection Solved By Verification 4-9 (5)

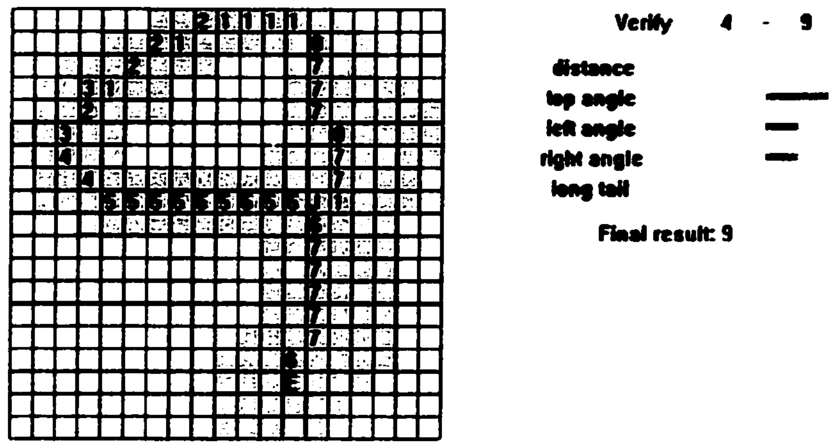


Figure 86. Rejection Solved By Verification 4-9 (6)

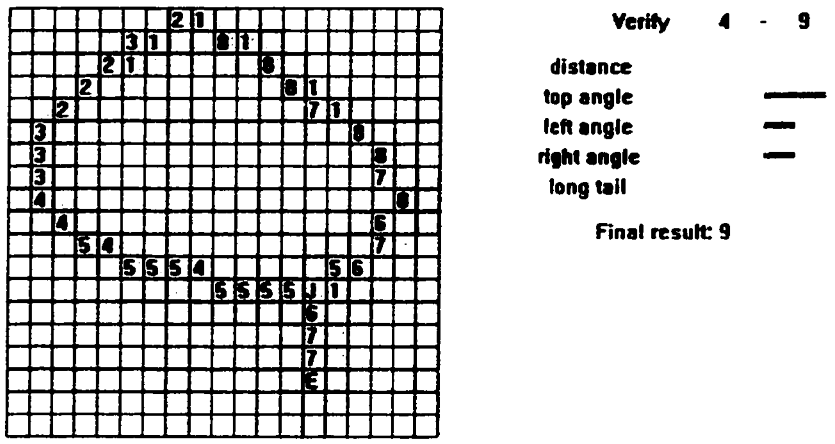


Figure 87. Rejection Solved By Verification 4-9 (7)

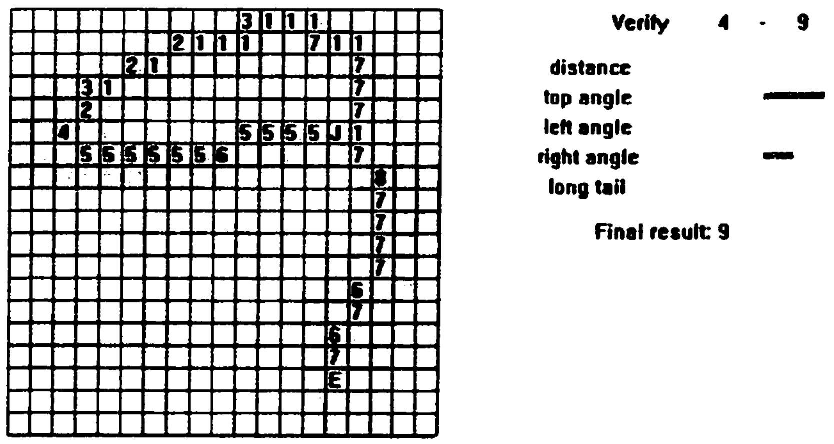


Figure 88. Rejection Solved By Verification 4-9 (8)

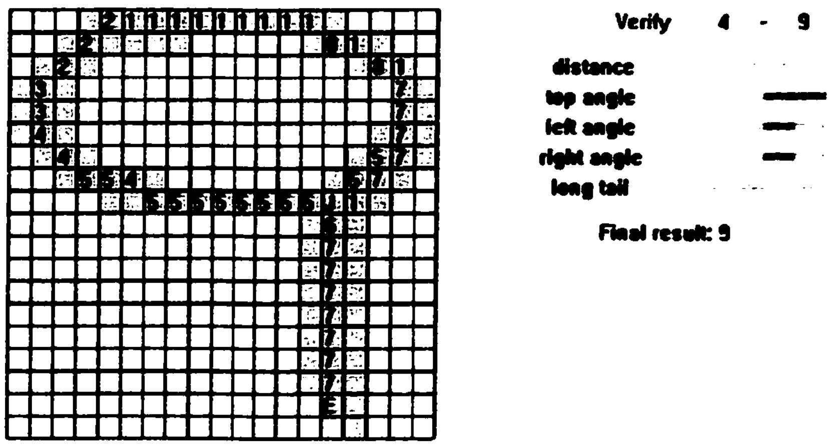


Figure 89. Rejection Solved By Verification 4-9 (9)

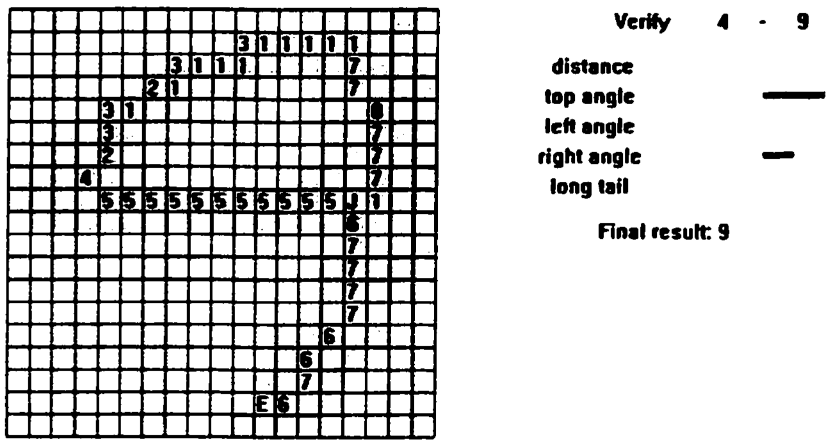


Figure 90. Rejection Solved By Verification 4-9 (10)

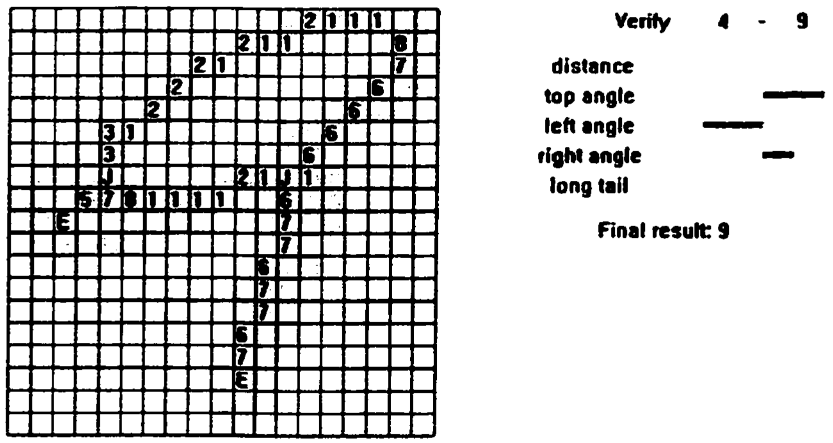


Figure 91. Rejection Solved By Verification 4-9 (11)

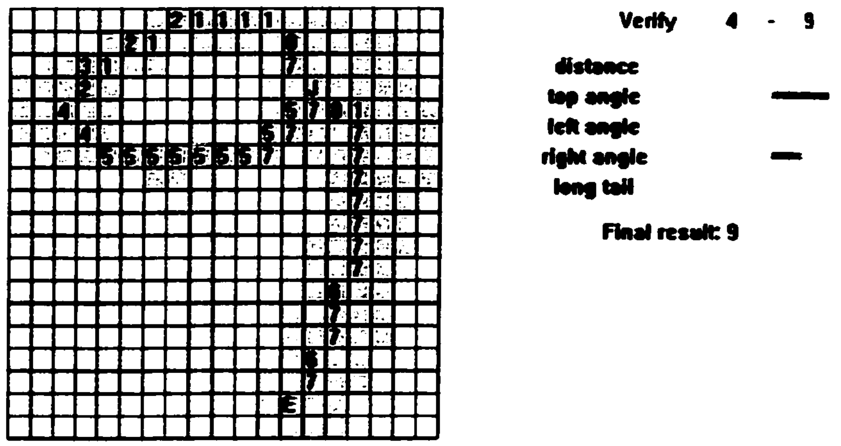


Figure 92. Rejection Solved By Verification 4-9 (12)

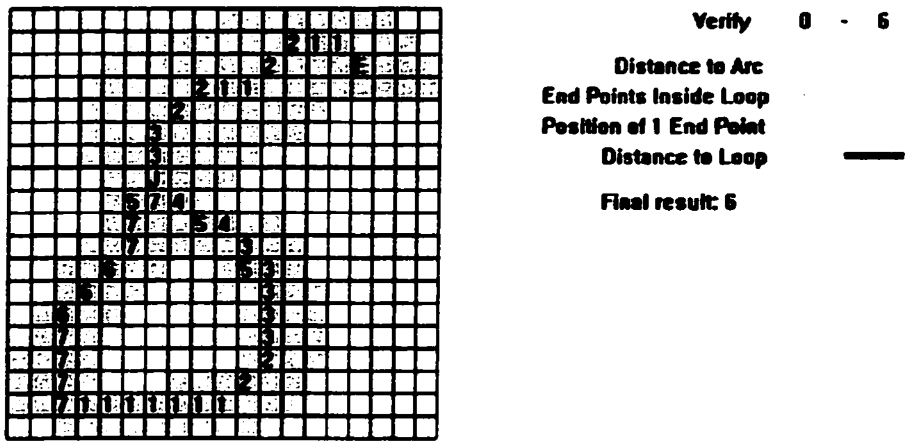
4.2.3 Images Rejected When The Confidence Levels Are Lower Than The Threshold

The images are rejected when the confidence levels are lower than the threshold, they can not identified by the verification stage. They are kept as same as the recognition without verification.

4.2.4 Images Rejected When Two Confidence Levels Are Too Close

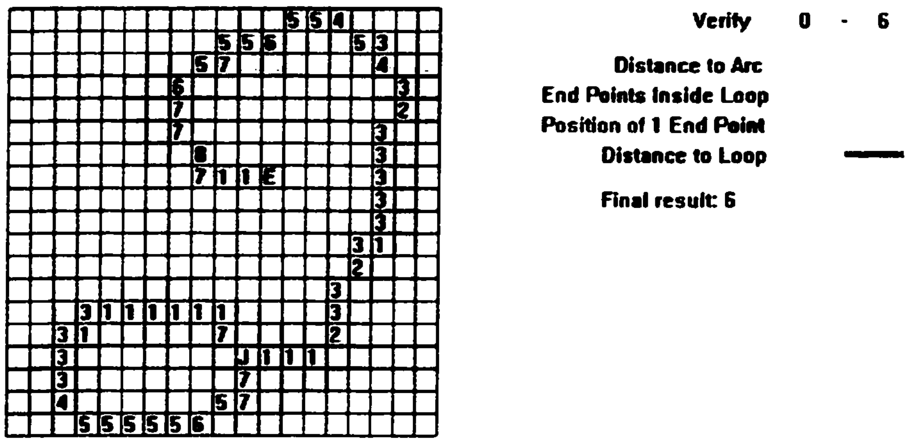
There are 3 numeral images are rejected in the recognition without verification because the best two confidence levels are too close. For the numeral shown in Figure 63 could not be identified at the verification stage because there is no verifier to check the confusion between '4' and '6'. For the numeral '8' showed in Figure 64 is verified to '6'. Figure 93 shows that the recognition result is changed from rejection to error. For the numeral '2' showed in Figure 65 is verified to '6'. Figure 94 shows that the recognition result changed from rejection to error also.

These errors may be corrected by combining the statistical method.



```
                 Verify   0 - 6
          Distance to Arc
  End Points Inside Loop
Position of 1 End Point
        Distance to Loop       ——

           Final result: 6
```

Figure 93. Verification Result (1)



```
                 Verify   0 - 6
          Distance to Arc
  End Points Inside Loop
Position of 1 End Point
        Distance to Loop       ——

           Final result: 6
```

Figure 94. Verification Result (2)

4.2.5 Images Incorrectly Recognized

Images incorrectly recognized in the recognition without verification cannot be corrected in the verification stage. They are kept the same as before in this stage. These recognition results will be re-evaluated and may be corrected by combining the structural method and statistical method.

4.3 *Combining Two Recognition Methods*

It has been observed that the statistical method is more robust than the structural method. Because both of the training data and the testing data are normalized, this will reduce the side effect of the noise created by normalization. Table 3 shows the test results of combining the structural recognition method and the statistical recognition method.

	0	1	2	3	4	5	6	7	8	9	Reject	Total	Recognition Rate	Rejection Rate	Error Rate
0	22										0	22	100.00%	0.00%	0.00%
1		19									0	19	100.00%	0.00%	0.00%
2	1		11								0	12	91.67%	0.00%	8.33%
3				14							0	14	100.00%	0.00%	0.00%
4			1		12					1	0	14	85.71%	0.00%	14.29%
5						14					0	14	100.00%	0.00%	0.00%
6		1					12				0	13	92.31%	0.00%	7.69%
7								5			0	5	100.00%	0.00%	0.00%
8		1							7		0	8	87.50%	0.00%	12.50%
9					1					14	0	15	93.33%	0.00%	6.67%
SUM											0	136	95.59%	0.00%	4.41%

Table 3. Substitution & Recognition Results by Combining Structural Method and Statistical Method

From the above table we can see that the recognition rate is increased from 68.42% to 100.00% for numeral '1', from 50.00% to 87.50% for numeral '8', from 78.57% to 85.71% for numeral '4' and from 80% to 100% for numeral '7'. The overall recognition rate is increased from 87.50% to 95.59% with a rejection rate of 0.00% and an error rate of 4.41%.

But still some numeral images are recognized incorrectly even by combining two recognition methods, structural and statistical.

Figures 95 to 106, show these errors. There are three major reasons:

- The preprocessing corrupted the features of the original image, such as loop was filled. The classifier cannot properly classify the input image based on the features extracted from the corrupted image. This affects the classification in the structural recognition method.

- More features should be extracted. The classifier needs more features in order to classify the input images precisely.
- Need more training data to create better templates for the structure method in order to improve the recognition rate of the statistical method.

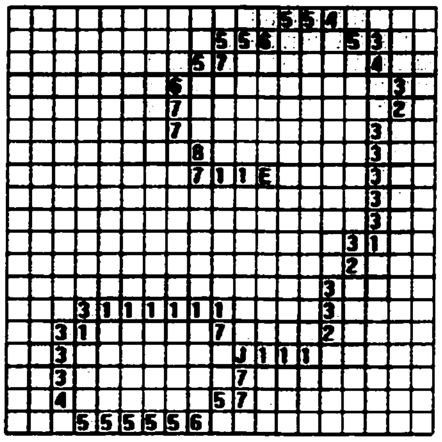


Figure 95. Final Recognition Error (1)

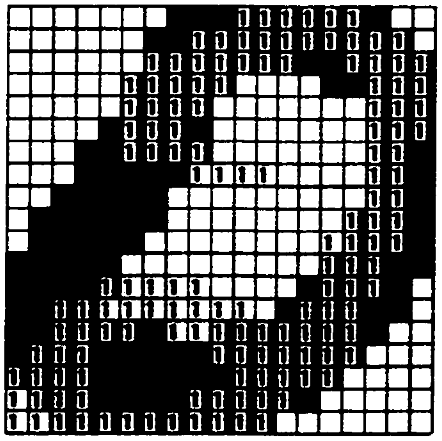
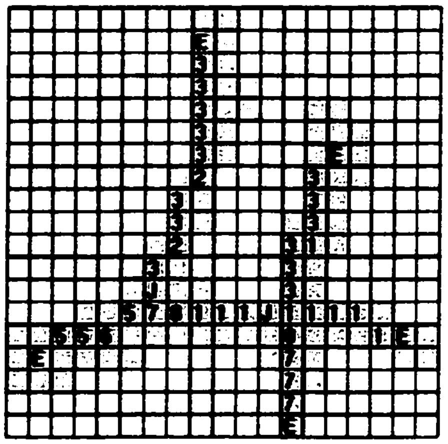
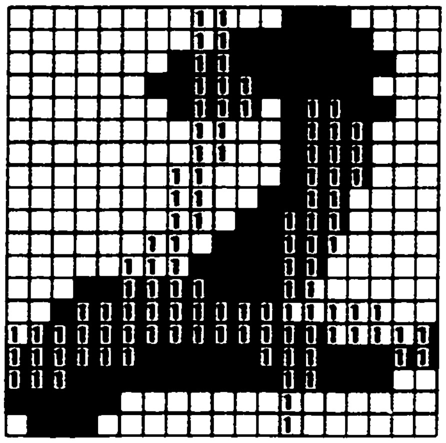

First Match: '0', Similarity = 0.5536
Second Match: '2', Similarity = 0.5487
Third Match: '2', Similarity = 0.5439

Figure 96. Statistical Method Recognition Error (1)



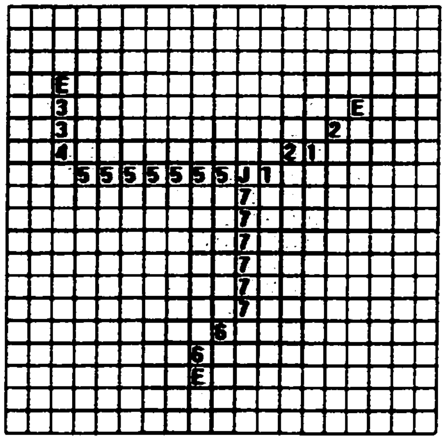
Final Result: '2'

Figure 97. Final Recognition Error (2)



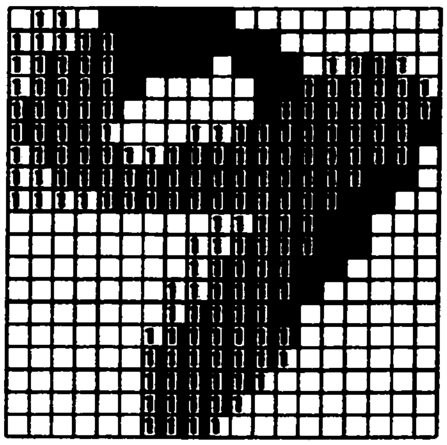
First Match: '2', Similarity = 0.4033
Second Match: '2', Similarity = 0.3967
Third Match: '8', Similarity = 0.3791

Figure 98. Statistical Method Recognition Error (2)



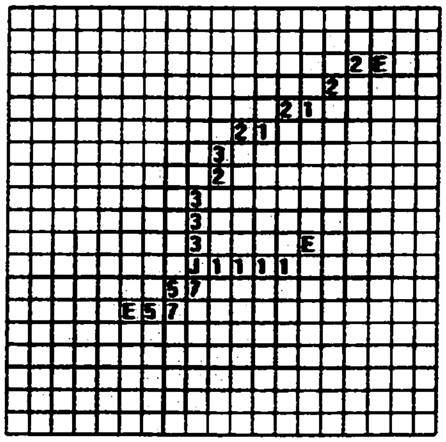
Final Result: '9'

Figure 99. Final Recognition Error (3)



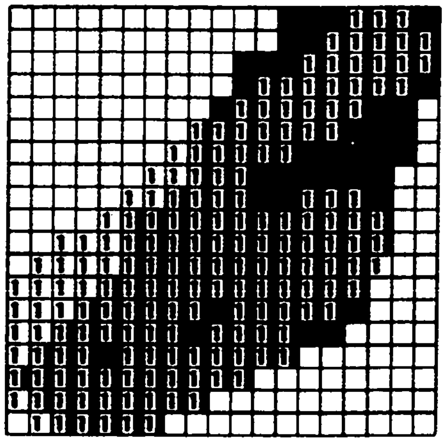
```
 First Match: '9'.  Similarity = 0.6327
Second Match: '9'.  Similarity = 0.5579
 Third Match: '9'.  Similarity = 0.5463
```

Figure 100. Statistical Method Recognition Error (3)



```
Final Result: '1'
```

Figure 101. Final Recognition Error (4)



```
 First Match: '1'.  Similarity = 0.707
Second Match: '1'.  Similarity = 0.6837
 Third Match: '1'.  Similarity = 0.6712
```

Figure 102. Statistical Method Recognition Error (4)

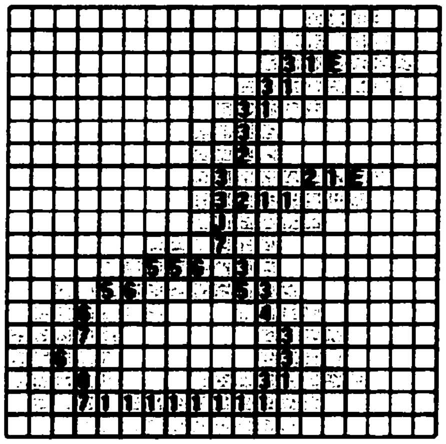


Figure 103. Final Recognition Error (5)

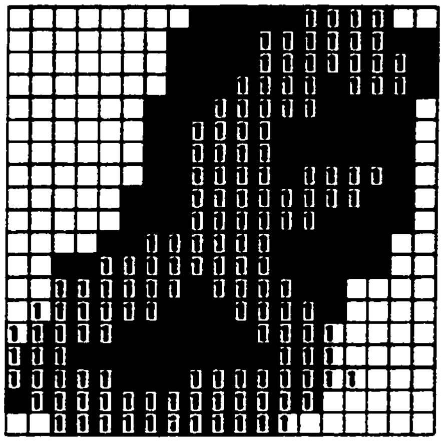

First Match: '1', Similarity = 0.5633
Second Match: '1', Similarity = 0.5592
Third Match: '8', Similarity = 0.5519

Figure 104. Statistical Method Recognition Error (5)

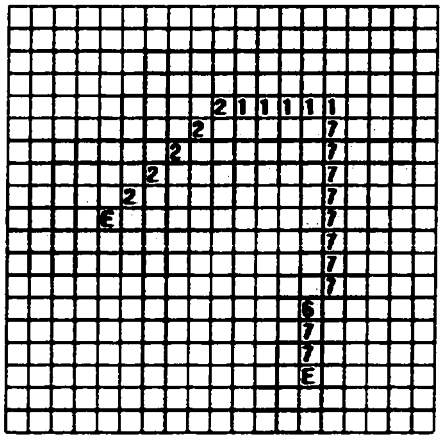


Figure 105. Final Recognition Error (6)

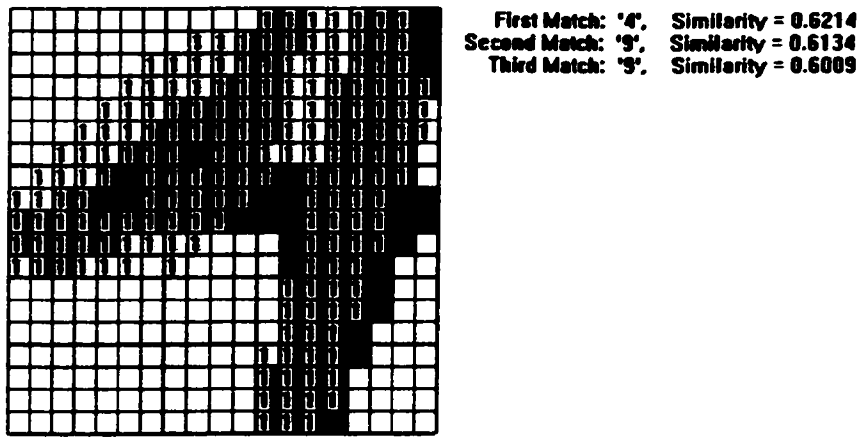

First Match: '4', Similarity = 0.6214
Second Match: '9', Similarity = 0.6134
Third Match: '9', Similarity = 0.6009

Figure 106. Statistical Method Recognition Error (6)

4.4 *Conclusion*

Raw percentage figures providing recognition, substitution and rejection rates of various systems on different databases should NOT be compared without much caution. A system obtaining a higher recognition rate than another is NOT necessarily 'better' and the one obtaining the highest in NOT necessarily 'the best'.

However, from the experiment we can see that the initial recognition rate was 71.32% using the structural method without verification. The verification stage improves the recognition rate from 71.32% to 87.50%, and the combination of the structural method with statistical method further increases the final recognition rate to 95.59%.

Table 4 shows the recognition rates for the individual numerals in three different conditions: recognition without verification, recognition with verification and recognition with combining two methods. Figure 107 shows the classification made a big improvement for the recognition rates of numerals '9' and '6'. The combining two methods further improved the recognition rates of numerals '8', '1', '7' and '4'.

	Recognition Rate	Rejection Rate	Error Rate
Without Verification	71.32%	25.00%	3.68%
With Verification	87.50%	7.35%	5.15%
Combining Two Methods	95.59%	0.00%	4.41%

Table 4. Recognition Rates In Different Cases

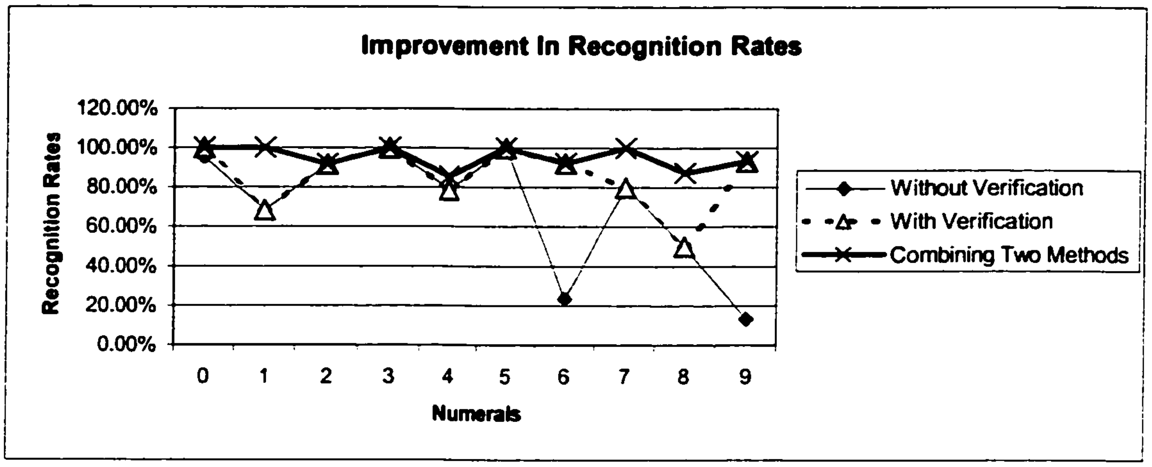


Figure 107. Improvement In Recognition Rates

In conclusion, the verification stage can help the system to distinguish some classes which look quite similar and cannot be distinguished in the classification stage. Combining different recognition methods can further enhance the recognition rate.

5. Further Improvement

The system still can be further improved by using the contour instead of the skeleton and refining the normalization algorithm.

5.1 *Use Contour Instead Of Skeleton.*

We can use the contour instead of the skeleton to avoid the loss of certain features. From the experiment it was observed that the numeral image might have lost some features after preprocessing. In Figure 108, the hole of the loop of numeral '8' becomes smaller after the normalization in size. Filling makes it worse because the holes are completely filled. This will produce mistakes in the feature extraction stage and classification stage.

```
0000000000000000011 0000000001111111100 0000111111111000000 0000000000000000011
0000000000011101110 0000011111111111110 0011111000111100000 0000111111100000111
0000000001111111100 0000111111111111111 0111000000001110001 0111111111110011110
0000000011110111000 0001111101111111111 1110000000001111110 0111111111111111100
0000000011101110000 0001111011111100000 1110000001111110000 1111111111111111100
0000000011011110000 0001111111111000000 1110001111111100000 1111100111111110000
0000000111111100000 0001111111100000000 1111011111000000000 1111100011111100000
0000000111111000000 0000111111000000000 0111111100000000000 1111100111111000000
0000000111100000000 0000011111000000000 0111111000000000000 0111111111110000000
0000001111000000000 0000111111110000000 0011110000000000000 0111111111100000000
0000011111000000000 0001111011111000000 0011110000000000000 0111111111000000000
0001111111000000000 0011111001111000000 0111111000000000000 0011111111000000000
0011100111000000000 0011110001111100000 0111111100000000000 0011111111000000000
0111000011000000000 0111100000111100000 0111111110000000000 0111111111100000000
1110000111000000000 0111000000111100000 0111101111000000000 0111111111100000000
1100000111000000000 1111000000111100000 0111101111000000000 0111101111100000000
1100011110000000000 1111000111111000000 0011111111000000000 0111111111100000000
1111111100000000000 1111111111111000000 0001111110000000000 0111111111100000000
0111111000000000000 0011111111100000000                     0111111111000000000
```

Figure 108. Normalization Makes Hole Smaller

Figure 109 shows that the input image has two loops, but one loop on the top part disappeared after the filling process. And skeletonization destroyed the feature of the loop on the top part.

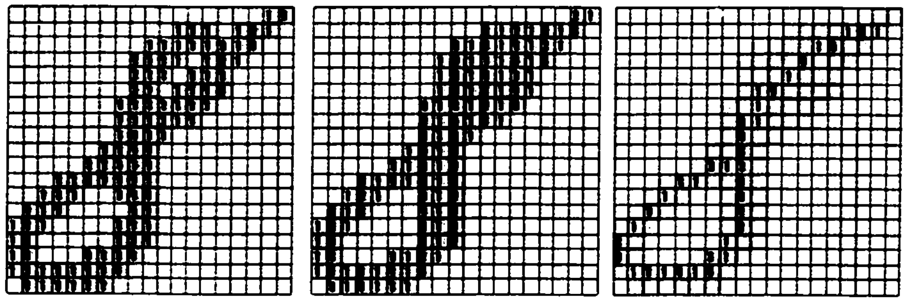
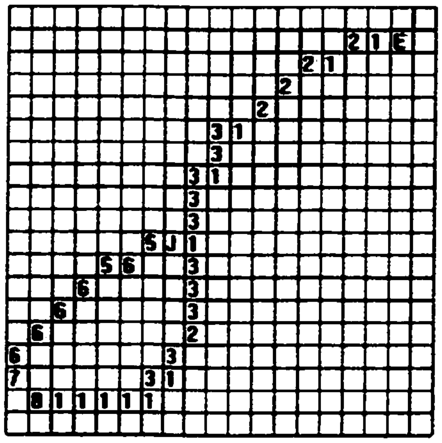


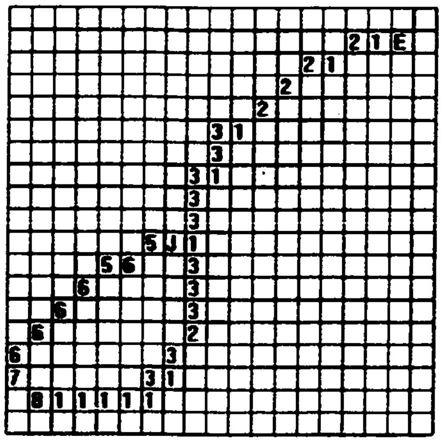
Figure 109. Some Features Lost In Filling And Skeletonization

Figure 110 shows the features extracted from the skeleton. One loop is missing.



```
There are 1 end points:
    [1, 17],
There are 1 Junctions:
    [10, 7],
There are 1 Loop(s)
There are 0 Cup(s)
There are 0 Cap(s)
There are 0 Left-Open-Concavity(s)
There are 0 Right-Open-Concavity(s)
```

Figure 110. Feature Extraction Result Of Numeral '8'

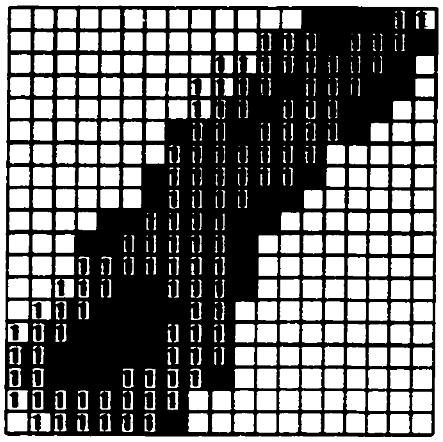


```
 First Match: '6',  ConfidenceLevel = 1.
Second Match: '0',  ConfidenceLevel = 0.8
 Third Match: '-1', ConfidenceLevel = 0.
```

Figure 111. Classification Result Of Numeral '8'

Figure 111 shows that the system could not recognize the target image properly.

Sometimes, this kind of problems could not be solved by statistical methods. Figure 112 shows the system produced the wrong result by using the statistical method.



```
 First Match: '1',   Similarity = 0.6188
Second Match: '1',   Similarity = 0.6114
 Third Match: '1'.   Similarity = 0.5855
```

Figure 112. Statistical Method Could Not Recognize The Numeral Properly

Figure 113 shows that the outer contour can keep the feature of the top loop even the hole has been filled.

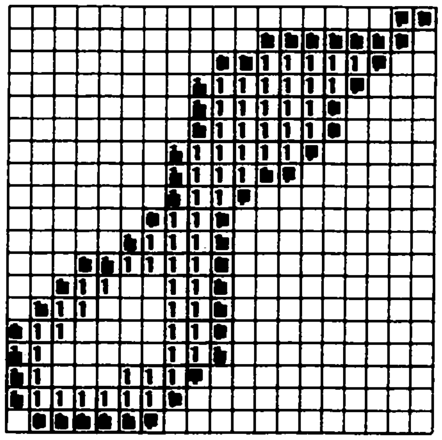


Figure 113. Outer Contour Of Numeral '8'

5.2 *Improve Normalization*

In the normalization process, considering the height-width ratio of the original image can help the image keep some original features. If the numeral images are normalized regardless the height-width ratio of the original, the images may become distorted and noisy. Figure 114 shows the numeral image '1' is distorted after normalizing it from size 7X27 to 19X19.

```
0000111
0001110
0011110
0011110
0111110
0111110
0111110
1111110
1111110
0111111
0111111
0111110
0111110
0111110
0111110
0111110
1111110
1111110
1111100
1111100
0111100
0111100
0111100
0111100
0111100
0111000
0011000
```

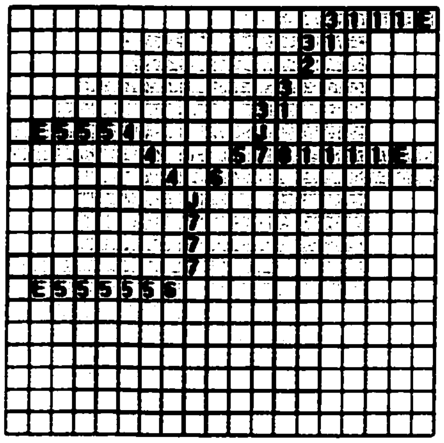
```
0000000000111111111
0000011111111111000
0000011111111111000
0001111111111111000
0011111111111111000
1111111111111111000
1111111111111111111
0001111111111111111
0001111111111111000
0001111111111111000
0001111111111111000
1111111111111111000
1111111111111111000
1111111111111100000
0011111111111100000
0001111111111100000
0001111111111100000
0001111111111100000
0000011111100000000
```

Figure 114. Original Normalization Algorithm

Since the training data are normalized to create the template in the statistical method, the distortion will have a smaller effect on the recognition results. However, it has a greater effect on feature extraction using the structural method.

Figure 115 shows that the numeral image '1' cannot be recognized properly using the structural method after normalization.

In the normalization process, considering the height-width ratio of the original image can help the image keep some original features. Figure 116 shows that the numeral image '1' is normalized to size 19X19 but keeping its original width-height ratio. The features of the original image are maintained.



```
    First Match:  '4',   ConfidenceLevel = 0.8
Second Match:  '7',   ConfidenceLevel = 0.7
   Third Match:  '5',   ConfidenceLevel = 0.6
```

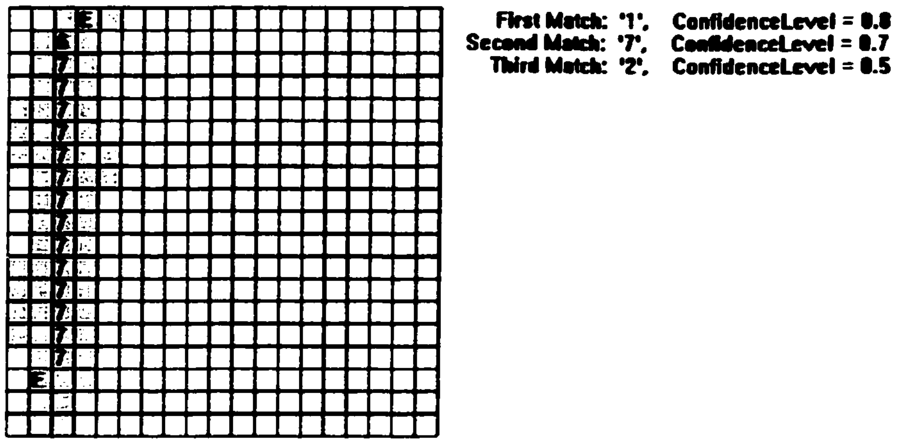
Figure 115. Structural Method Cannot Recognize The Numeral Properly

```
0000111
0001110
0011110
0011110
0111110
0111110
0111110
1111110
1111110
0111111
0111111
0111110
0111110
0111110
0111110
0111110
1111110
1111110
1111100
1111100
0111100
0111100
0111100
0111100
0111100
0111000
0011000
```

```
0011100000000000000
0111000000000000000
0111000000000000000
0111000000000000000
1111000000000000000
1111000000000000000
1111100000000000000
0111100000000000000
0111000000000000000
0111000000000000000
0111000000000000000
1111000000000000000
1111000000000000000
1111000000000000000
1111000000000000000
0111000000000000000
0111000000000000000
0110000000000000000
0110000000000000000
```

Figure 116. Refined Normalization Algorithm

Figure 117 shows that the input image of numeral '1' can be recognized properly using the structural method after refining the normalization algorithm.



```
 First Match:  '1',   ConfidenceLevel = 0.8
Second Match:  '7',   ConfidenceLevel = 0.7
 Third Match:  '2',   ConfidenceLevel = 0.5
```

Figure 117. Structural Method Can Recognize The Numeral Properly

6. References

1. S. Mori, Ching Y. Suen and K. Yamamoto, "Historical Review of OCR Research and Development," Proceedings of the IEEE, vol. 80, pp. 1029-1058, 1992.

2. R. Legault, "Unconstrained Handwritten Numeral Recognition: A Contribution Towards Matching Human Performance," A Thesis in the Department of Computer Science, Concordia University, 1997.

3. L. Lam and Ching Y. Suen, "Structural Classification and Relaxation Matching of Totally Unconstrained Handwritten ZIP-Code Numbers," Pattern Recognition, vol. 21, no. 1, pp. 19-31, 1988.

4. Anil K. Jain, Robert P. W. Duin and Jianchang Mao, "Statistical Pattern Recognition: A Review," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, pp. 4-37, January 2000.

5. Anil K. Jain, Jianchang Mao and K. M. Mohiuddin, "Artificial Neural Networks: a Tutorial," Computer, vol. 29, no. 3, pp. 31-44, 1996.

6. C. Nadal, R. Legault and Ching Y. Suen, "Complementary Algorithms for the Recognition of Totally Unconstrained Handwritten Numerals," Proceedings 10th International Conference on Pattern Recognition, vol. 1, pp. 443-449, 1990.

7. Ching Y. Suen, C. Nadal, R. Legault, T. A. Mai and L. Lam, "Computer Recognition of Unconstrained Handwritten Numerals," Proceedings of the IEEE, vol. 80, no. 7, pp. 1162--1180, July 1992.

8. P. Ahmed and Ching Y. Suen, "Computer Recognition of Totally Unconstrained Handwritten ZIP Codes," Int. J. Pattern Recognition Artif. Intell. vol. 1, no. 1, pp. 1-15, 1987.

9. Christine Nadal and Ching Y. Suen, "Applying Human Knowledge to Improve Machine Recognition of Confusing Handwritten Numerals," Pattern Recognition, vol. 26, no. 3, pp. 381-389, 1993.

10. Y. T. Zhang and Ching Y. Suen, "A Fast Parallel Algorithm for Thinning Digital Patterns," Commun. ACM vol. 27, no. 3, pp. 236-239, 1984.

11. C. Y. Suen, "Handwriting Generation, Perception and Recognition", Acta Psychologica, vol. 54, pp. 295-312, 1983.

12. M. Ahmed and R. K. Ward, "An Expert System for General Symbol Recognition," Pattern Recognition vol. 33, pp. 1975-1988, 2000.

13. R. Legault and Ching Y. Suen, "Optimal Local Weighted Averaging Methods In Contour Smoothing," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 8, August 1997.

14. Ching Y. Suen and Robert J. Shillman, "Low Error Rate Optical Character Recognition of Unconstrained Handprinted Letters Based on a Model of Human Perception," IEEE Transactions on Systems, Man, and Cybernetics, pp. 491-495, June 1977.