VOT 71903

Moment-based Extraction on Handwritten Digits

Project Leader
Jumail Taliba

Researcher
Assoc. Prof. Dr Siti Mariyam Hj. Shamsuddin

Research Assistant
Tan Shuen Chuan

RESEARCH MANAGEMENT CENTER
UNIVERSITI TEKNOLOGI MALAYSIA

2005

Moment-based Extraction on Handwritten Digits

PROJECT REPORT

Authors
Jumail Taliba

RESEARCH MANAGEMENT CENTER
UNIVERSITI TEKNOLOGI MALAYSIA

2005

Acknowledgement

We would like to express our sincere gratitude to Research Management Center (RMC), Universiti Teknologi Malaysia for their approval on our project the Moment-based Extraction on Handwritten Digits under the Short-term Research program. Also, we would like to thank the management and staff of RMC who contribute directly and indirectly to the success of this project.

Abstract

Handwritten digits recognition software have become a highly demand applications to the market. Manufacturing industries as well as post offices are among the users of these applications. In the past few years, several approaches have been used in development of handwritten recognition applications. However, the accuracy of recognition varies between one and another. In this study, the approach of moment-based techniques are employed on handwritten characters.. These include geometric moments, Zernike moments and contour sequence moments. Classification and recognition results are analyzed to determine the necessity of operation thinning when dealing with the moment functions. A Simple Block Segmentation with Moore Tracing Algorithm (SBS & MNTA) is used in image segmentation while Safe-point Thinning Algorithm (SPTA) is applied in image thinning process. Results obtained have shown that operation thinning should be excluded as its deteriorates the recognition accuracy. Contour sequence moments exhibited the highest recognition rate compared to Geometric moments and Zernike moments.

Abstrak

Perisian pengecaman nombor tulisan tangan telah menjadi aplikasi yang mempunyai permintaan yang tinggi di pasaran. Industri pemprosesan dan pejabat pos adalah antara golongan pengguna bagi aplikasi ini. Bebebrapa tahun kebelakangan ini, beberapa pendekatan telah digunakan dalam pembangunan aplikasi pengecaman nombor tulisan tangan. Walau bagaimanapun, ketepatan pengecaman adalah berbeza diantara satu dengan yang lain. Dalam kajian ini teknik berasaskan momen telah digunakan dalam proses pengecaman tulisan tangan. Ini termasuklah momen geometri, momen Zernike dan momen kontur berjujukan. Keputusan pengecaman dianalisa untuk menentukan keperluan perlaksanaan operasi penipisan apabila fungsi momen yang berlainan digunakan. *Simple Block Segmentation with Moore Tracing Algorithm (SBS & MNTA)* digunakan dalam operasi segmentasi imej, proses penipisan imej. Keputusan yang diperolehi menunjukkan bahawa operasi penipisan tidak perlu digunakan kerana ia mengurangkan ketepatan pengecaman. Momen kontor berjujukan menghasilkan keputusan pengecaman yang terbaik berbanding momen geometri dan momen Zernike.

Table of Content

Chapter 1
Introduction

1.1 Research Objectives

i. Develop and apply image segmentation (SBS & MNTA algorithm) and thinning (SPTA algorithm) in evaluation of handwritten digits recognition.
ii. Study the feasibility of operation thinning for different moment functions used.
iii. Evaluate the use of Geometric Moments, Zernike Moments and Contour Sequence Moments in feature extractions of handwritten digits.
iv. Present the result of handwritten digits classification using feature extracted from moment functions afore mentioned with neuro-fuzzy classifier.
v. Generalize an evaluation of moment invariant functions used and present a reference for future research when cope with image moment computation.
vi. Develop a tentative application showing the result (table, graph, chart, etc) of moment feature extraction using Borland Delphi 7.0.

1.2 Research Scope

i. Techniques pre-processing (thinning and segmentation) is implemented and is applied if necessary using Safe-point Thinning Algorithm (SPTA) and Simple Block Segmentation with Moore Neighbor Tracing Algorithm (SBS & MNTA).
ii. Digit images used in evaluations and experiments are 28x28 pixel in *.raw format.
iii. Digit images used are disconnected and unbroken, range between 0 and 9.
iv. Moment functions utilized include Geometric Moments, Zernike Moments and Contour Sequence Moments.

1.3 Report Organization

Chapter 1: Introduction to the project Moment.

Chapter 2: Methodologies used in computation of Geometric Moments, Zernike Moments and Contour Sequence Moments.

Chapter 3: Algorithms and implementation for image pre-processing, feature extraction using moment functions and neuro-fuzzy classification.

Chapter 4: Experiments and result of feature extraction and neuro-fuzzy classification.

Chapter 5: Discussion and Conclusion.

Chapter 2
Research Methodology

2.1 Introduction

The framework of our neuro-fuzzy classification using isolated handwritten digits is presented in Figure 2.1. The intermediate operations involved are summarized as below: image pre-processing, feature extraction using moment functions and neuro-fuzzy training and also classification (recognition).

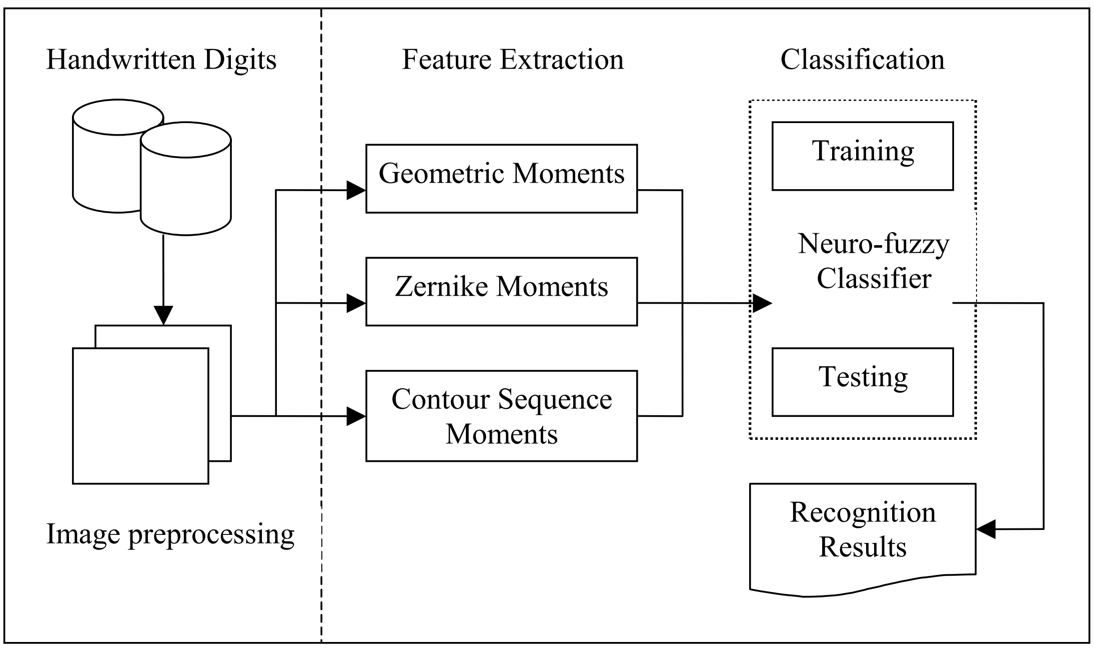


Figure 2.1: A framework of Neuro-fuzzy Classification System.

2.2 Image Pre-processing

In this stage, noise removal, segmentation and thinning are carried out in order to supply a noiseless data for subsequent stage. Median filter is used to smooth the data while keeping the small and sharp details. In this stage, a median value of a set of pixels with $N \times N$ dimension is used to substitute the target processed pixel. Meanwhile, image thresholding is also applied to convert the gray level image into bi-level (black & white) image using threshold value, $\theta = 128$. Safe-point Thinning Algorithm (SPTA) is used in image thinning while new character segmentation is proposed here called Simple Block Segmentation with Moore Neighbor Tracing Algorithm (SBS & MNTA).

2.2.1 Simple Block Segmentation with Moore Neighbor Tracing Algorithm (SBS & MNTA)

SBS & MNTA is utilized in extracting each single character image from a block of image that contains multiple images. The input images is digitized and stored in a matrix denominated original-pixel. Segmentation process of a gray level image that constitutes of 5 steps is described as follows (Gray boundary lines in Figure 2.2.1.1-2.2.1.5 are intentionally added by author to give a better understanding):

Step 1: Block Image Segmentation

This step aims to determine the minimum image area that contain of black pixel for the sake of minimizing the target process boundary. Firstly, scan the input gray level image original-pixels from left to right and from top to bottom, search for the left most coordinate-x, right most coordinates-x, upper most coordinate-y and lower most coordinate-y value of the image that contained black pixel. These 4 values denominate as minX, maxX, minY and maxY. Figure 2.2.1.1 illustrates the result of a sample image after go through Step 1.

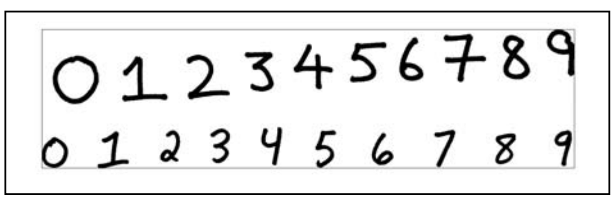


Figure 2.2.1.1: Image produced after Block Image Segmentation.

Step 2: Row Image Segmentation

At this stage, the image boundary that delimited in Step 1 is analyzed again to separate the block image that possible of containing multiple row of digit image. Scan the image original-pixels from left to right and from top to bottom within the range of minX, maxX and minY, maxY, search for the first horizontal line with black pixel (which is the upper most coordinate-y) and also last horizontal line with black pixel (which is the lower most coordinate-y) for each row of character image. Figure 2.2.1.2 illustrates the result of a sample image after go through Step 2.

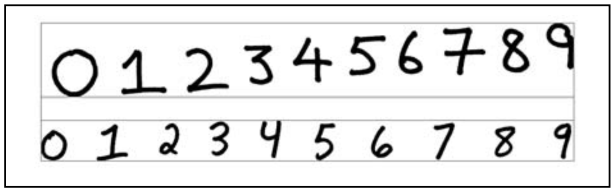


Figure 2.2.1.2: Image produced after Row Image Segmentation.

Step 3 & 4: Single Character Image Segmentation

In Step 3, the width for each digit image is determined by testing and verifying area minimum that contained of black pixels. For each row of character image and within the upper most coordinate-y and lower most coordinate-y, scan the image original-pixels from top to bottom and from left to right, search for the first vertical line with black pixel (denominates as character_minX) and also last vertical line with black pixel (denominates as character_maxX) for each single character image. Figure 2.2.1.3 illustrates the result of a sample image after go through first stage in Single Character Image Segmentation.

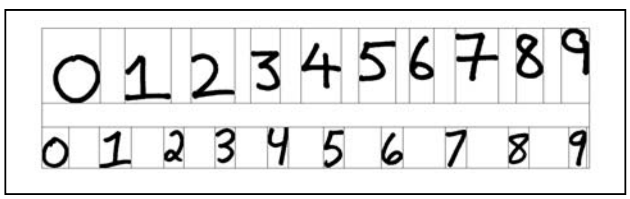


Figure 2.2.1.3: Image produced after first stage in Single Character Image Segmentation.

After the width for each digit image is obtained, Step 4 is carried out with the objective to find out the height for each digit image. For each single character image and within the range of character_minX, character_maxX and upper most coordinate-y, lower most coordinate-y, scan the image original-pixels from left to right and from top to bottom, look for the first upper most coordinate-y that contain black pixel (denominates as characater_minY) and lower most coordinate-y that contain black pixel (denominates as character_maxY). This step is repeated until the entire character row is processed. Figure 2.2.1.4 illustrates the result of a sample image after go through Step 4.

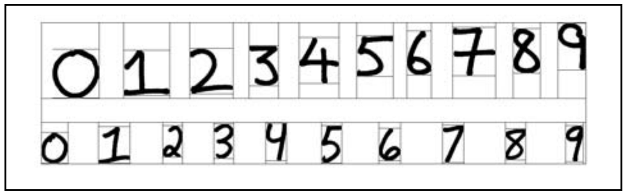


Figure 2.2.1.4: Image produced after second stage in Character Image Segmentation.

Step 5: Image Comparison using Moore Neighbor Tracing Algorithm

Subsequently, Step 5 is conducted in order to separate the image that possible of still containing more than 1 digit. Each character image extracted is contour traced using Moore Neighbor Tracing Algorithm. The image produced is compared with the width and height of the extracted image. If the image produced after operation of contour tracing is same with the extracted image, this mean that the extracted image only contain single character image. Otherwise, the extracted image contains more than one character image and need to be extracted again using Moore Neighbor Tracing Algorithm. Generally, images that are boundary-overlapped need to go through Step 5 as they most probably contain more than one character image. Figure 2.2.1.5 illustrates a sample of boundary-overlapped character image.

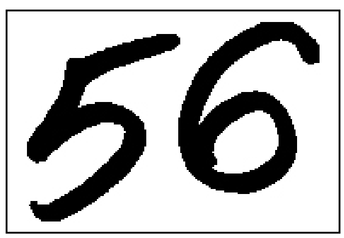


Figure 2.2.1.5: Sample of boundary-overlapped numerical characters image.

Simple Block Segmentation that combines with Moore Neighbor Tracing Algorithm terminates after completely processing and verifying the entire character images contain only single digit image. Figure 2.2.1.6 illustrates the operation involve in each step.

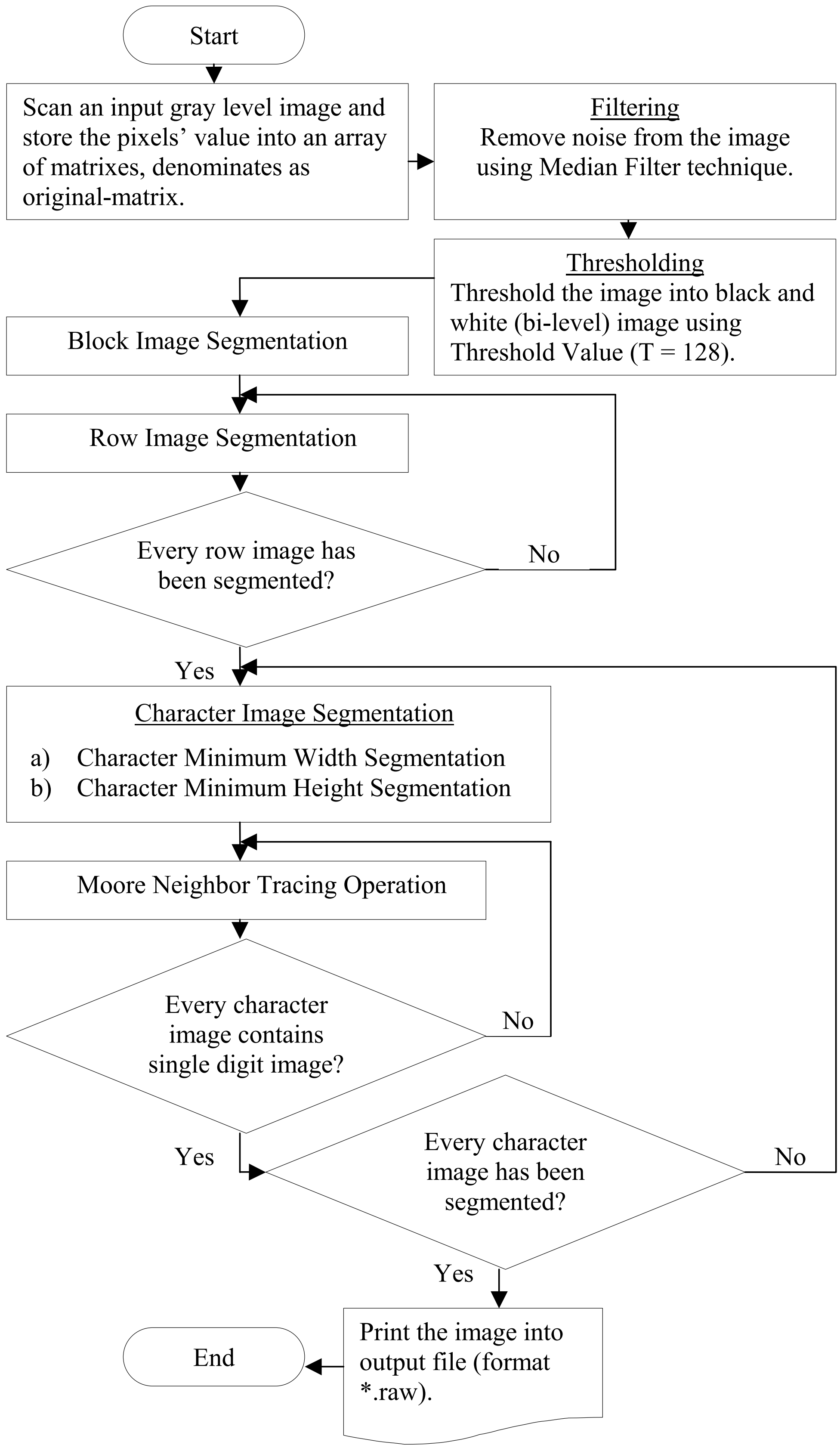


Figure 2.2.1.6: Segmentation operation using Simple Block Segmentation combines with Moore Neighbor Tracing Algorithm (SBS & MNTA).

2.2.2 Safe-point Thinning Algorithm (SPTA)

Safe-point Thinning Algorithm (SPTA) is used in image thinning process. Algorithm SPTA involved two scanning in each execution for each pixel. In first scanning, the entire right edge-point and left edge-point that are not safe-point is marked for later deletion. The same process is repeated for each top edge-point and bottom edge-point in the second scanning. Operation deletion is carried out until there isn't marking point.

The steps involved in Safe Point Thinning Algorithm (SPTA) are described as below:

1) Read the image from left to right and from top to bottom.
2) Store the pixel value into a variable, P.

3) Execute the following steps for all pixels in row and column.
4) Verify whether point P is black dot and is not marked. If point P is black dots and is not marked, the following steps wouldn't be continued.

5) Check whether p is edge-point. SPTA verify edge-point according to 4 types stated as below:

   P is left edge point, if the neighbour of x_4 is white dot.

   P is right edge point, if the neighbour of x_0 is white dot.

   P is top edge point, if the neighbour of x_2 is white dot.

   P is bottom edge point, if the neighbour of x_6 is white dot.

6) Verify whether point P is safe-point. Boolean operation to determine the safe-point are listed as below:

   Left safe-point, $S_4 = x_1 * (x_2+x_3+x_7+x_8) * (x_3+!x_4) * (x_7+!x_6) == 0$,

   Right safe-point, $S_0 = x_5 * (x_6+x_7+x_3+x_4) * (x_7+!x_8) * (x_3+!x_2) == 0$,

   Top safe-point, $S_2 = x_7 * (x_8+x_1+x_5+x_6) * (x_1+!x_2) * (x_5+!x_4) == 0$,

   Bottom safe-point, $S_6 = x_3 * (x_4+x_5+x_1+x_2) * (x_5+!x_6) * (x_1+!x_8) == 0$.

7) Each edge-point that is not safe-point (point that failed to meet the rule of Boolean operation) will be marked and deleted. Else, point P is labeled as one (1).
8) Step 2) to Step 3) is repeated for the entire image pixel.

2.3 Feature Extraction

Three type of afore mentioned moment functions are described in this research; they are geometric moments, Zernike moments and contour sequence moments. Geometric moments are computed using conventional method and this involved translation, scale and rotation invariants. Meanwhile, Zernike moments are computed in corresponding with geometric moments' expression to achieve translation and scale invariants. Besides, contour sequence moments is also applied in this project in image moments calculation. The computation of these moment functions are detailed here.

2.3.1 Geometric Moments Computation

The computation steps of geometric moments are described as below:

1) Read an input image data from left to right and from top to bottom.

2) Threshold the image data to extract the target process area.

3) Compute the image moment value, m_{pq} until third order with formula:

$$m_{pq}' = \iint_{\delta} (x')^p (y')^q f'(x', y') dx' dy' \quad ; \quad p, q = 0, 1, 2, \ldots$$

4) Compute the intensity moment, (x_0, y_0) of image with formula:

$$x_0 = m_{10} / m_{00} ; \qquad y_0 = m_{01} / m_{00} .$$

5) Compute the central moments, μ_{pq} with formula :

$$\mu_{pq} = \iint_{\delta} (x - x_0)^p (y - y_0)^q f(x, y) dx dy \quad ; \quad p, q = 0, 1, 2, \ldots$$

6) Compute normalized central moment, η_{pq} to be used in image scaling until third order with formula:

$$\gamma = (p+q+2)/2, \quad \eta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^{\gamma/2}} \quad , p+q \le 3.$$

7) Compute geometric moments, φ_1 to φ_4 with respect to translation, scale and rotation (geometric moment invariants) invariants with formula below:

$$\varphi_1 = \eta_{20} + \eta_{02}$$

$$\varphi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}{}^2$$

$$\varphi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$\varphi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \qquad (2.3.1.1)$$

The computed numerical values of φ_1 to φ_7 are very small, thus the logarithms of the absolute values of the functions are used as features representing the image. The computation steps of geometric moments are summarized in Figure 2.3.1.2.

2.3.1.1 Framework of Geometric Moments Computation

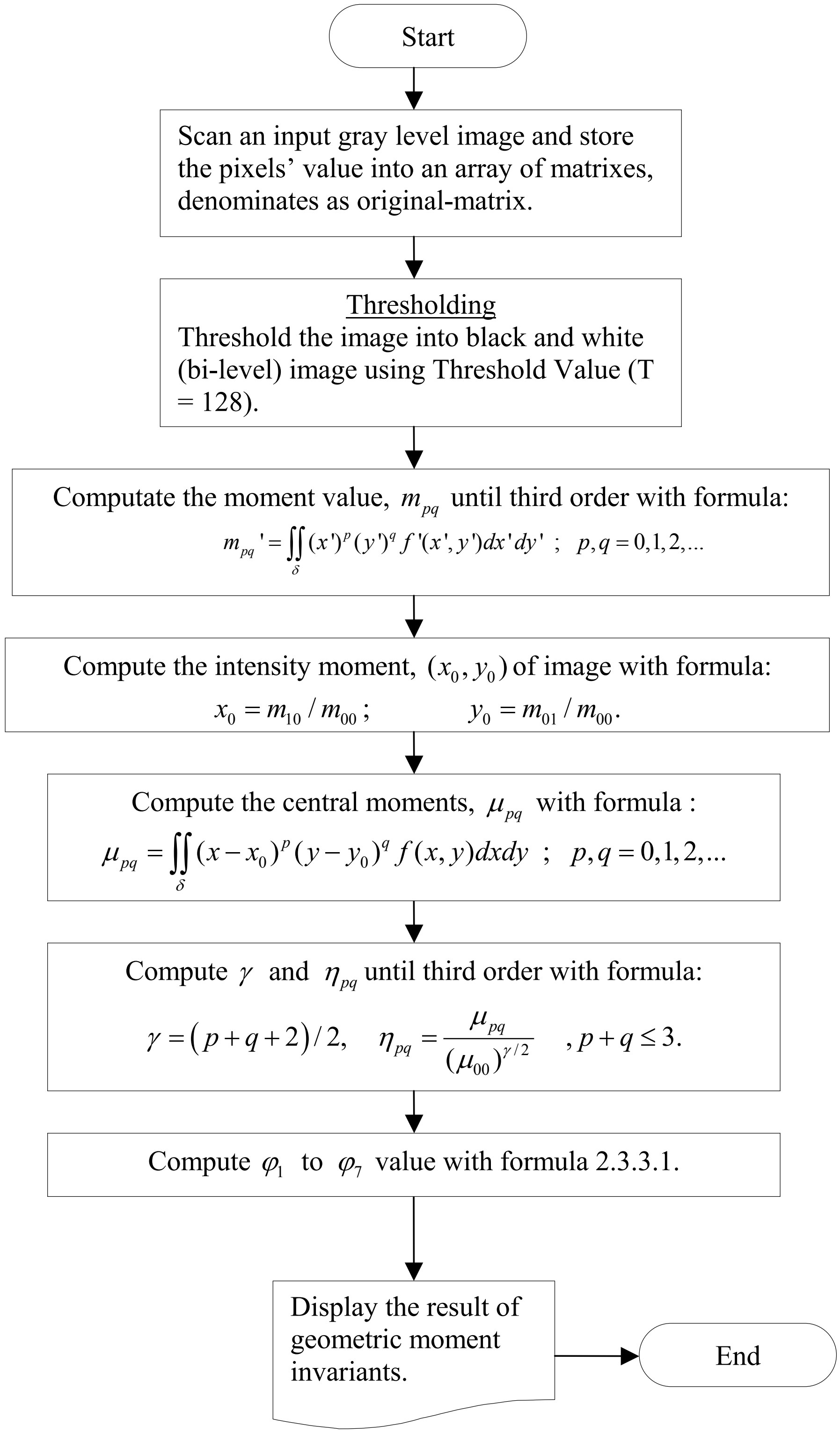


Figure 2.3.1.2: Flow chart above illustrates how the Geometric Moment Invariants are computed.

2.3.2 Zernike Moments Computation

The computation steps of Zernike moments are described as below:

1) Read an input image data from left to right and from top to bottom.

2) Threshold the image data to extract the target process area.

3) Compute the image moment value, m_{pq} until third order with formula:

$$m_{pq}' = \iint_{\delta} (x')^p (y')^q f'(x', y') dx' dy' \ ; \quad p, q = 0, 1, 2, \ldots$$

4) Compute the intensity moment, (x_0, y_0) of image with formula:

$$x_0 = m_{10} / m_{00} \ ; \qquad\qquad y_0 = m_{01} / m_{00}.$$

5) Compute the central moments, μ_{pq} with formula :

$$\mu_{pq} = \iint_{\delta} (x - x_0)^p (y - y_0)^q f(x, y) dx dy \ ; \quad p, q = 0, 1, 2, \ldots$$

6) Compute normalized central moment, η_{pq} to be used in image scaling until third order with formula:

$$\gamma = (p + q + 2)/2, \quad \eta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^{\gamma/2}} \quad , p + q \leq 3.$$

7) Compute Zernike moment invariants to rotation, translation and scale correspond to geometric moments with formula below:

$$Z_{20} = (3/\pi) \ [2(m_{20} + m_{02}) - m_{00}]$$

$$|Z_{22}|^2 = (3/\pi)^2 \left[(m_{20} - m_{02})^2 + 4{m_{11}}^2 \right]$$

$$|Z_{31}|^2 = (12/\pi)^2 \left[(m_{30} + m_{12})^2 + (m_{03} + m_{21})^2 \right]$$

$$|Z_{33}|^2 = (4/\pi)^2 \left[(m_{30} - 3m_{12})^2 + (m_{03} - 3m_{21})^2 \right] \qquad (3.2)$$

Zernike moments of order 3 are utilized because results proved that moments of order 3 are already adequate for feature representation. The calculated Zernike moments are small, thus $\log_{10}|Z|$ is also applied in representing the image. The computation steps of Zernike moments are summarized in Figure 2.3.2.1.

2.3.2.1 Framework of Zernike Moments Computation

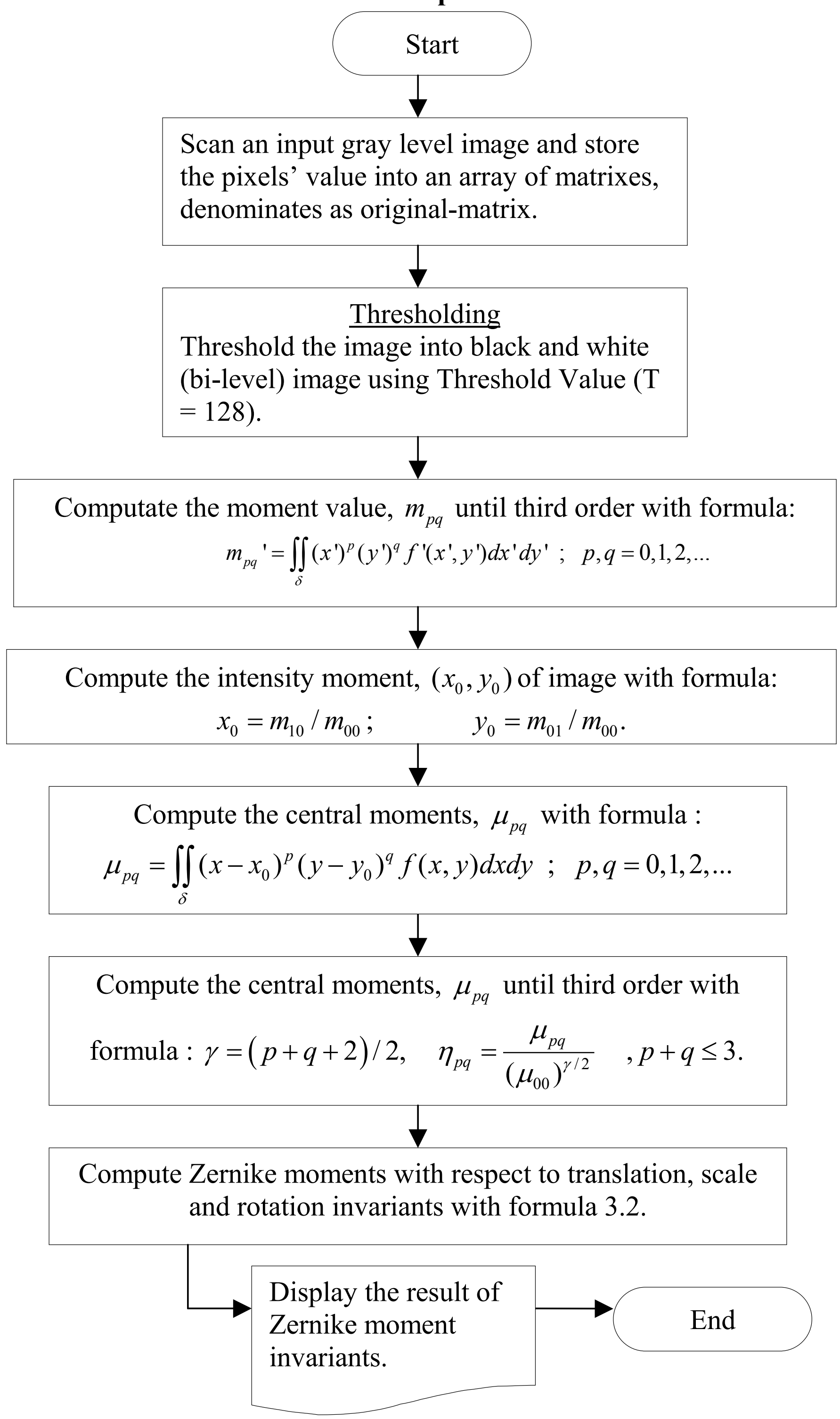


Figure 2.3.2.1: Flow chart of Zernike moments computation.

2.3.3 Contour Sequence Moments Computation

The computation steps of contour sequence moments are described as below:

1) Read an input image data from left to right and from top to bottom.

2) Threshold the image data to extract the target process area.

3) Compute the Euclidean Distance *z(i)*, i = 1, 2, 3,…,N of the vector connecting the centroid.

4) Compute the rth moment value, m_r until fifth order with formula:

$$m_r = \frac{1}{N}\sum_{i=1}^{N}\left[z(i)\right]^r$$

5) Compute the rth central moment with formula:

$$M_r = \frac{1}{N}\sum_{i=1}^{N}\left[z(i) - m_1\right]^r$$

6) Compute the four lower order moments with formula:

$$F_1 = \frac{(M_2)^{\frac{1}{2}}}{m_1}; \qquad F_3 = \frac{M_4}{(M_2)^2};$$

$$F_2 = \frac{M_3}{(M_2)^{\frac{3}{2}}}; \qquad F_4 = \frac{M_5}{(M_2)^{\frac{5}{2}}};$$

As mentioned before, higher order moments are more sensitive to noise and the resulting classifier will be less tolerant to noise. Thus, only these four low order moments which are stable are used as input to be fed into the neuro-fuzzy classification system. The computation steps of contour sequence moments are summarized in Figure 2.3.3.1.

2.3.3.1 Framework of Contour Sequence Moments Computation

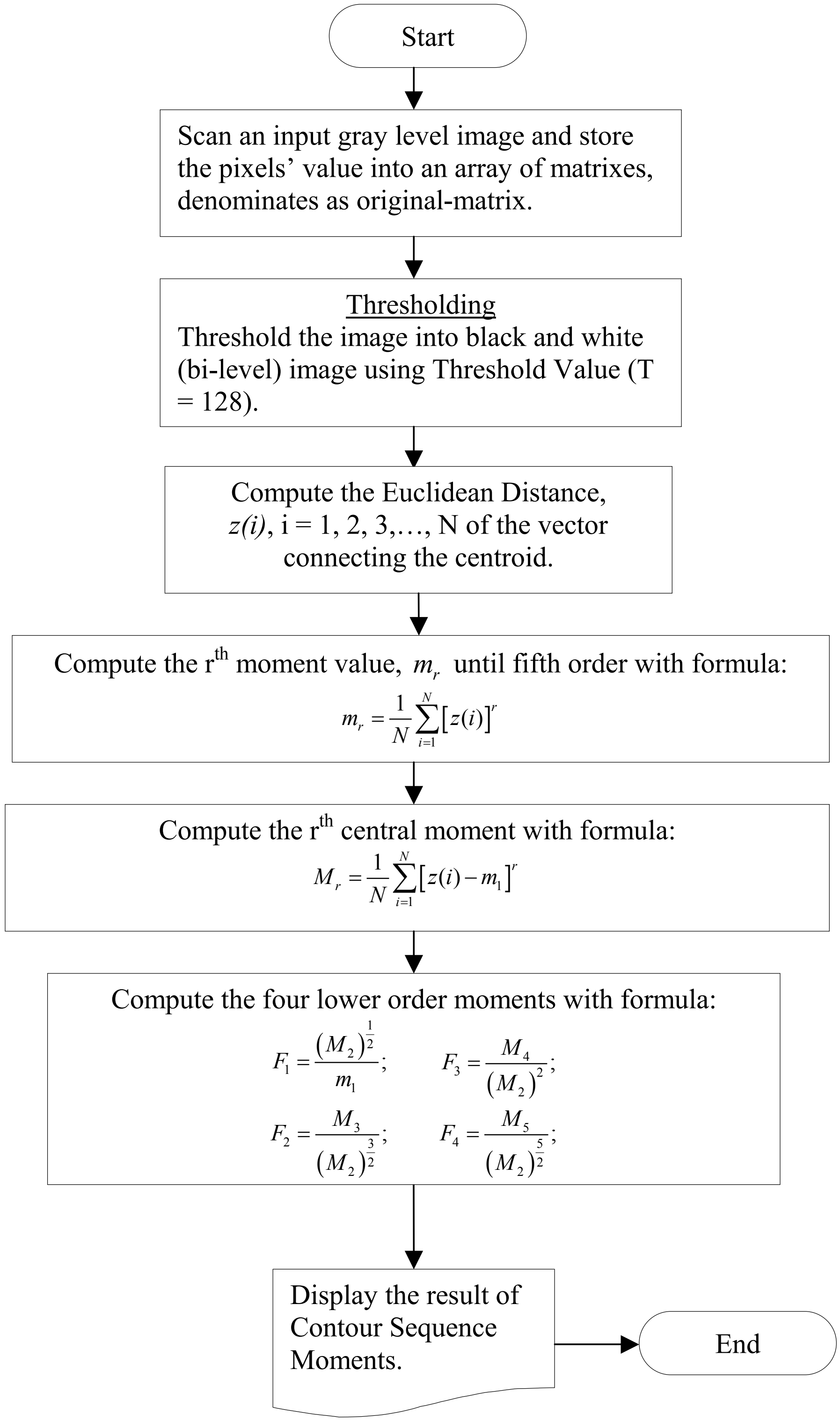


Figure 2.3.3.1: A framework of contour sequence moments computation

2.4 Neuro-fuzzy Classification

The extracted features from moment functions are used in training and classification using neuro-fuzzy classifier. Results of classification using these features are compared in terms of accuracy with and without applying thinning operation. Feed forward neural network with back propagation learning algorithm and sigmoid activation function are utilized in classification and recognition stage. The network weights are initialized using triangular membership functions

The steps of neuro-fuzzy classification are explained as below:

1) Feed the neuro-fuzzy system with n input patterns. In our case, four geometric moments feature, four Zernike moments feature and four contour sequence moments feature are utilized respectively in each case.

2) Setup the neural network model: one input layer with n neurons, M hidden layer with N neurons and one output layer with P neuron. n is the number of input features used. In our case, we set the M to 2 and an appropriate N is determined using try and error approach. P is set to 10 as our output is a combination of 10 features.

3) Set the learning rate, η and momentum rate, α .

4) Initialize the connection weights and node threshold (bias, θ) of hidden and output layer to small random values, range between [-0.5, 0.5] using triangular fuzzy membership function.

    a) Generate a random values, x.

    b) Pass x into the triangular membership function.

$$\mu(x) = \begin{cases} 0, & \text{if x} \le a - \frac{b}{2} \\ 1 - \frac{2|x_i - a|}{b}, & \text{if } a - \frac{b}{2} \le x < a + \frac{b}{2} \\ 0, & \text{if x} \ge a + \frac{b}{2} \end{cases}$$

c) Assign the generated value to initial network weights for each node.

5) Set the maximum allowed network error, E_{max}.

6) Activate the back-propagation neural network by applying inputs $x_1(p), x_2(p), \ldots, x_n(p)$ and desired outputs $y_{d,1}(p), y_{d,2}(p), \ldots, y_{d,n}(p)$.

a) Calculate the actual outputs of the neurons in the hidden layer:

$$y_j(p) = sigmoid\left[\sum_{i=1}^{n} x_i(p) \times w_{ij}(p) - \theta_j\right],$$

where n is the number of inputs of neuron j in the hidden layer, and *sigmoid* is the sigmoid activation function.

b) Calculate the actual outputs of the neurons in the output layer:

$$y_k(p) = sigmoid\left[\sum_{j=1}^{m} x_{jk}(p) \times w_{jk}(p) - \theta_k\right],$$

where m is the number of inputs of neuron k in the output layer.

7) Update the weights in the back-propagation network propagating backward the errors associated with output neurons.

a) Calculate the error gradient for the neurons in the output layer:

$$\delta_k(p) = y_k(p) \times \left[1 - y_k(p)\right] \times e_k(p)$$

where

$$e_k(p) = y_{d,k}(p) - y_k(p)$$

b) Calculate the weight corrections:

$$\Delta w_{jk}(p) = \alpha \times y_j(p) \times \delta_k(p)$$

Update the weights at the output neurons:

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p)$$

8) Update the weights in the back-propagation network propagating backward the errors associated with hidden neurons.

a) Calculate the error gradient for the neurons in the hidden layer:

$$\delta_j(p) = y_j(p) \times \left[1 - y_j(p)\right] \times \sum_{k=1}^{l} \delta_k(p) \times w_{jk}(p)$$

b) Calculate the weight corrections:

$$\Delta w_{ij}(p) = \alpha \times x_i(p) \times \delta_j(p)$$

Update the weights at the hidden neurons:

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$$

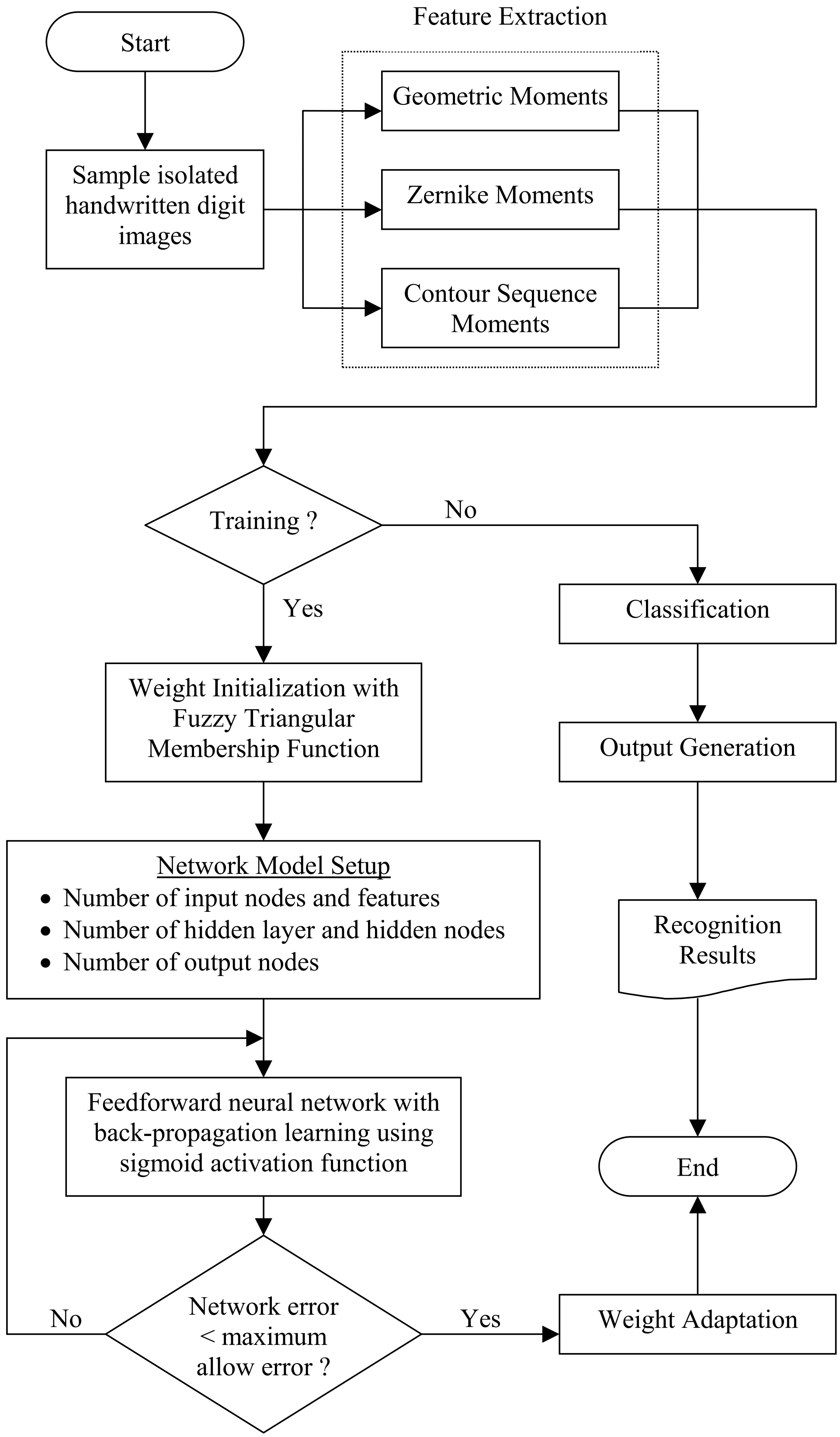
9) Compute the network squared error, *Error* at the output layer:

$$E = \frac{1}{2} \sum_{p} \sum_{k} \left(t_{kp} - o_{kp}\right)^2$$

where t_{kp} and o_{kp} are the target and actual outputs of neuron *'k'* for pattern *'p'*.

If *Error* > E_{max}, then repeat step 6 – 9. Else, terminate the network training. In our case, we set the number of maximum epoch for network training. Therefore, our training will terminate when the *Error* < E_{max} or the network fails to converge within a specific epoch size. A framework of the neuro-fuzzy training and classification system is illustrated in Figure 2.4.1.

2.4.1 Framework of Neuro-fuzzy Classification



Chapter 3
Algorithm and Implementation

In this section, the approaches used afore mentioned is translated into algorithm in order be incorporated and executed in the prototype (program) developed. They included algorithm for: image thresholding, segmentation and thinning, feature extraction with three moment functions and also the structure of the neuro-fuzzy classifier.

3.1 Image Pre-processing

3.1.1 Algorithm image thresholding

The gray level images are converted into bi-level (black and white) images. Algorithm used is presented in Figure 3.1.1.

```
for i:=0 to imageHeight-1 do
begin
   for j:=0 to imageWidth-1 do
   begin
         if(pixelValue[i,j] >= 128)then pixelValue[i,j] := 255     //white
   else if(pixelValue[i,j] <   128)then pixelValue[i,j] := 0;       //black
   end;
end;
```

Figure 3.1.1: Algorithm image thresholding

3.1.2 Simple Block Segmentation with Moore Neighbor Tracing Algorithm (SBS & MNTA)

SBS & MNTA is a newly implemented segmentation technique based on assumption that an image file only contains multiple disconnected and unbroken numeral characters. Four sub-modules are implemented to find the left-most, right-most, upper-most and lower-most black pixel resides in an image area with height and width image is specified. The summary of algorithm exploited is presented in Figure 3.1.2.1, Figure 3.1.2.2, Figure 3.1.2.3 and Figure 3.1.2.4.

```
for i:=upper to lower-1 do
    for j:= left to right-1 do
        if(pixelValue[i,j]=0)then
            begin

            //find the left most black pixel coordinate
            if(foundLeftMostPixelCoord = false)then
                begin
                    leftMostPixelCoord := j;
                    foundLeftMostPixelCoord := true;
                end

             //verify whether there is other pixels' coordinate that is left most
             //then the previous found pixel
            else if(foundLeftMostPixelCoord = true)then
                begin
                    //assign the new left most pixel coordinate value
                    if(j<leftMostPixelCoord)then
                        leftMostPixelCoord := j;
                end;
            end;
```

Figure 3.1.2.1: Algorithm to find the left most black pixel resides in a specific image area

```
for i:=upper to lower-1 do
    for j:= left to right-1 do
        if(pixelValue[i,j]=0)then

            begin
            //find the right most black pixel coordinate
            if(foundRightMostPixelCoord = false)then
                begin
                    rightMostPixelCoord := j;
                    foundRightMostPixelCoord := true;
                end

             //verify whether there is other pixels' coordinate that is right most
             //then the previous found pixel
            else if(foundRightMostPixelCoord = true)then
                begin
                    //assign the new right most pixel coordinate value
                    if(j>rightMostPixelCoord)then
                        rightMostPixelCoord := j;
                end;
            end;
```

Figure 3.1.2.2: Algorithm to find the right most black pixel resides in a specific image area

```
for j:= left to right-1 do
    for i:=upper to lower-1 do
        if(pixelValue[i,j]=0)then

            //find the upperMostPixel coordinate
            if(foundUpperMostPixelCoord = false)then
            begin
                upperMostPixelCoord := i;
                foundUpperMostPixelCoord := true;
            end

            //verify whether there is other pixels' coordinate that are upper
            //most than the previous found pixel
            else if(foundUpperMostPixelCoord = true)then
            begin
                //assign the new upper most pixel coordinate value
                if(i<upperMostPixelCoord)then
                    upperMostPixelCoord := i;
            end;
```

Figure 3.1.2.3: Algorithm to find the upper most black pixel resides in a specific image area

```
for j:= left to right-1 do
    for i:=upper to lower-1 do
        if(pixelValue[i,j]=0)then

            //find the lowerMostPixel coordinate
            if(foundLowerMostPixelCoord = false)then
            begin
                lowerMostPixelCoord := i;
                foundLowerMostPixelCoord := true;
            end

            //verify whether there is other pixels' coordinate that are lower
            //most than the previous found pixel
            else if(foundLowerMostPixelCoord = true)then
            begin
                //assign the new lower most pixel coordinate value
                if(i>lowerMostPixelCoord)then
                    lowerMostPixelCoord := i;
            end;
```

Figure 3.1.2.4: Algorithm to find the lower most black pixel resides in a specific image area

3.1.3 Safe-point Thinning Algorithm (SPTA)

Safe-point Thinning Algorithm (SPTA) used is presented in Figure 3.1.3.1.

```
turn := 0;
finishProcessAllPixel := false;
while(finishProcessAllPixel = false) do
begin
        finishProcessAllPixel := true;          turn := (turn+1) mod 2;

        for y:=1 to height-2 do          //initialize the pixelOnFlag to false
        begin
                for x:=1 to width-2 do
                        pixelOnFlag[y,x] := 0;
        end;

        for y:=1 to height-2 do
        begin
        for x:=1 to width-2 do

  begin
      if(pixelValue[y,x] = 0)then      //black pixel
      begin
            blackPixel := 0;
            for j:=-1 to 1 do begin
               for i:=-1 to 1 do begin
               if(pixelValue[y+j,x+i] = 0) then
                        blackPixel:= blackPixel + 1;
               end;
            end;

      if((blackPixel > 2) and (blackPixel < 8))then
      begin
      kernelValue[0] := pixelValue[y-1,x-1];  kernelValue[5] := pixelValue[y+1,x  ];
      kernelValue[1] := pixelValue[y-1,x  ];    kernelValue[6] := pixelValue[y+1,x-1];
      kernelValue[2] := pixelValue[y-1,x+1];  kernelValue[7] := pixelValue[y  ,x-1];
      kernelValue[3] := pixelValue[y  ,x+1];   kernelValue[8] := pixelValue[y-1,x-1];
      kernelValue[4] := pixelValue[y+1,x+1];

      whitePixel := 0;
      for z:=0 to 7 do
               if((kernelValue[z] = 255) and (kernelValue[z+1] = 0))then
                        whitePixel := whitePixel + 1;
```

```
      if(whitePixel = 1)then
      begin

      if(  (turn=0) and ((kernelValue[3]=255) or (kernelValue[5]=255)
          or (kernelValue[1]=255) and (kernelValue[7]=255))  ) then
      begin
            pixelOnFlag[y,x] := 1;            finishProcessAllPixel := false;
      end
      else if ( (turn=1) and ((kernelValue[1]=255) or
            (kernelValue[7]=255) or
            (kernelValue[3]=255) and (kernelValue[5]=255)) ) then
      begin
            pixelOnFlag[y,x] := 1;            finishProcessAllPixel := false;
      end;
      end;

      end;
      end;
   end;
   end;

      for y:=1 to height-2 do
      for x:=1 to width-2 do
            if(pixelOnFlag[y,x] = 1) then
                  pixelValue[y,x] := 255;           //delete the pixel
   end;
```

Figure 3.1.3.1: Safe-point Thinning Algorithm (SPTA)

3.2 Feature Extraction with Moment Functions

3.2.1 Computation of Geometry Moment Invariants

Algorithm of geometric moments computation is presented in Figure 3.2.1.1.

// Compute the moment value, m_{pq} until third order.

```
for p:=0 to 3 do
  for q:=0 to 3 do
        begin
        moment[p,q] := 0.0;

           for i:=0 to height-1 do
             for j:=0 to width-1 do
                begin
                       if (p=0) and (q=0) then
                        moment[p,q] := moment[p,q] + 1

                      else if (p=0) then
                        moment[p,q] := moment[p,q] + Power(j, q)

                      else if (q=0) then
                        moment[p,q] := moment[p,q] + Power(i, p)

                      else
                        moment[p,q] := moment[p,q] + Power(i, p)* Power(j, q);
                end;
         end;
```

// Compute the intensity moment, (x_0, y_0) about the x-axis and y-axis of image.

```
xCenter := moment[1,0]/moment[0,0];
yCenter := moment[0,1]/moment[0,0];
```

// Compute the central moments, μ_{pq} (with respect to the intensity centroid).

```
for p:=0 to 3 do
 for q:=0 to 3 do
        begin
          miu[p,q] := 0.0;

          if((p+q) <=3 )then
```

```
        for i:=0 to height-1 do
        for j:=0 to width-1 do
        begin
           if( p=0 ) then
           begin
                   if( q=0 ) then miu[p,q] := moment[p,q]
              else if( q<>0 ) then miu[p,q] := miu[p,q] + Power(j - yCenter, q);
           end
           else if( p<>0 )then
           begin
                   if( q=0 ) then miu[p,q] := Power(i - xCenter, p)
              else if ( q<>0 )then
              miu[p,q] := miu[p,q] + Power(i-xCenter,p) * Power(j - yCenter,q);
           end;
        end;
      end;
```

// Compute γ and η_{pq} , then calculate the moment invariants in respect to
//translation, scale and rotation of an image.

```
for p:=0 to 3 do
  for q:=0 to 3 do
        if((p+q<4) and (p+q>=2)) then
           begin
              gamma := (p+q)/2.0 + 1.0;
              norm[p,q] := miu[p,q]/Power(miu[0,0],gamma)
           end;

phi1 := norm[2,0] + norm[0,2];

phi2 := Power(norm[2,0] - norm[0,2], 2) + 4* Power(norm[1,1], 2);

phi3 := Power(norm[3,0] - 3*norm[1,2], 2) + Power(3*norm[2,1] - norm[0,3], 2);

phi4 := Power(norm[3,0] + norm[1,2], 2) + Power(norm[2,1] + norm[0,3], 2);
```

Figure 3.2.1.1: Algorithm of geometric moments computation

3.2.2 Computation of Zernike Moment Invariants

Algorithm of Zernike moments computation is presented in Figure 3.2.2.1.

```
// Compute the moment value, m_pq until third order.
for p:=0 to 3 do
  for q:=0 to 3 do
        begin
        moment[p,q] := 0.0;

           for i:=0 to height-1 do
             for j:=0 to width-1 do
                 begin
                        if (p=0) and (q=0) then
                         moment[p,q] := moment[p,q] + 1

                        else if (p=0) then
                         moment[p,q] := moment[p,q] + Power(j, q)

                        else if (q=0) then
                         moment[p,q] := moment[p,q] + Power(i, p)

                        else
                         moment[p,q] := moment[p,q] + Power(i, p)* Power(j, q);
                 end;
        end;

// Compute the intensity moment, (x_0, y_0) about the x-axis and y-axis of image.

xCenter := moment[1,0]/moment[0,0];
yCenter := moment[0,1]/moment[0,0];

// Compute the central moments, μ_pq (with respect to the intensity centroid).
for p:=0 to 3 do
 for q:=0 to 3 do
        begin
          miu[p,q] := 0.0;

          if((p+q) <=3 )then
```

Figure 3.2.2.1: Algorithm of Zernike moments computation

```
for i:=0 to height-1 do
        for j:=0 to width-1 do
        begin
           if( p=0 ) then
           begin
                   if( q=0 ) then miu[p,q] := moment[p,q]
                 else if( q<>0 ) then miu[p,q] := miu[p,q] + Power(j - yCenter, q);
           end
           else if( p<>0 )then
           begin
                   if( q=0 ) then miu[p,q] := Power(i - xCenter, p)
                 else if ( q<>0 )then
                 miu[p,q] := miu[p,q] + Power(i-xCenter,p) * Power(j - yCenter,q);
           end;
        end;
      end;
```

// Compute γ and η_{pq}, then calculate the moment invariants in respect to
//translation, scale and rotation of an image.

```
for p:=0 to 3 do
  for q:=0 to 3 do
        if((p+q<4) and (p+q>=2)) then
           begin
              gamma := (p+q)/2.0 + 1.0;
              norm[p,q] := miu[p,q]/Power(miu[0,0],gamma)
           end;

  //computation of Zernike Moment Invariants until order 3

  ZMI[2,0] := (3/pi) * (2 * (norm[2,0]+norm[0,2]) - norm[0,0]);

  ZMI[2,2] := Power(3/pi, 2.0) * (Power((norm[2,0] - norm[0,2]),2.0)
                 + 4 * Power(norm[1,1],2.0));

  ZMI[3,1] := Power(12/pi,2.0)*(Power(norm[3,0]+norm[1,2],2.0)
                 +Power(norm[0,3]+norm[2,1],2.0));

  ZMI[3,3] := Power(4/pi,2.0) * (Power((norm[3,0]-3*norm[1,2]),2.0) +
                 Power((norm[0,3] - 3*norm[2,1]),2.0));

  ZMI[2,0]:= (Log10(Abs(ZMI[2,0])));

  ZMI[2,2]:= (Log10(Abs(ZMI[2,2])));

  ZMI[3,1]:= (Log10(Abs(ZMI[3,1])));

  ZMI[3,3]:= (Log10(Abs(ZMI[3,3])));
```

3.2.3 Computation of Contour Sequence Moments

Algorithm of contour sequence moments computation is presented Figure 3.2.3.1.

```
// Compute the moment value, m_pq until third order.
for p:=0 to 3 do
  for q:=0 to 3 do
        begin
        moment[p,q] := 0.0;

            for i:=0 to height-1 do
              for j:=0 to width-1 do
                  begin
                          if (p=0) and (q=0) then
                           moment[p,q] := moment[p,q] + 1

                         else if (p=0) then
                           moment[p,q] := moment[p,q] + Power(j, q)

                         else if (q=0) then
                           moment[p,q] := moment[p,q] + Power(i, p)

                         else
                           moment[p,q] := moment[p,q] + Power(i, p)* Power(j, q);
                         nContour := nContour + 1;
                  end;
         end;

// Compute the intensity moment, (x0, y0) about the x-axis and y-axis of image.

        xCenter := moment[1,0]/moment[0,0];
        yCenter := moment[0,1]/moment[0,0];

//calculate euclidean distance
  for n:=0 to totalMoment-1 do
  begin
    sum := 0.0;
    begin
      for y:=0 to height-1 do
      for x:=0 to width-1 do
      begin
         ed[y,x] := Sqrt(Power(x - xCenter, 2.0) + Power(y - yCenter, 2.0));
      end;
    end;
  end;
```

Figure 3.2.3.1: Algorithm of contour sequence moments computation

```
//calculate the rth moment
for n:=0 to totalMoment-1 do
  begin
  sum := 0.0;

   for y:=0 to height-1 do
      for x:=0 to width-1 do
      begin
          sum := sum + Power(ed[y,x], n);
      end;
     rMoment[n] := 1 / nContour * sum;
    end;

//calculate the rth central moment
for n:=0 to totalMoment-1 do
  begin
      sum := 0.0;
      begin
        for y:=0 to height-1 do
        for x:=0 to width-1 do
          begin
            sum := sum + Power(ed[y,x] - rMoment[0], n);
          end;
      end;
      rCentralMoment[n] := 1 / nContour * sum;
  end;

// Normalized amplitude variation
F[1] := Power(rCentralMoment[1], 0.5) / rMoment[0];

// Coefficient of skewness
F[2] := rCentralMoment[2] / Power(rCentralMoment[1], 1.5);

// Coefficient of kurtosis
F[3] := rCentralMoment[3] / Power(rCentralMoment[1],2);

// For the 4th feature
F[4] := rCentralMoment[4] / Power(rCentralMoment[1],2.5)
```

Chapter 4
Experiment and Results

4.1 Neuro-fuzzy Classification

4.1.1 Feature Extraction of Digit Images

Four set of feature extracted from each moment functions is used as the inputs and fed into the neuro-fuzzy network for training and testing purpose. Four set of geometric moments are used and Zernike moments until third order, $Z_{20}, |Z_{22}|^2, |Z_{31}|^2$ and $|Z_{33}|^2$ are used to represent the features of digits. A total of four lower order moments in contour sequence moments are used as network input features. The calculated value of geometric moments and Zernike moments, F are insignificant, thus $\log_{10}|F|$ is applied to represent the images. Table 4.1, Table 4.2 and Table 4.3 illustrated features of 10 digits 0 extracted using geometric moments, Zernike moments and contour sequence moments.

Table 4.1: Extracted features of 10 digits 0 with geometric moments

φ_1	φ_2	φ_3	φ_4
-0.70740	-1.35606	-3.01467	-3.77361
-0.73260	-1.22780	-2.54807	-3.24797
-0.71731	-1.18102	-2.72851	-3.93310
-1.06623	-1.73776	-4.76563	-4.64301
-0.93278	-1.52876	-3.81407	-4.24388
-0.81164	-1.47272	-3.57920	-4.13660
-0.78394	-1.60268	-3.12202	-3.43368
-0.86787	-1.52239	-2.81233	-3.66443
-0.40384	-0.83744	-2.44235	-2.67630
-1.06337	-2.11614	-3.80914	-4.25853

Table 4.2: Extracted features of 10 digits 0 with Zernike moments

Z_{20}	$\|Z_{22}\|^2$	$\|Z_{31}\|^2$	$\|Z_{33}\|^2$
-0.42639	-1.39612	-2.60955	-2.80485
-0.45160	-1.26785	-2.08391	-2.33825
-0.43630	-1.22108	-2.76904	-2.51869
-0.78523	-1.77782	-3.47895	-4.55581
-0.65178	-1.56882	-3.07982	-3.60425
-0.53064	-1.51277	-2.97254	-3.36938
-0.50294	-1.64273	-2.26962	-2.91220
-0.58687	-1.56244	-2.50036	-2.60251
-0.12284	-0.87749	-1.51224	-2.23253
-0.78236	-2.15620	-3.09447	-3.59932

Table 4.3: Extracted features of 10 digits 0 with contour sequence moments

F_1	F_2	F_3	F_4
30.05362	6.47126	45.25373	333.90161
32.71818	5.95364	38.32488	260.43063
31.69769	6.13232	40.62963	283.97861
31.05886	6.29241	42.85799	308.45520
29.86565	6.52130	45.97804	342.18240
27.97482	6.96417	52.44178	416.88776
28.06775	6.94976	52.24658	414.80444
27.20984	7.17573	55.71798	457.03427
36.38897	5.34856	30.92183	188.66597
28.23771	6.91055	51.66535	407.97510

The numerical values of Table 4.1, Table 4.2 and Table 4.3 are plotted in Figure 4.1, Figure 4.2 and Figure 4.3 to show the deviation of values for the sample with respect to different size, style and orientations. From the listed figure, it can be observed that the higher the moments' order, the sign of deviation is more evident. This is because higher order moments contain finer details about the image and are often more sensitive to variation of style, orientations and image noise.

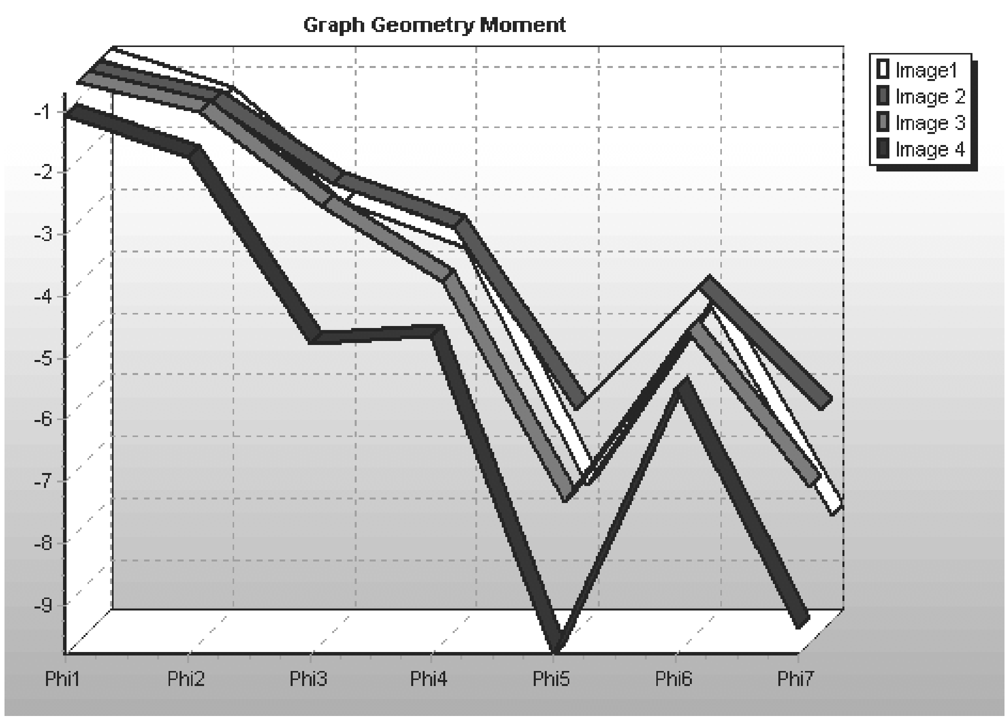


Figure 4.1: Features of digit 0 extracted using geometric moments

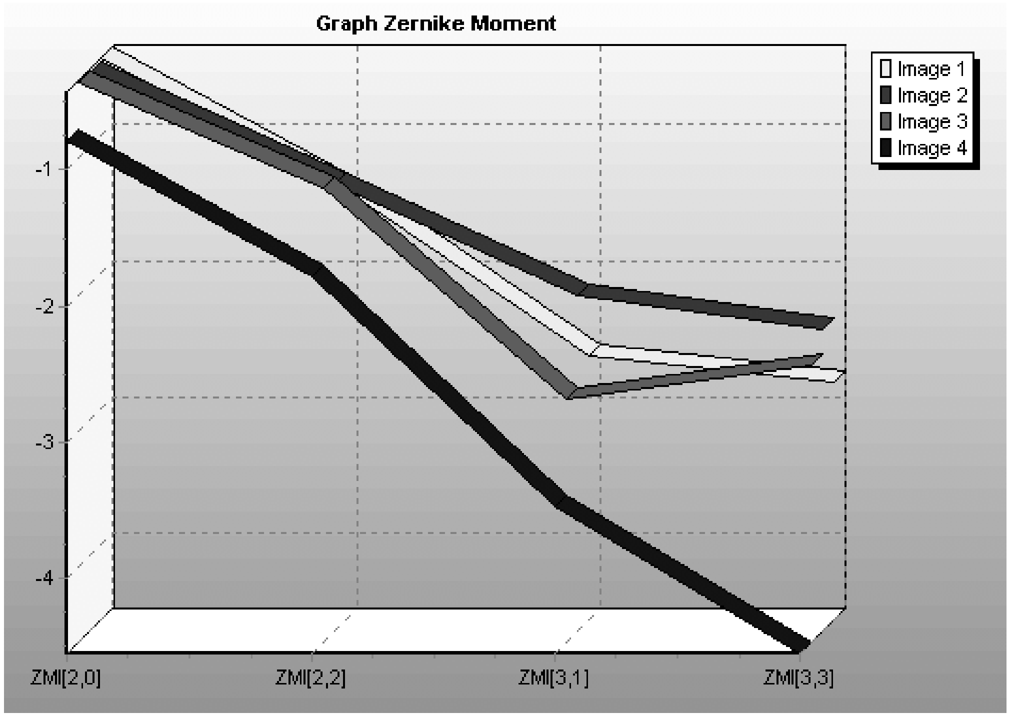


Figure 4.2: Features of digit 0 extracted using Zernike moments

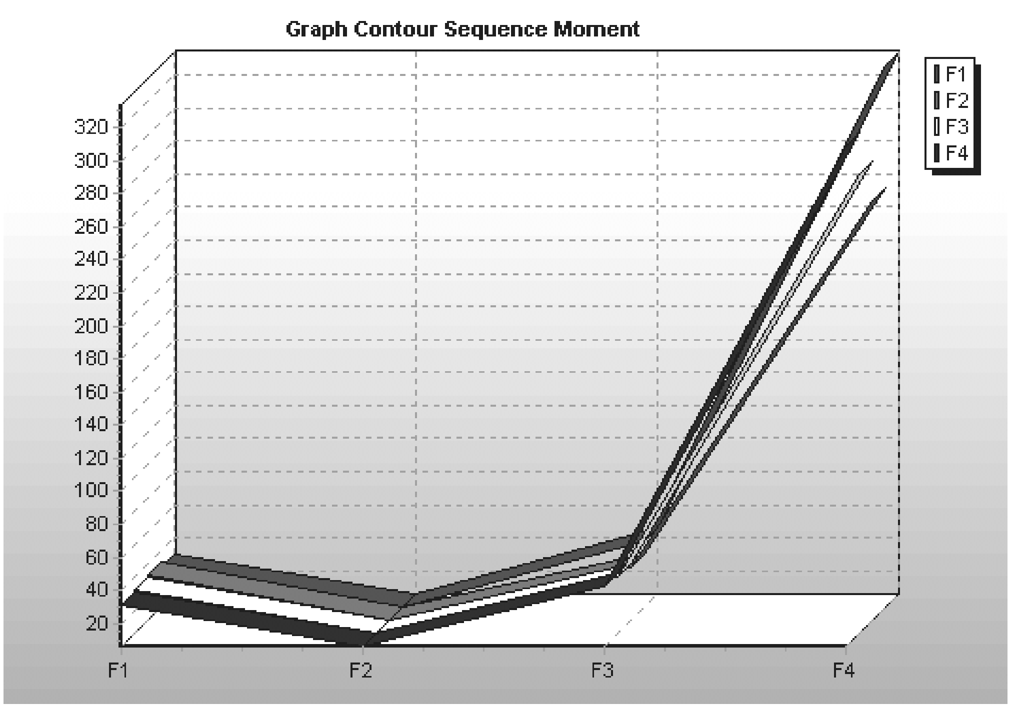


Figure 4.3: Features of digit 0 extracted using contour sequence moments

4.1.2 Intraclass Invariants

Several studies review that good features are those features with small intraclass invariance and larger interclass separation where features from difference classes should exhibit dissimilarities numerically. From Figure 4.1 to Figure 4.6, it can be seen that features extracted from contour sequence moments show better intraclass representation with close similarity compared with geometric moments and contour sequence moments. As illustrated in Figure 4.1 to Figure 4.6, both geometric moments and Zernike moments have possess dissimilarity features values. The purpose of classifications is to differentiate between classes; therefore, contour sequence moments in this case better in representing isolated handwritten digits from same class.

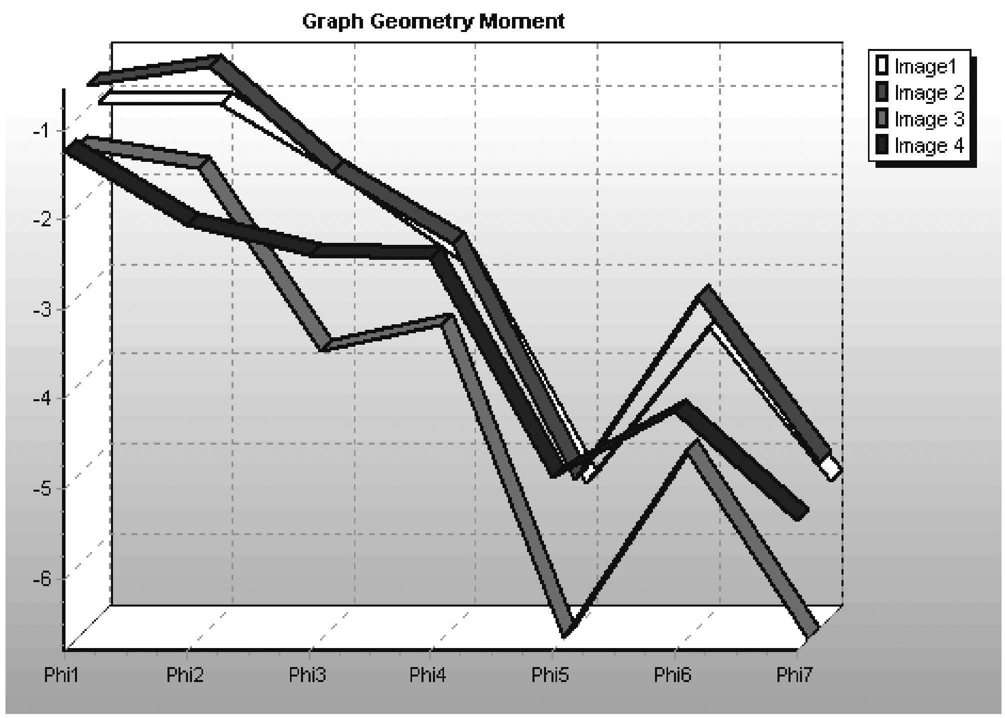


Figure 4.4: Intraclass invariance for features of digit 1 extracted using geometric moments

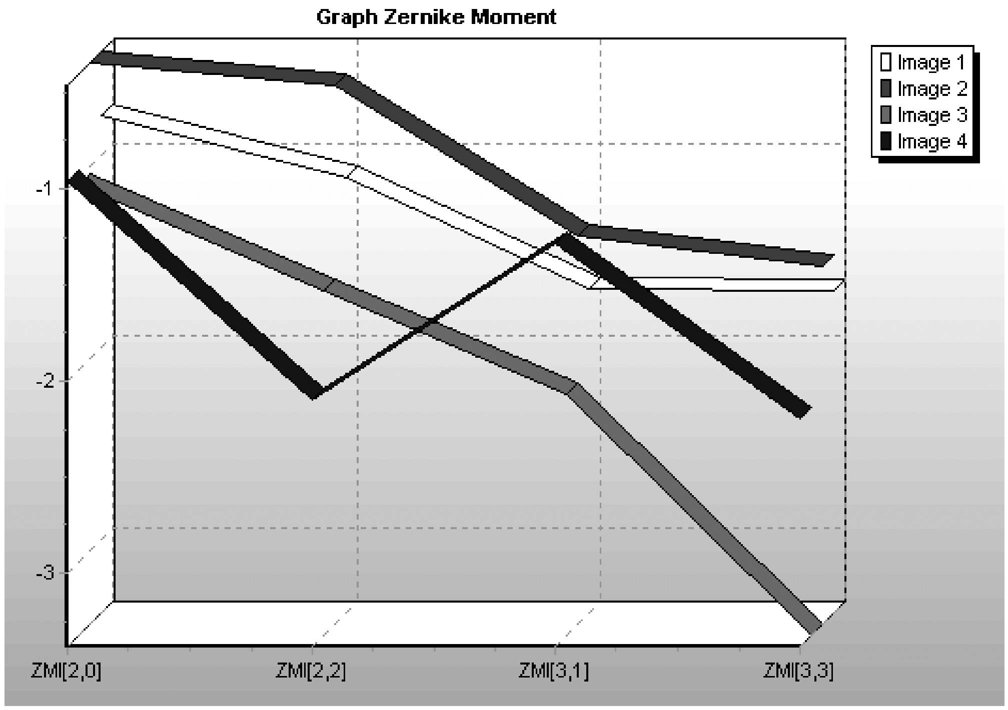


Figure 4.5: Intraclass invariance for features of digit 1 extracted using Zernike moments

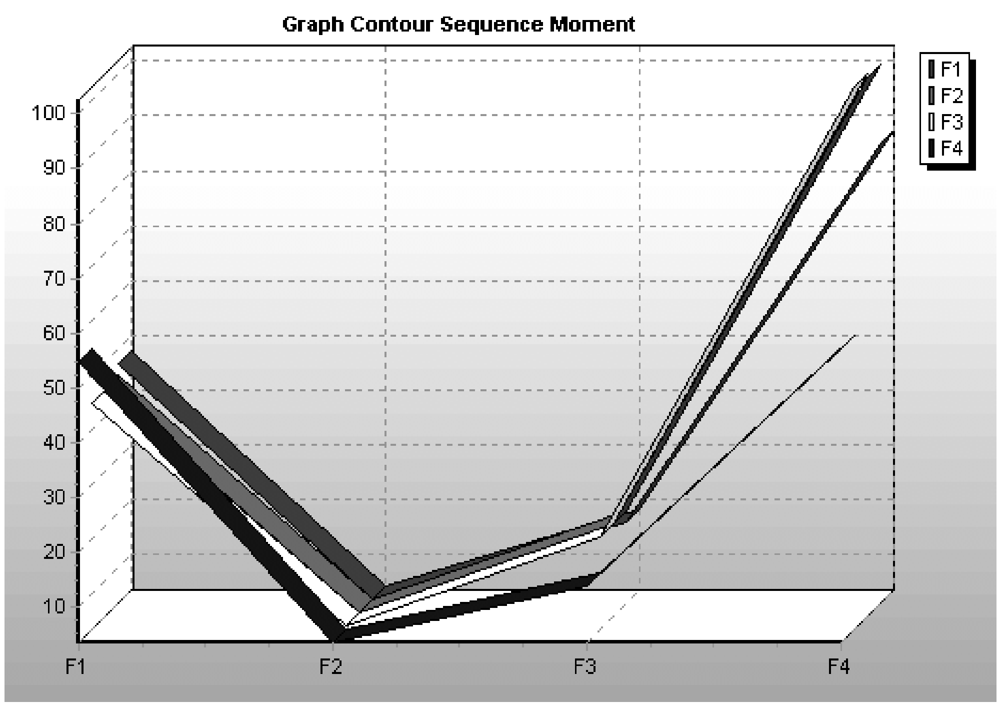


Figure 4.6: Intraclass invariance for features of digit 1 extracted using contour sequence moments

4.1.3 Interclass Invariants

Features extracted from different classes supposed to show variation in order to be used in classification and recognition purpose. The features used should be typical and unique in symbolizing a particular class of digits. In relation to that, a small deviation between features should be considered for a better differentiation rate. Figure 4.7, Figure 4.8 and Figure 4.9 illustrated the interclass invariance for features of digit 0 to 9 using geometric moments, Zernike moments and contour sequence moments. From these figures, it can be observed that Zernike moments performed better in representing the digit images as it exhibited greater divergences compared with geometric moments and contour sequence moments.

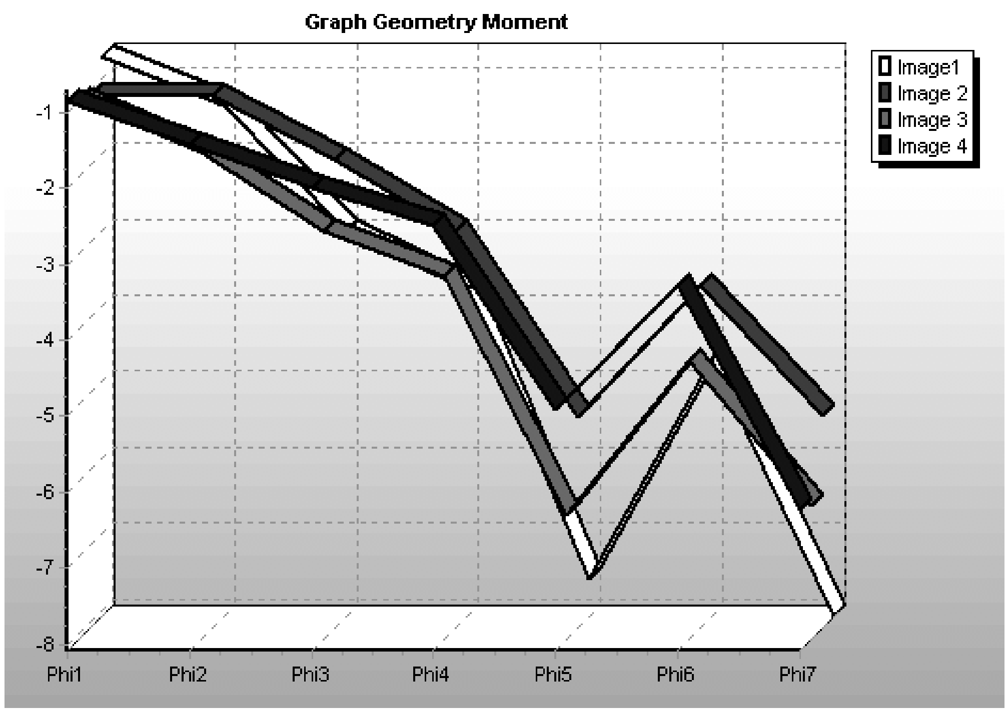


Figure 4.7: Interclass invariance for features of digit 0 to 4 extracted using geometric moments

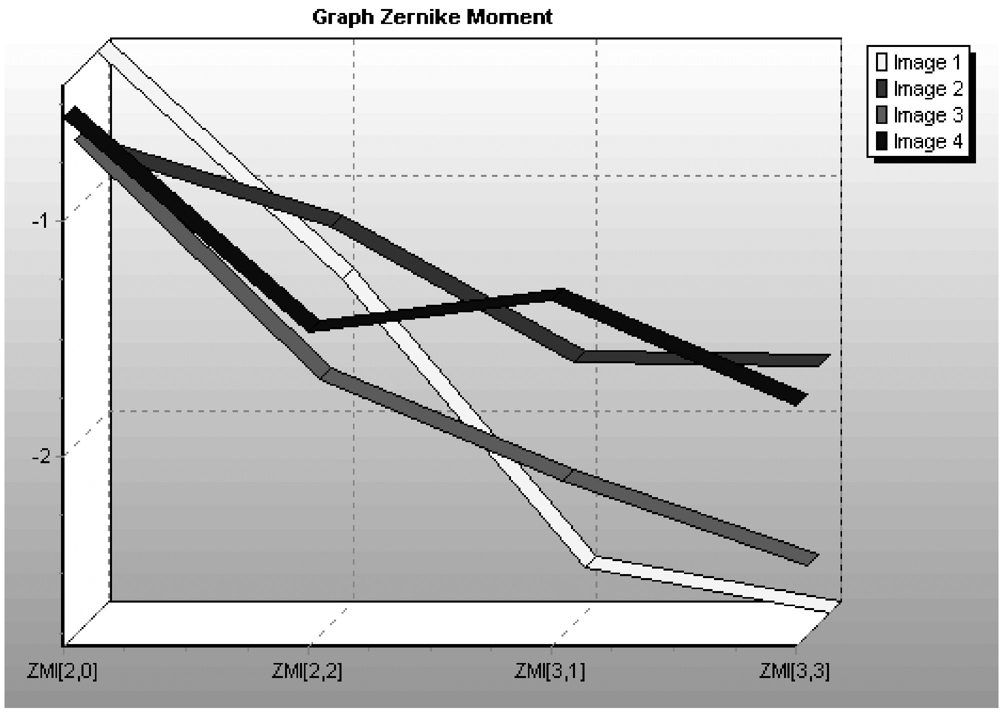


Figure 4.8: Interclass invariance for features of digit 0 to 4 extracted using Zernike moments

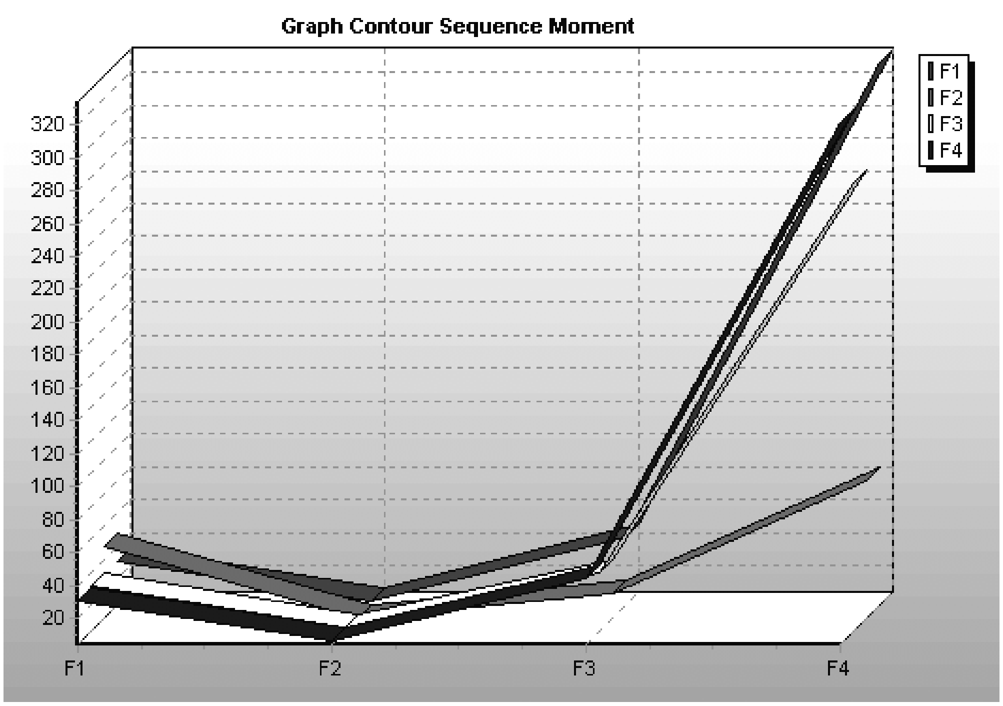


Figure 4.9: Interclass invariance for features of digit 0 to 4 extracted using contour sequence moments

4.2 Classification and Experimental Results

4.2.1 Network Training

In this project, standard back-propagation model with sigmoid logistic activation function is used in the training and classification phases. Sigmoid logistic function is utilized in both input-to-hidden layer and hidden-to-output layer.

The input data for the model are numerical values extracted from isolated digit images using geometric moments, Zernike moments and contour sequence moments. Five hundreds samples of isolated handwritten digit images are used in network training. Two samples digit with each sample contains 50 set digits image are applied in testing and classification phases. The network weights are initialized with Triangular Membership function and its efficiency of improving convergence rate is verified.

The learning rate, α is set to 0.05 while momentum rate, η is set to 0.3. The target outputs for each digit images are set to zero except for those that correspond to the class accordingly as shown in Table 4.4. For example, for digit 0, all output are set to zero except for the first output node while for digit 9, only the last output node is set to one. Figure 4.10 illustrates a sample of isolated handwritten digits used in network training.

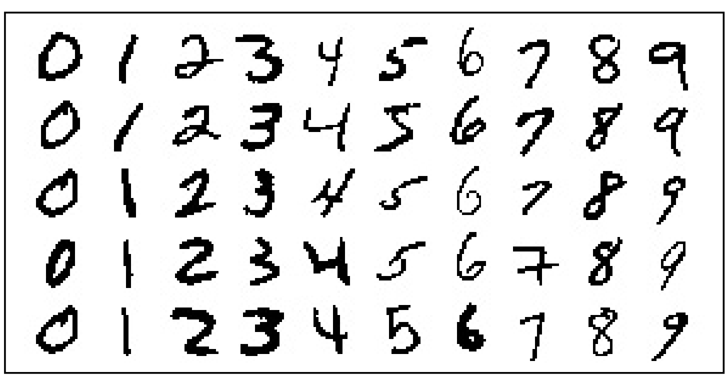


Figure 4.10: Sample of isolated handwritten digits for network training

Table 4.4: Target output for network input data

Digit	Target Network Output									
0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	1

4.2.2 Weight Initialization with Triangular Membership Function

Network weights are initialized using Triangular Membership function to verify its feasibility in enhancing the network convergence rate. From Table 4.5, it can be observed that network weights can be initialized with appropriate parameters using Triangular Membership Function, and in particular case decrease the iteration number required to converge to the solution.

Table 4.5: Weight initialization with Triangular Membership Function

Moment Functions	Triangular Membership (iteration)	Random Number (iteration)
Geometric	14569	14636
Zernike	11129	8985
Contour Sequence	4666	6514

4.2.3 Normalization of Input Features

Normalization is a transformation applied uniformly to each element in a set of data so that the set has some specific statistical property. In this project, normalization is conducted on features extracted from moment functions into range of [0, 1]. According to several studies, normalization of input features has shown to speed up convergence and recognition rate. In this project, the min-max normalization technique is applied using formula below:

Normalized value,

$$newX = \frac{oldX - \min Val}{\max Val - \min Val}\left(new\_\max Val - new\_\min Val\right) + new\_\min Val$$

Where

new$X$ is the new normalized value,

$oldX$ is the original value,

minVal is the smallest value of the sample,

maxVal is the largest value of the sample,

new_maxVal is the new largest value,

new_minVal is the new smallest value.

For example, in Table 4.6, Original value, $oldX =$ and

The new normalized value, new$X$ = -.42639 and minVal = -6.08105,

maxVal = 0.30229, new_maxVal = 1, new_minVal = 0.

The new normalized value,

$$newX = \frac{-0.42639 - (-6.08105)}{0.30229 - (-6.08105)}\left(1-0\right) + 0 = 0.88585$$

Table 4.6: Normalized Zernike moments value of sample digit images

Digit	Z_{20}	$\lvert Z_{22}\rvert^2$	$\lvert Z_{31}\rvert^2$	$\lvert Z_{33}\rvert^2$	Digit	Z_{20}	$\lvert Z_{22}\rvert^2$	$\lvert Z_{31}\rvert^2$	$\lvert Z_{33}\rvert^2$
0	0.88585	0.73393	0.54384	0.51324	5	0.91561	0.80988	0.78045	0.82952
	0.88190	0.75403	0.62618	0.58634		0.86971	0.73597	0.59548	0.58186
	0.88429	0.76135	0.51885	0.55807		0.86738	0.74657	0.71267	0.66990
	0.82963	0.67413	0.40764	0.23894		0.94300	0.91644	0.90204	0.80185
	0.85054	0.70688	0.47017	0.38801		0.92626	0.89612	0.62105	0.59219
1	0.82799	0.77657	0.68736	0.68436	6	0.85611	0.65027	0.70948	0.56921
	0.87979	0.86077	0.73867	0.71495		0.85400	0.67850	0.57877	0.48279
	0.78933	0.70310	0.61896	0.42296		0.88991	0.69566	0.81864	0.78576
	0.80135	0.62445	0.75167	0.60907		0.87209	0.68076	0.68888	0.59731
	0.80950	0.63759	0.78412	0.65494		0.81078	0.58814	0.58526	0.51893
2	0.84262	0.68290	0.61545	0.55951	7	0.87377	0.70605	0.77898	0.76604
	0.88260	0.73467	0.72755	0.70875		0.88286	0.75406	0.67184	0.71084
	0.88615	0.74325	0.72803	0.72091		0.88850	0.78149	0.72241	0.74834
	0.84970	0.66431	0.55055	0.51113		0.89573	0.74698	0.73471	0.64611
	0.85702	0.69726	0.67331	0.62884		0.86706	0.54720	0.74945	0.78970
3	0.82778	0.59188	0.63666	0.61061	8	0.84640	0.64069	0.29756	0.49776
	0.85311	0.73466	0.70027	0.60999		0.85241	0.76285	0.39483	0.57343
	0.82332	0.64362	0.72892	0.65093		0.86212	0.78463	0.51179	0.56568
	0.86402	0.72157	0.74364	0.67344		0.83666	0.71045	0.32753	0.46923
	0.84550	0.66364	0.68873	0.59072		0.85123	0.63596	0.56458	0.64083
4	0.87101	0.73092	0.62293	0.66241	9	0.88966	0.75260	0.67923	0.76581
	0.95006	0.84950	0.84079	0.88790		0.86065	0.63413	0.65928	0.74641
	0.89286	0.80254	0.77954	0.76524		0.86080	0.78236	0.80102	0.78369
	0.89972	0.76729	0.65735	0.68573		0.87720	0.78852	0.68773	0.69496
	0.87560	0.63537	0.50357	0.78738		0.88966	0.75260	0.67923	0.76581

Table 4.6 above is the normalized values for Zernike moments for digit images used in training. These normalized values are fed into network training and classification phases.

4.2.4 Recognition Results between Moment Functions

Five hundreds digit images are trained and the weights adapted are tested with two samples digits with each sample consists 50 digit images. The classification and recognition results are summarized in tables below.

Table 4.7: Recognition rates of unthinned isolated digits (0 – 9) using geometric moments, Zernike moments and contour sequence moments in feature extraction

Moment Functions	Recognition Accuracy (%)		
	Group 1	Group 2	Average
Geometric	36	40	38
Zernike	38	43	40.5
Contour Sequence	51	60	55.5

From the results obtained from Table 4.7, it can be concluded that contour sequence moments are superior compare with geometric moments and Zernike moments in representing the features of a digit image.

4.2.5 Recognition Results between Thinned and Unthinned Digit Image

In general, image thinning operation is proved tend to improve the accuracy of image especially character image. Thus, the necessity of this preprocessing technique is validated its efficiency when moment functions are used in feature extraction. Unthinned digit images are used in training stages while thinned and unthinned digit images are tested in classification to validate its effectiveness. The classification and recognition results of thinned and unthinned digit images are summarized in tables below.

Table 4.8: Recognition rates of thinned and unthinned digit 0 to 9 using geometric moments in feature extraction

Geometric Moments	Recognition Accuracy		
	Group 1	Group 2	Average (%)
Thinned Image	32	36	34
Unthinned Image	36	40	38

Table 4.9: Recognition rates of thinned and unthinned digit 0 to 9 using Zernike moments in feature extraction

Zernike Moments	Recognition Accuracy		
	Group 1	Group 2	Average (%)
Thinned Image	21	20	20.5
Unthinned Image	38	43	40.5

Table 4.10: Recognition rates of thinned and unthinned digit 0 to 9 using contour sequence moments in feature extraction

Contour Sequence Moments	Recognition Accuracy		
	Group 1	Group 2	Average (%)
Thinned Image	40	49	44.5
Unthinned Image	51	60	55.5

The features extracted from moment functions are then feed into the network established for network training. Fuzzy triangular membership function is used to deduce an initial network weights for faster network convergence purpose. The results of recognition and classification of unthinned isolated handwritten digits are unsatisfactory of having around 40% accuracy rate for Zernike moments, 38% accuracy rate for geometric moments and 55.5% using contour sequence moments. The unthinned images possessed higher recognition rates compare with thinned images by 20% for Zernike moments.

Chapter 5
Discussion and Conclusion

5.1 Introduction

The main objective of this research is to implement a neuro-fuzzy classifier on isolated handwritten digits using feature extracted from geometric moments, Zernike moments and contour sequence moment. The strengths of these three moment functions are further verified and tested to see which moment functions suitably represents an image and thus would provide a promising classification rate. Operation thinning is also validated and justified to determine its requirement in enhancing the recognition rates.

Fuzzy weight initialization is applied in the neural network with Triangular membership function and the result obtained shown that the network convergence rates is shortened. In our network established, sigmoid activation function is used in both input-to-hidden layer and hidden-to-output layer. With the implementation of image preprocessing stage, feature extraction using moment functions and neuro-fuzzy classification on isolated handwritten digits as described in Section 4, the objectives of the research has been fulfilled.

5.2 Discussion of Results

Experimentations from Section 4 has shown that Zernike moment invariants are superior to geometric moments and contour sequence moments in representing features of isolated handwritten digits with obvious interclass invariants. In terms of intraclass invariants, contour sequence moments exhibited better result with small deviation. Our network suffers from low recognition rates may due to the network is trained inappropriately resulting with high network error. However, from Table 4.8, Table 4.9 and Table 4.10, it can be observe that contour sequence moments possess higher recognition rates even though Zernike moments shown higher intraclass invariants. From these 3 tables, we also can conclude that thinning operation should be excluded as it brings down the recognition rate of isolated handwritten digits.

Triangular membership function is applied to generate initial weights for the network. Experiments results showed the improvement of this method to the convergence rate of network training compared to random method. But in most of the case, the parameter used should be appropriately set. The recognition accuracy should be higher and the network convergence will rise in theory but in our case, it deteriorates the recognition rate with introduction of ambiguity. Thus, in our case normalization operation is excluded.

Our neuro-fuzzy network suffered from low recognition rate may be due to the following reasons:

- The network architecture (2 hidden layers, with 150 neurons in the first layer and 75 neurons in the second layer) may be improperly set up.
- The maximum allow network error is high, around 0.04.
- 500 set training data is inadequate for network training.
- Standard back-propagation instinctly suffered from slow convergence, thus the number of epoch used (20000) should be increased.

5.3 Recommendation for Future Works

Below are some suggestions that could lead to the improvement of the recognition of isolate handwritten digits and some possible points that could lead to future research.

- **Utilization of other neural network model for comparison**
  Recurrent neural network or Radial Basis function can be applied in network training and classification to compare with the neuro-fuzzy classification implemented in this study.

- **Implementation in other realm where applicable**
  The methodologies used in feature extraction with moment functions and neuro-fuzzy classification may be extended to other objects such as recognition of primitives shape, handwritten character or biometric identification.

- **Utilization of others moment functions**
  There are several others moment functions could be used in feature extraction of an image. For example, weighted central moments and cross-weighted moments which can be applied in image analysis application.

5.4 Conclusion

This project presents a comparison of effectiveness between geometric moments, Zernike moments and contour sequence moments in representing the description of an image. Network training and testing (classification) is conducted using standard back-propagation with sigmoid activation where the network weights are initialized using Triangular Membership function.

In this project, operations below are conducted:

- Apply feature extraction methodologies using geometric moments, Zernike moments and contour sequence moments.
- Apply fuzzy triangular membership function in network weights initializations.
- Present comparison between geometric moments, Zernike moments and contour sequence moments in terms of image representation efficiency.
- Justify the requirement of operation thinning when geometric moments, Zernike moments and contour sequence moments are used in feature extractions.

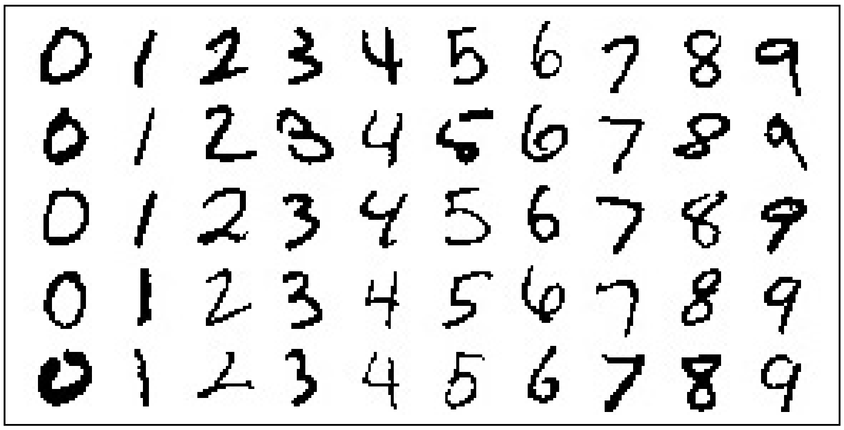
Based on the experimentations performed, it can be concluded that:

- Operation thinning conducted to a digit image will decrease the classification and recognition accuracy rate and thus can be neglected.
- Fuzzy Triangular membership function reduces neural network training duration with appropriate parameters being set correctly.
- Contour sequence moments are better in representing an image description compare with geometric moments and Zernike moments.

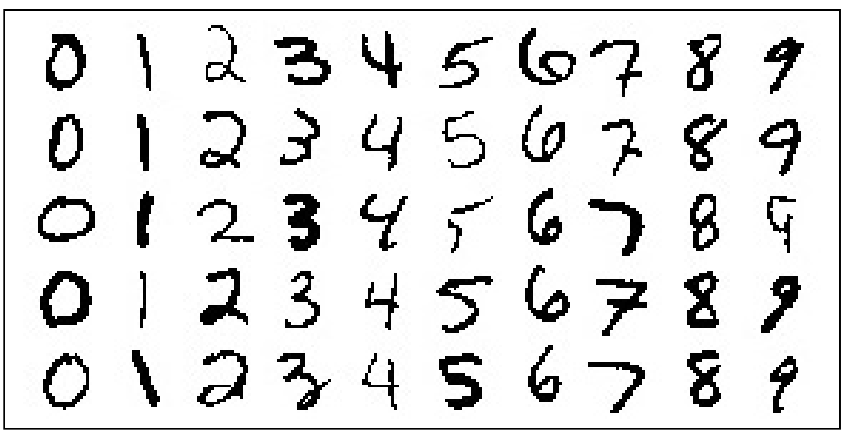
Future study is suggested in order to improve the recognition accuracy of isolated handwritten digits and to seek for any improvement in feature extraction where applicable.

Appendix A

Samples of Unthinned Isolated Handwritten Digits (Classification)

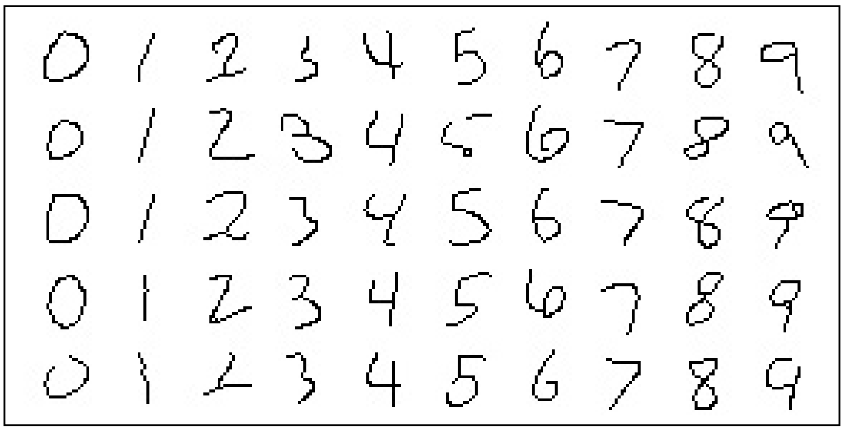


Group 1: 50 samples of unthinned isolated handwritten digits

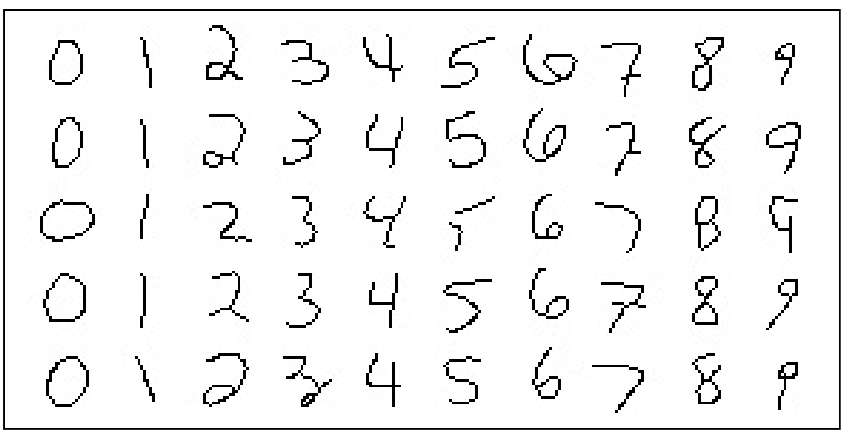


Group 2: 50 samples of unthinned isolated handwritten digits

Samples of Thinned Isolated Handwritten Digits (Classification)



Group 1: 50 samples of thinned isolated handwritten digits



Group 2: 50 samples of thinned isolated handwritten digits

Appendix B

Features Extracted using Moment Functions

Digit	φ_1	φ_2	φ_3	φ_4	Digit	φ_1	φ_2	φ_3	φ_4
	-0.7074	-1.3561	-3.0147	-3.7736		-0.8104	-1.3431	-2.5766	-3.4440
	-0.7074	-1.3561	-3.0147	-3.7736		-0.8253	-1.2754	-2.0147	-2.6959
	-0.7326	-1.2278	-2.5481	-3.2480		-0.3426	-0.1910	-1.1724	-1.4871
	-0.7173	-1.1810	-2.7285	-3.9331		-0.4494	-0.3208	-2.5107	-3.2808
0	-1.0662	-1.7378	-4.7656	-4.6430	5	-0.6012	-0.6077	-1.5816	-2.0327
	-0.8116	-1.4727	-3.5792	-4.1366		-0.4007	-0.3747	-1.3362	-1.8049
	-0.7839	-1.6027	-3.1220	-3.4337		-0.8523	-1.8275	-1.0881	-1.9997
	-0.8679	-1.5224	-2.8123	-3.6644		-1.1743	-2.0665	-1.7006	-2.3456
	-0.4038	-0.8374	-2.4424	-2.6763		-0.8444	-1.5766	-2.2075	-3.2327
	-1.0634	-2.1161	-3.8091	-4.2585		-0.4847	-0.6053	-1.1801	-1.5840
	-1.0767	-1.0839	-1.9223	-2.8574		-0.8972	-1.8901	-2.6574	-2.7163
	-1.0767	-1.0839	-1.9223	-2.8574		-0.9107	-1.7099	-3.2090	-3.5506
	-0.7461	-0.5464	-1.7271	-2.5299		-0.6815	-1.6003	-1.2751	-2.0195
	-1.3235	-1.5528	-3.5910	-3.2941		-0.7952	-1.6955	-2.4781	-2.8477
1	-1.2468	-2.0549	-2.4030	-2.4470	6	-1.1865	-2.2867	-2.9783	-3.5092
	-0.7198	-0.5531	-2.7115	-3.1067		-1.0085	-2.1359	-2.8904	-4.0043
	-0.8483	-1.6137	-1.7445	-2.5214		-1.0774	-2.0855	-2.6740	-3.4141
	-1.3332	-1.8785	-2.8153	-2.9025		-0.9532	-1.7364	-2.5517	-3.3932
	-0.7787	-0.5663	-2.4411	-3.3238		-0.9655	-1.7362	-2.9911	-3.5867
	-0.9224	-0.9766	-1.7752	-3.1600		-0.8658	-1.6349	-2.8125	-3.5998
	-0.9833	-1.6818	-2.7193	-3.3165		-0.7845	-1.5340	-1.4010	-2.2726
	-0.7281	-1.3514	-1.7667	-2.6009		-0.7264	-1.2276	-1.7534	-2.9565
	-0.7054	-1.2966	-1.6890	-2.5979		-0.6905	-1.0525	-1.5139	-2.6337
	-0.9833	-1.6818	-2.7193	-3.3165		-0.6443	-1.2727	-2.1665	-2.5552
2	-0.9382	-1.8005	-3.0281	-3.7308	7	-0.8273	-2.5480	-1.2499	-2.4611
	-0.7625	-1.3089	-2.4396	-2.9206		-0.7397	-1.5523	-1.4712	-2.3406
	-0.9647	-1.9641	-2.5168	-3.3547		-0.6632	-0.9113	-1.1291	-1.9779
	-0.5837	-1.0801	-1.3713	-2.1161		-0.7589	-0.9915	-1.5373	-2.3580
	-0.8220	-1.6659	-2.9032	-3.7233		-0.6230	-1.1436	-0.8042	-1.5836
	-0.8510	-1.5571	-2.1264	-2.7043		-0.6692	-1.1075	-1.4586	-2.1260
	-1.0781	-2.2628	-2.3932	-3.1811		-0.9592	-1.9512	-3.1135	-5.3457
	-0.9164	-1.3514	-2.3971	-2.7751		-0.9209	-1.1714	-2.6305	-4.7248
	-1.1065	-1.9325	-2.1358	-2.5921		-0.8588	-1.0324	-2.6799	-3.9782
	-0.8467	-1.4350	-1.9921	-2.4982		-1.0214	-1.5059	-3.2956	-5.1544
3	-0.8467	-1.4350	-1.9921	-2.4982	8	-0.9283	-1.9814	-2.2003	-3.6412
	-0.6658	-0.9387	-2.1779	-2.6216		-0.9333	-1.2610	-2.4577	-4.2577
	-0.7734	-1.0848	-1.4636	-2.2368		-0.8656	-1.1317	-2.1415	-3.0203
	-0.9450	-1.3895	-2.1868	-2.6822		-0.9964	-1.4573	-2.4366	-4.7163
	-0.8150	-1.7917	-1.5044	-2.1742		-0.9366	-1.3047	-2.4743	-5.2764
	-1.0657	-2.0714	-2.0661	-2.8969		-1.0444	-1.4320	-3.1417	-3.9903
	-0.8021	-1.3753	-2.0625	-3.2688		-0.6830	-1.2369	-1.4024	-2.9094
	-0.2975	-0.6184	-0.6231	-1.8780		-0.8682	-1.9931	-1.5263	-3.0367
	-0.6626	-0.9181	-1.4061	-2.2690		-0.8673	-1.0469	-1.2883	-2.1319
	-0.6188	-1.1431	-1.9137	-3.0491		-0.7626	-1.0076	-1.8547	-2.8551
4	-0.7728	-1.9852	-1.2647	-4.0306	9	-0.7772	-0.8557	-1.0954	-1.7664
	-0.7665	-1.2882	-1.2513	-2.1662		-0.9665	-1.3864	-1.6519	-2.5869
	-0.8767	-1.7289	-1.9645	-4.1572		-0.8962	-1.7532	-2.2395	-4.1992
	-0.4719	-0.9602	-2.4697	-3.2661		-0.9702	-1.2856	-2.2504	-2.8114
	-0.7388	-0.9386	-1.9348	-3.7228		-0.6692	-1.4856	-1.3002	-2.7051
	-0.6369	-1.2926	-2.0324	-3.1878		-0.8848	-1.4078	-1.8993	-2.9066

Sample of Features Extracted using Geometric Moments

Digit	Z_{20}	$\lvert Z_{22}\rvert^2$	$\lvert Z_{31}\rvert^2$	$\lvert Z_{33}\rvert^2$	Digit	Z_{20}	$\lvert Z_{22}\rvert^2$	$\lvert Z_{31}\rvert^2$	$\lvert Z_{33}\rvert^2$
	-0.4264	-1.3961	-2.6096	-2.8049		-0.5294	-1.3831	-2.2799	-2.3668
	-0.4264	-1.3961	-2.6096	-2.8049		-0.5443	-1.3155	-1.5318	-1.8048
	-0.4516	-1.2679	-2.0839	-2.3383		-0.0616	-0.2311	-0.3230	-0.9626
	-0.4363	-1.2211	-2.7690	-2.5187		-0.1684	-0.3608	-2.1167	-2.3009
0	-0.7852	-1.7778	-3.4790	-4.5558	5	-0.3202	-0.6478	-0.8687	-1.3718
	-0.5306	-1.5128	-2.9725	-3.3694		-0.1197	-0.4148	-0.6408	-1.1263
	-0.5029	-1.6427	-2.2696	-2.9122		-0.5713	-1.8676	-0.8356	-0.8783
	-0.5869	-1.5624	-2.5004	-2.6025		-0.8933	-2.1065	-1.1815	-1.4908
	-0.1228	-0.8775	-1.5122	-2.2325		-0.5634	-1.6167	-2.0687	-1.9977
	-0.7824	-2.1562	-3.0945	-3.5993		-0.2037	-0.6454	-0.4199	-0.9702
	-1.0425	-1.5929	-2.1300	-3.3812		-0.6162	-1.9302	-1.5522	-2.4476
	-0.7957	-1.1240	-1.6934	-1.7125		-0.6297	-1.7499	-2.3866	-2.9992
	-0.4651	-0.5865	-1.3659	-1.5173		-0.4005	-1.6404	-0.8554	-1.0653
	-1.0425	-1.5929	-2.1300	-3.3812		-0.5142	-1.7355	-1.6837	-2.2682
1	-0.9658	-2.0950	-1.2829	-2.1932	6	-0.9055	-2.3267	-2.3451	-2.7685
	-0.4388	-0.5931	-1.9426	-2.5016		-0.7275	-2.1759	-2.8402	-2.6806
	-0.5672	-1.6537	-1.3574	-1.5347		-0.7964	-2.1255	-2.2501	-2.4642
	-1.0522	-1.9186	-1.7384	-2.6054		-0.6722	-1.7765	-2.2292	-2.3419
	-0.4977	-0.6064	-2.1597	-2.2313		-0.6845	-1.7762	-2.4227	-2.7813
	-0.6414	-1.0166	-1.9959	-1.5654		-0.5848	-1.6749	-2.4358	-2.6026
	-0.7023	-1.7219	-2.1524	-2.5095		-0.5035	-1.5741	-1.1086	-1.1912
	-0.4471	-1.3914	-1.4369	-1.5568		-0.4454	-1.2676	-1.7925	-1.5435
	-0.4244	-1.3366	-1.4338	-1.4792		-0.4094	-1.0925	-1.4697	-1.3041
	-0.7023	-1.7219	-2.1524	-2.5095		-0.3633	-1.3128	-1.3911	-1.9567
2	-0.6572	-1.8406	-2.5667	-2.8183	7	-0.5463	-2.5881	-1.2971	-1.0401
	-0.4815	-1.3490	-1.7566	-2.2298		-0.4587	-1.5924	-1.1766	-1.2614
	-0.6837	-2.0042	-2.1906	-2.3070		-0.3822	-0.9514	-0.8138	-0.9193
	-0.3027	-1.1202	-0.9520	-1.1614		-0.4779	-1.0316	-1.1939	-1.3275
	-0.5410	-1.7060	-2.5592	-2.6933		-0.3420	-1.1836	-0.4196	-0.5944
	-0.5700	-1.5972	-1.5402	-1.9165		-0.3882	-1.1475	-0.9619	-1.2487
	-0.7971	-2.3029	-2.0170	-2.1833		-0.6782	-1.9913	-4.1816	-2.9037
	-0.6354	-1.3915	-1.6110	-2.1873		-0.6399	-1.2115	-3.5607	-2.4207
	-0.8255	-1.9726	-1.4281	-1.9260		-0.5778	-1.0725	-2.8141	-2.4701
	-0.5657	-1.4750	-1.3342	-1.7823		-0.7404	-1.5460	-3.9903	-3.0858
3	-0.5657	-1.4750	-1.3342	-1.7823	8	-0.6473	-2.0215	-2.4771	-1.9904
	-0.3848	-0.9787	-1.4575	-1.9681		-0.6523	-1.3011	-3.0936	-2.2479
	-0.4924	-1.1249	-1.0728	-1.2538		-0.5846	-1.1718	-1.8562	-1.9317
	-0.6640	-1.4296	-1.5181	-1.9769		-0.7154	-1.4973	-3.5523	-2.2267
	-0.5340	-1.8317	-1.0102	-1.2945		-0.6556	-1.3448	-4.1124	-2.2645
	-0.7847	-2.1114	-1.7328	-1.8563		-0.7633	-1.4720	-2.8262	-2.9318
	-0.5211	-1.4153	-2.1047	-1.8527		-0.4020	-1.2769	-1.7453	-1.1926
	-0.0165	-0.6584	-0.7140	-0.4133		-0.5872	-2.0332	-1.8727	-1.3164
	-0.3816	-0.9582	-1.1050	-1.1963		-0.5863	-1.0870	-0.9679	-1.0785
	-0.3378	-1.1832	-1.8850	-1.7038		-0.4816	-1.0477	-1.6910	-1.6449
4	-0.4918	-2.0253	-2.8666	-1.0549	9	-0.4962	-0.8957	-0.6024	-0.8855
	-0.4855	-1.3283	-1.0022	-1.0415		-0.6855	-1.4265	-1.4228	-1.4421
	-0.5957	-1.7689	-2.9931	-1.7546		-0.6152	-1.7932	-3.0351	-2.0297
	-0.1909	-1.0003	-2.1021	-2.2598		-0.6892	-1.3256	-1.6474	-2.0406
	-0.4578	-0.9787	-2.5588	-1.7249		-0.3882	-1.5256	-1.5410	-1.0904
	-0.3559	-1.3327	-2.0238	-1.8226		-0.6038	-1.4478	-1.7425	-1.6895

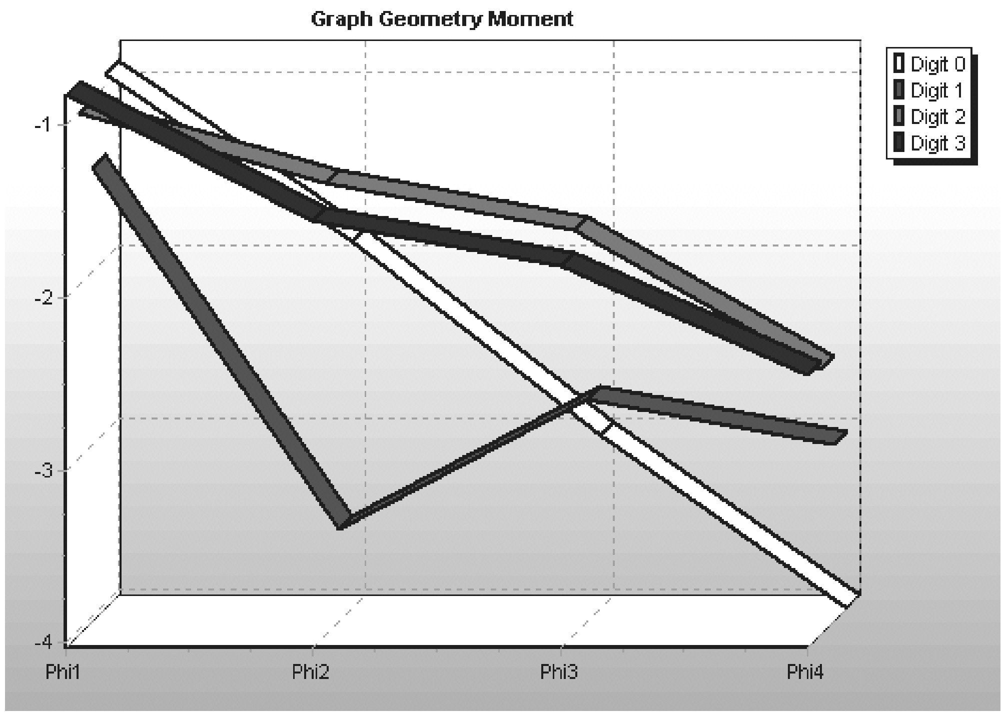
Sample of Features Extracted using Zernike Moments

Digit	F_1	F_2	F_3	F_4	Digit	F_1	F_2	F_3	F_4
	50.2167	3.8718	16.1974	71.4888		36.2630	5.3922	31.4780	194.2048
	32.7182	5.9536	38.3249	260.4306		34.7145	5.6083	34.0012	217.5718
	31.6977	6.1323	40.6296	283.9786		47.6684	4.0988	18.1830	85.2292
	31.0589	6.2924	42.8580	308.4552		44.6780	4.3563	20.5125	101.9353
0	29.8657	6.5213	45.9780	342.1824	5	36.8113	5.3132	30.5652	185.8412
	27.9748	6.9642	52.4418	416.8878		44.3675	4.3929	20.8691	104.6745
	28.0678	6.9498	52.2466	414.8044		43.7432	4.4601	21.5202	109.6651
	27.2098	7.1757	55.7180	457.0343		43.0862	4.5132	22.0100	113.2482
	36.3890	5.3486	30.9218	188.6660		34.9214	5.6003	33.9567	217.6072
	28.2377	6.9106	51.6654	407.9751		42.5045	4.5763	22.6326	118.1067
	47.2617	4.1319	18.4748	87.2680		45.0185	4.3244	20.2147	99.7348
	45.1121	4.3441	20.4452	101.7592		34.3768	5.6480	34.4519	221.6177
	44.7041	4.3619	20.5790	102.5231		51.2294	3.8040	15.6481	67.9602
	54.6110	3.5838	13.9087	57.0631		43.7343	4.4581	21.4958	109.4442
1	57.1824	3.4168	12.6354	49.3708	6	32.4959	6.0158	39.1760	269.6055
	45.7394	4.2631	19.6566	95.7066		35.2245	5.5427	33.2431	210.6178
	39.4226	4.9323	26.2862	147.8011		33.0096	5.9162	37.8774	256.1920
	47.6489	4.0952	18.1433	84.8985		33.5264	5.8065	36.4460	241.4431
	44.0236	4.4203	21.1194	106.4844		35.8625	5.4283	31.8524	197.2657
	49.7931	3.9155	16.5815	74.1468		33.6820	5.7874	36.2226	239.3785
	32.0893	6.0743	39.9025	276.7614		41.9646	4.6435	23.3163	123.6040
	34.3992	5.6565	34.5807	223.0917		36.7353	5.2893	30.2222	182.1493
	36.0624	5.4107	31.6710	195.7990		48.0012	4.0498	17.7208	81.8023
	29.2992	6.6570	47.9353	364.5183		40.8594	4.7530	24.4011	132.1143
2	28.7609	6.7885	49.8657	386.9464	7	50.7548	3.8445	15.9904	70.2448
	36.9143	5.2598	29.8798	179.0006		35.5788	5.4991	32.7447	206.0997
	29.3973	6.6374	47.6605	361.4563		39.6699	4.9078	26.0377	145.8020
	38.9633	4.9866	26.8622	152.6314		38.1145	5.1010	28.1153	163.4865
	28.8922	6.7220	48.8050	373.7141		42.4988	4.5752	22.6177	117.9658
	29.4886	6.6141	47.3190	357.5067		36.5995	5.3312	30.7484	187.3071
	35.7312	5.4676	32.3541	202.2851		34.0218	5.7567	35.8969	236.6769
	30.5943	6.3701	43.8806	319.1389		34.8529	5.5763	33.5942	213.4949
	34.8568	5.5777	33.6155	213.7359		32.3283	6.0229	39.2160	269.5122
	30.4797	6.3886	44.1232	321.6571		33.8005	5.7535	35.7700	234.6381
3	27.2030	7.1719	55.6432	455.9419	8	38.9541	4.9839	26.8263	152.2684
	35.6907	5.4540	32.1545	200.0724		31.9914	6.1079	40.3795	282.0665
	43.7070	4.4522	21.4245	108.7969		35.5108	5.4761	32.4047	202.3144
	32.9829	5.9057	37.7101	254.1864		35.5157	5.4778	32.4292	202.5867
	42.2291	4.6096	22.9680	120.7817		34.3958	5.6552	34.5617	222.8731
	35.8475	5.4232	31.7782	196.4486		35.6687	5.4466	32.0458	198.8678
	36.9260	5.2636	29.9335	179.5740		34.0949	5.7046	35.1669	228.7455
	40.6442	4.7925	24.8329	135.8336		38.3245	5.0736	27.8142	160.8754
	39.2642	4.9797	26.8460	152.9527		43.7346	4.4582	21.4971	109.4558
	31.9330	6.0834	39.9766	277.0798		47.2262	4.1252	18.4001	86.6410
4	48.0582	4.0604	17.8362	82.7533	9	40.4371	4.8328	25.2811	139.7422
	44.7382	4.3692	20.6644	103.2793		43.3915	4.4822	21.7106	110.9565
	31.5944	6.1620	41.0451	288.5620		36.9289	5.2646	29.9473	179.7212
	33.5127	5.8013	36.3648	240.4851		45.0354	4.3279	20.2561	100.0986
	43.4204	4.4887	21.7896	111.6767		38.3274	5.0745	27.8263	160.9994
	36.2546	5.3895	31.4383	193.7707		39.9174	4.8820	25.7731	143.6511

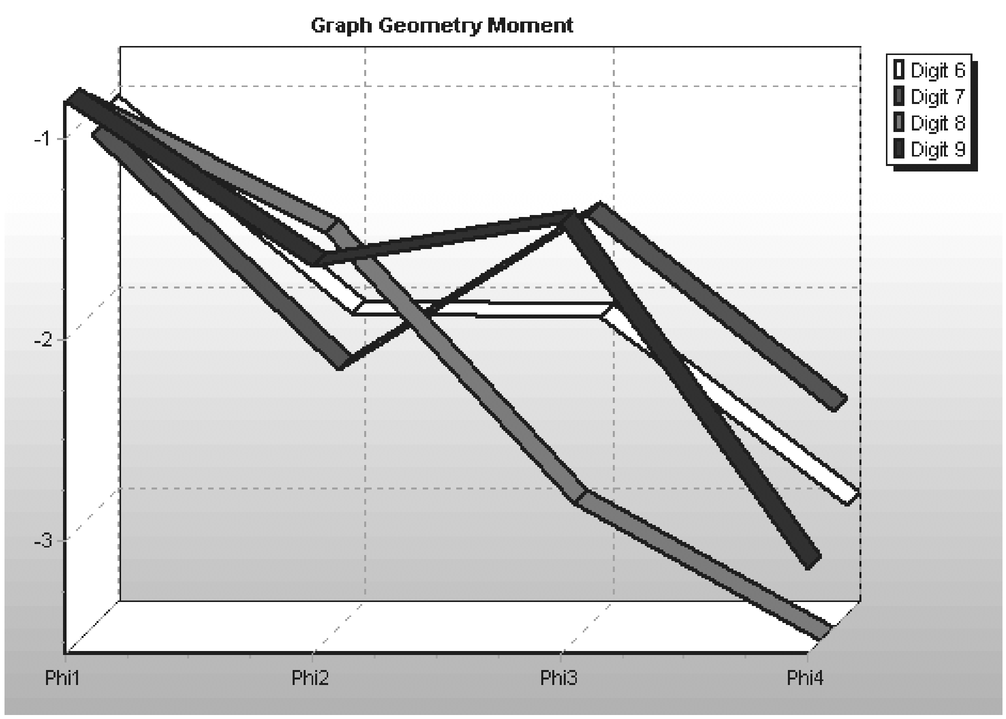
Sample of Features Extracted using Contour Sequence Moments

Appendix C

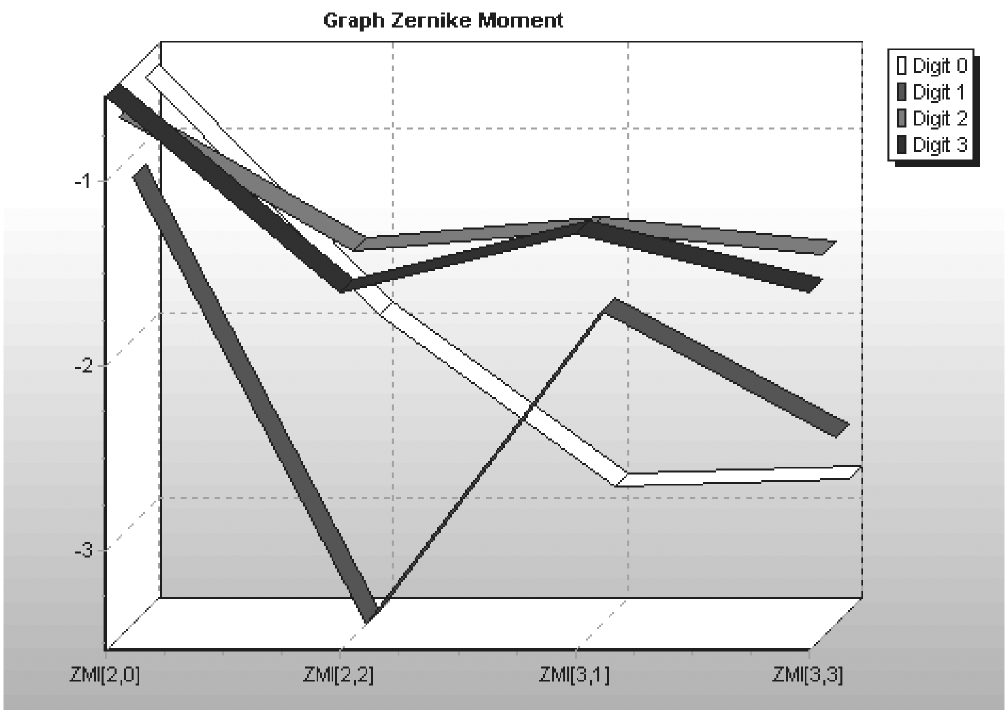
Intraclass and Interclass Invariants between Moment Functions



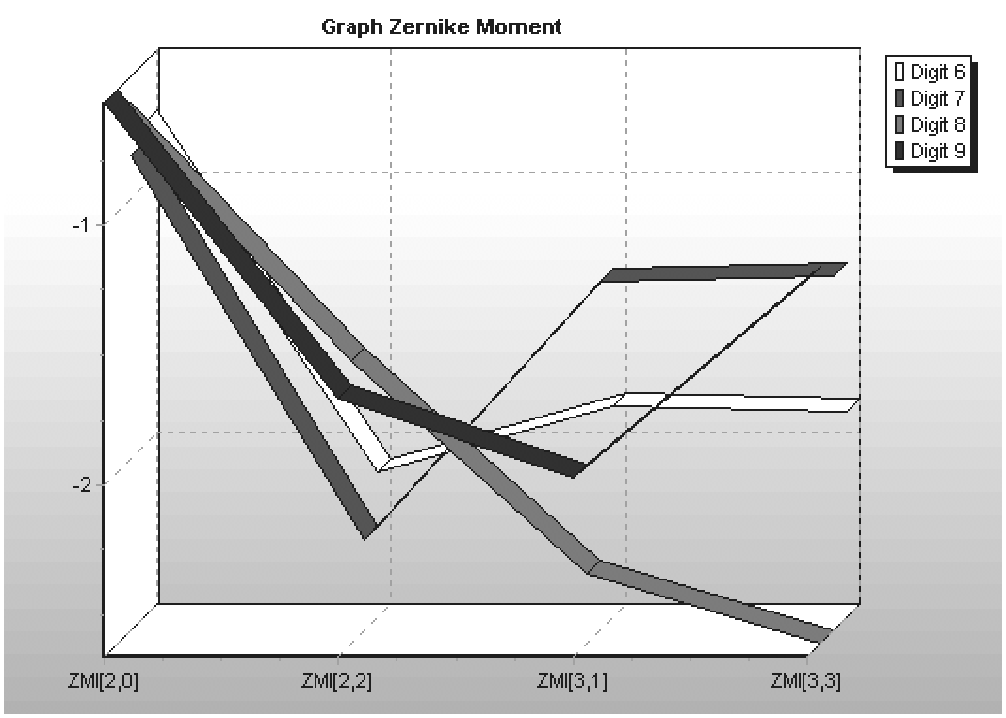
Interclass invariants between digits 0, 1, 2 and 3 using geometric moments' features.



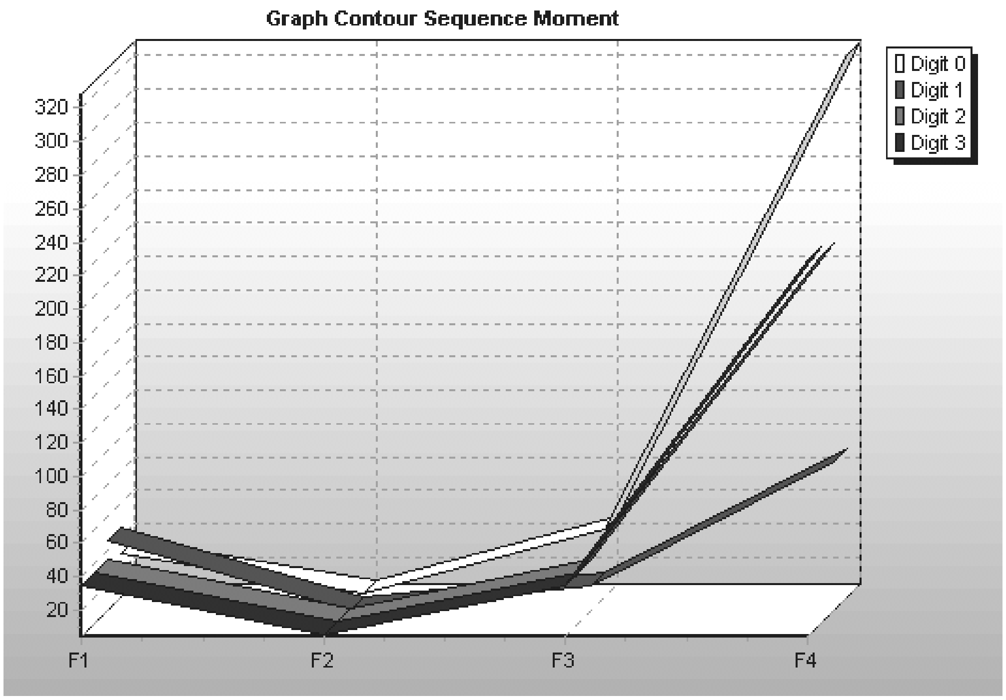
Interclass invariants between digits 6, 7, 8 and 9 using geometric moments' features.



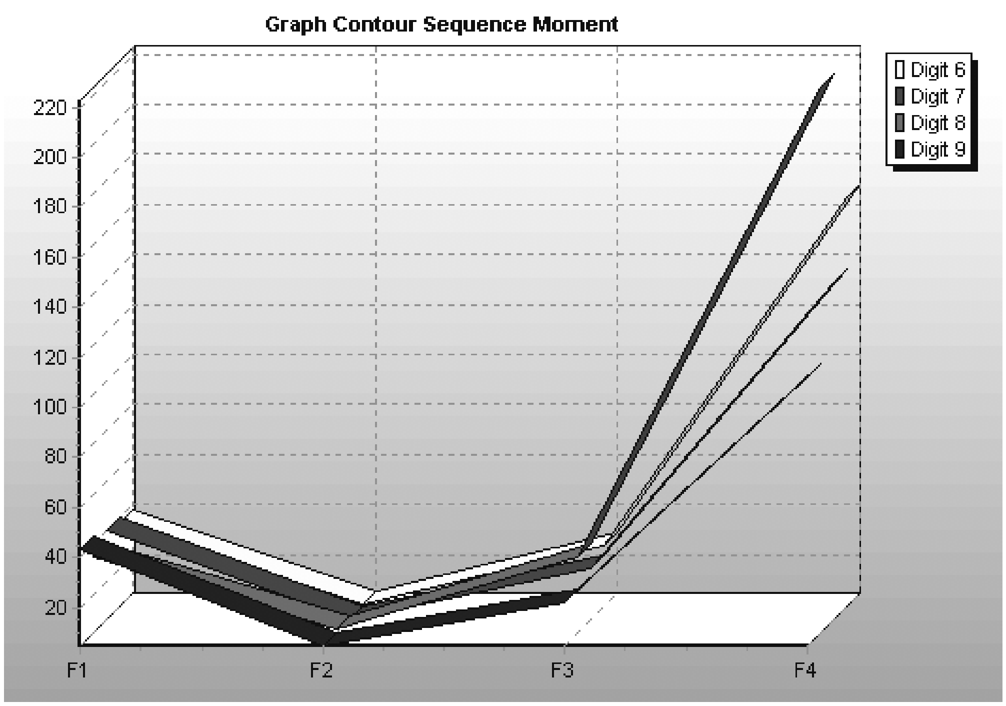
Interclass invariants between digits 0, 1, 2 and 3 using Zernike moments' features.



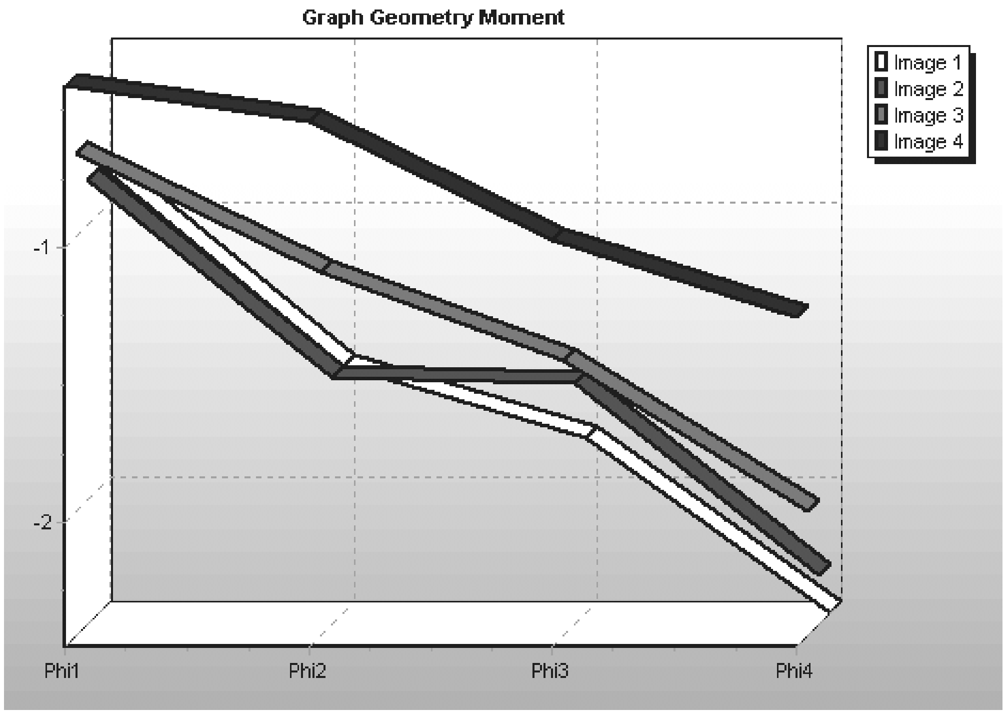
Interclass invariants between digits 6, 7, 8 and 9 using Zernike moments' features.



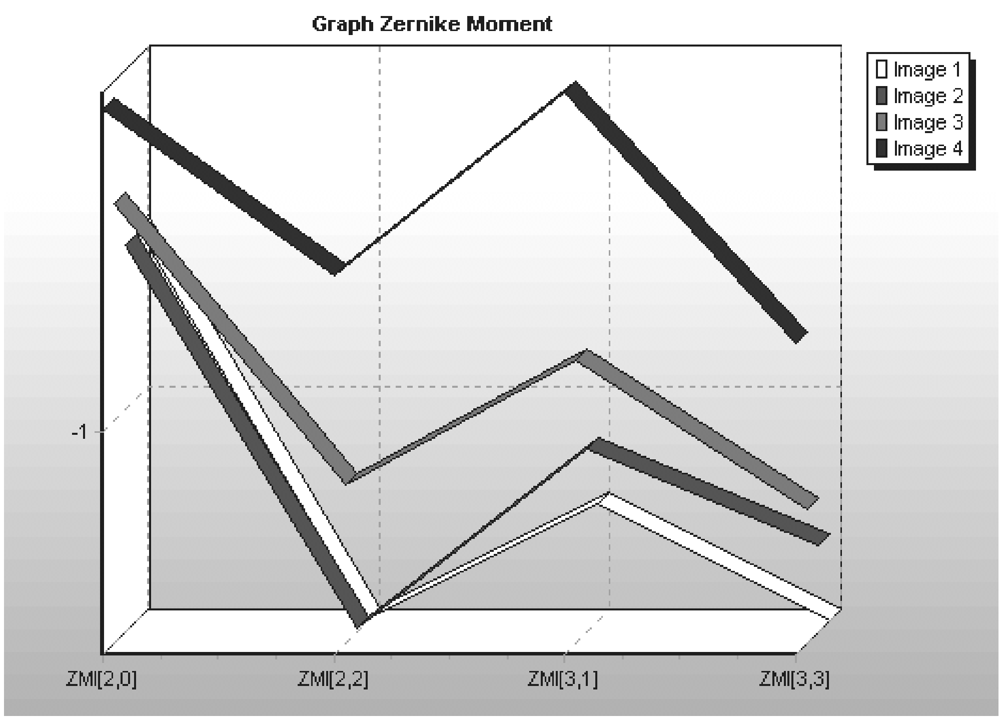
Interclass invariants between digits 0, 1, 2 and 3 using contour sequence moments' features.



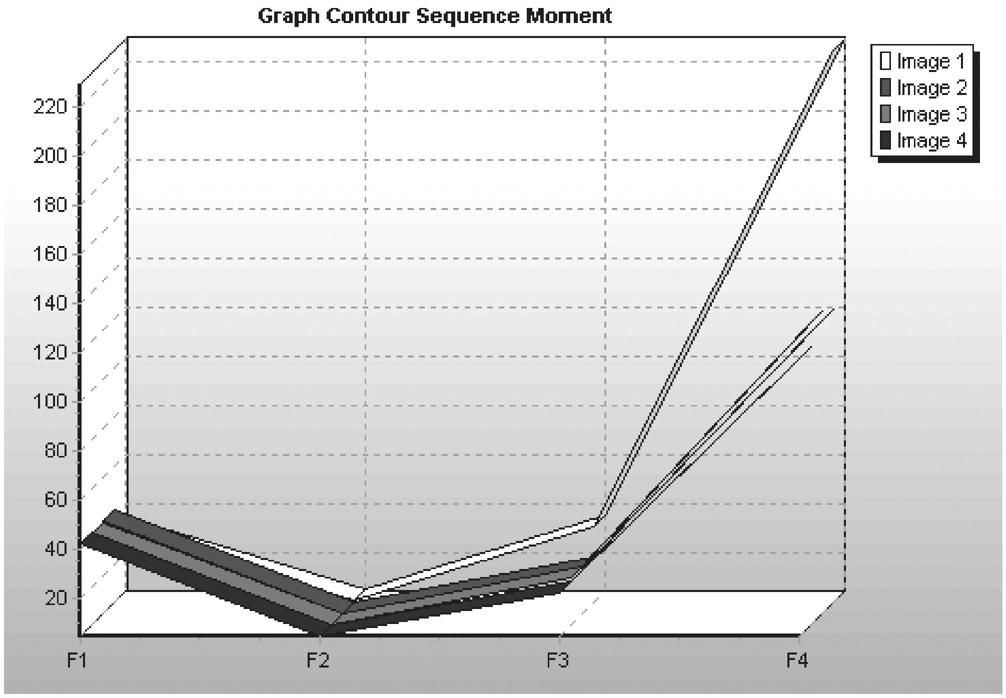
Interclass invariants between digits 6, 7, 8 and 9 using contour sequence moments' features.



Intraclass invariants for 4 samples of digit 3 using geometric moments' features.



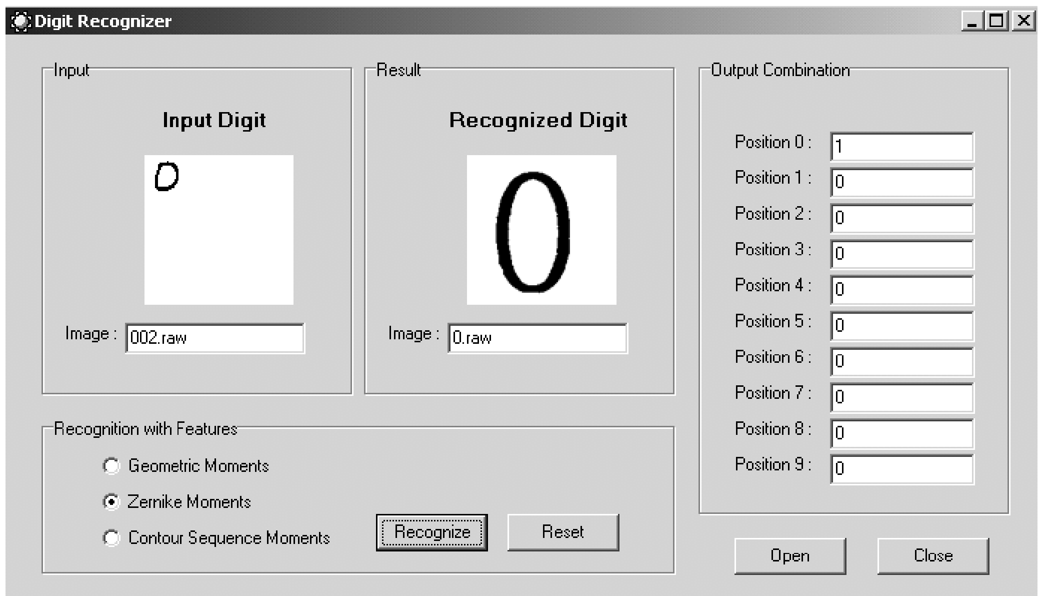
Intraclass invariants for 4 samples of digit 3 using Zernike moments' features.



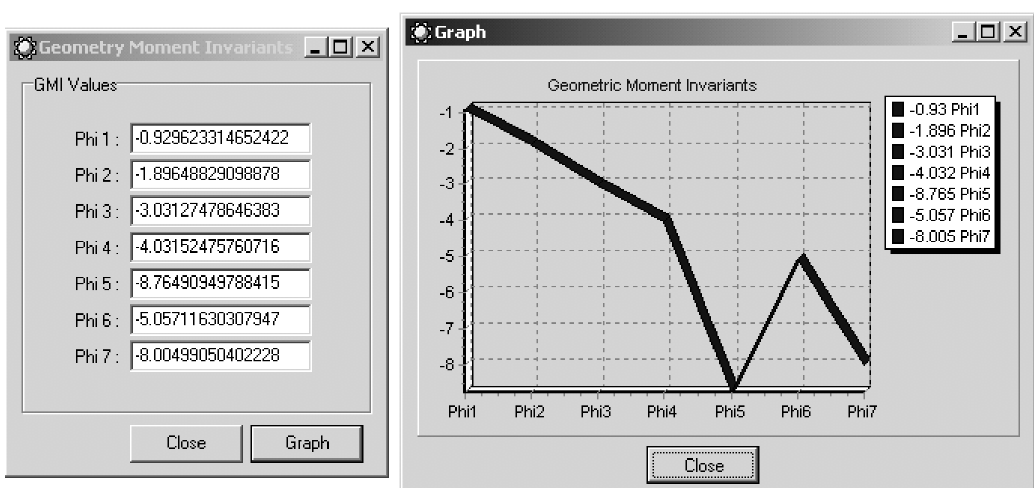
Intraclass invariants for 4 samples of digit 3 using contour sequence moments' features.

Appendix D

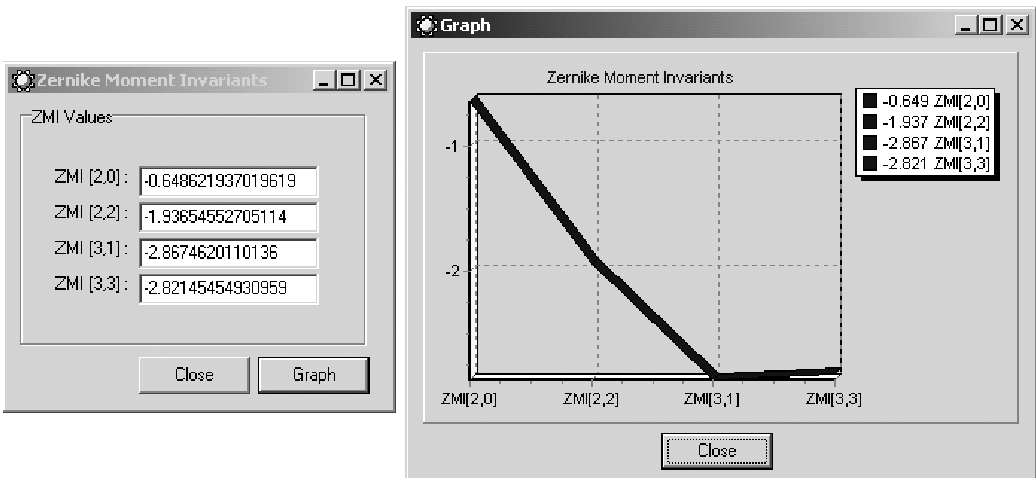
Interface of Prototype System



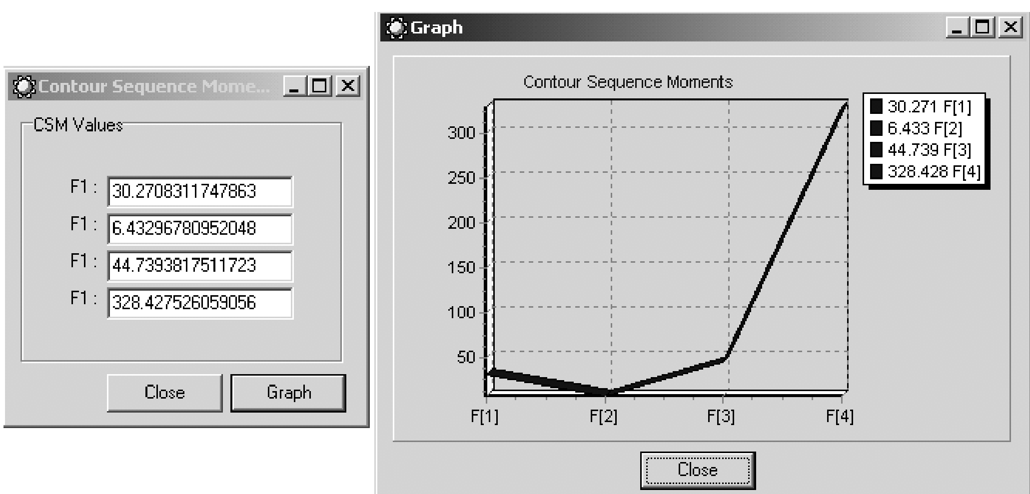
User interface for digit recognition prototype



User interface for computation of geometric moments and graph viewing



User interface for computation of Zernike moments and graph viewing



User interface for computation of contour sequence moments and graph viewing