

# **EC386 - MINI PROJECT IN IMAGE PROCESSING**

**Project Title:**

**SIGNATURE VERIFICATION USING IMAGE PROCESSING**



**Project Mentor(s):**

**Dr.DEEPU V S**

**Submitted By:**

**1.NAME: VIVEK N C (Roll.No. 201EC167)**

**2.NAME: GANESH S G (Roll.No. 201EC121)**

## **TABLE OF CONTENTS**

<b>Sl.No</b>	<b>TOPIC</b>	<b>Page No.</b>
<b>01.</b>	<b>INTRODUCTION</b>	<b>03</b>
<b>02.</b>	<b>PROBLEM STATEMENT</b>	<b>03</b>
<b>03.</b>	<b>MOTIVATION</b>	<b>03</b>
<b>04.</b>	<b>METHODOLOGY</b>	<b>03</b>
<b>05.</b>	<b>RESULTS</b>	<b>13</b>
<b>06.</b>	<b>SCOPE FOR IMPROVEMENT</b>	<b>13</b>
<b>07.</b>	<b>REFERENCES</b>	<b>11</b>

## **INTRODUCTION:**

Signature verification is a biometric verification which is an important research area targeted at automatic identity verification applications such as legal, banking and other high security environments. Such applications need their own exclusive software for signature verification. Biometrics based authentication systems are better in terms of security than traditional authentication techniques such as passwords etc. It is due to the fact that biometric characteristics of every person are unique and cannot be lost, stolen or broken. There are two types of biometrics: Behavioural and Physiological. Handwriting, speech etc. come under behavioural biometrics. Iris pattern, fingerprint etc. are part of physiological biometrics.

There are two methods for signature verification: Offline and Online, which depends on the signature acquisition method. Offline signature verification, after having complete signature on the paper, it can be acquired from scanners or cameras. In online method, during signing process, it can be acquired in parallel with digitizing tablets or any other special hardware. The purpose of signature verification is to classify the input signature as genuine or forge by matching it against the database signature image using some distance measure. Forgery means that an individual is trying to make false signatures of any other individual to become authenticated.

## **PROBLEM STATEMENT:**

Every person has a unique signature that is used primarily for personal identification and verification of important documents or legal transactions. Mostly used to authenticate checks, draughts, certificates, approvals, letters, and other legal documents. Because a signature is used in such critical activities, verification of its authenticity is essential.

## **MOTIVATION:**

Even in today's age signature is the most important and most used form of proof of identity of a person. Signatures are used for all documents in financial services, legal services or on letters, documents, checks, security documents. Thus to reducing illegal acts like forgery are a great concern in our day and age. Negligence of signature verification is one of the main cause of fraud in our country.

This was the motivation for our project.

## **METHOD OF APPROACH:**

It has been observed that an offline signature verification process consists of following steps:

1. Signature acquisition
2. Signature Pre-processing
3. Feature Extraction
4. Signature Verification

### **1) SIGNATURE ACQUISITION:**

Signature database plays a significant role in the process of signature verification. But there is no standard offline signature database for the researchers. The major setback faced by the researchers in offline signature verification is the non-availability of a sufficiently large signature database with required quality. Because of

privacy issues, not many people agree to make their signatures available to others and that too for practicing forgeries. Therefore, it is still not easy to develop a sufficiently large database with quality signatures. Some limited numbers of offline signature databases are available. But their sizes are smaller and they lack in quality.

In pattern classification, there are two phases - (i) training and (ii) testing. In training phase, the signature classifier is modeled. Sufficient numbers of signature samples are required to model the classifier. The larger database represents a population properly and it helps in producing more reliable results. In most of the pattern recognition techniques, large datasets help in achieving better results. A small training set may result unsatisfactory verification performance. Lesser number of signature samples against high number of extracted features is a snag in signature verification.

Signature images can be acquired with a digital scanner or digital camera. Digital scanner is preferable over a camera for acquiring a document image because of the following reasons:

**(i) Lighting:** A scanner has its own lighting system for illuminating the object (document). This always maintains a constant light in the image. But in case of a camera, ambient light condition varies and hence it affects the image and results inconsistent images.

**(ii) Focus:** A scanner always focuses sharply on its object because the flat bed is at fixed position. But a camera may not focus on the document as good as a scanner and there is always a chance of tilt. But a scanner positions the document perfectly flat.

**(iii) Quality:** In case of a document, the image quality of a scanner is always better than that of a camera.

### **Scanner Resolution required for offline signature**

If a signature is scanned with a higher resolution, the image is sampled at a higher sampling rate. Thus, size of the scanned image becomes larger. It occupies more storage space. On the other hand, an image scanned with low resolution takes lesser space but suffers from information loss due to low sampling rate.. They investigated the performance of an offline signature verification system using Hidden Markov Model with signature image resolutions ranging from 45dpi to 600dpi. They found that 150dpi was the appropriate resolution for acquisition of signature image in their experiment.

### **Signature Collection methods used in developing our signature databases:**

While collecting the genuine signature samples, following protocols were followed:

(i) All the signatures were collected on white A4 size paper.

(ii) Different pens were used by the signer.

(iii) Signatures were collected in seven different sessions in a span of nine days. Some signatures were collected in the morning, some in the noon, some in the evening and some were collected in the late night.

(iv) Signature images were scanned by a digital scanner with 200dpi resolution.

For producing the forged signatures, the forgers were provided the genuine signatures. To reproduce the signatures, they were allowed to practice the signature as long as they wished. Like the genuine signature samples, the forged signature samples were also collected in an A4 size white paper and scanned with 200dpi. All scanned signature images were stored in JPEG format.

**Some available offline signature databases:**

- i) GPDS-39
- ii) GPDS-100
- iii) GPDS-160
- iv) GPDS-960
- v) MCYT-75

**2) Pre-processing:**

After capturing the signature samples, the next step is to enhance the images and make them ready for the subsequent processing. That is the scanned images need to be pre-processed before giving them to the next process. Pre-processing is done using signal processing algorithms. Pre-processing greatly helps to improve the performance of feature extraction and classification. It reduces computational cost in classification. Depending on the type of signature pattern, signature image quality and classification techniques to be used, pre-processing operations are determined. It must be kept in mind that during pre-processing, information from the images should not be discarded. Loss of information in pre-processing will affect the overall accuracy of the signature verification system.

**These are the pre-processing steps were used in our project :**

**i)Resizing:**

Signature lengths are different for different signers. Even the lengths of the signatures of a single person are also not equal. But when a grid based signature verification approach is used, the signatures are projected on the grid of same size. Hence, all the signatures must be of same size. Therefore in that case, resizing of signature becomes important. The most basic method of image resizing is a kind of geometric transformation. In this method, there are two basic operations: (i) spatial transformation and (ii) gray level interpolation.

In spatial transformation, some pixels or points ('tie-points') are selected whose positions in the original image and the resized image are precisely known. From their locations in the two images, a spatial transformation equation is formulated. This equation is used as a mapping equation to find out the positions of all the pixels in the new resized image.

Gray level interpolation is used to assign gray levels to the new pixels in the resized image. It uses a nearest neighbour approach. In this method, gray level is assigned according to the pixel which is the nearest to the mapped pixel.

## **ii) Filtering:**

A scanned signature image may contain noise. Noise in the image deteriorates the feature extraction and its successive processes. Hence, filtering of noise is an unavoidable pre-processing step in pattern recognition. It has been observed that the scanned images are usually affected by salt-pepper noise. A median filter effectively removes such type of noise preserving the edges of the images. We applied a median filter of  $3 \times 3$  window on our signature images.

The median filter is a non-linear spatial filter that uses a sub-image area or window. This window is usually of square shape and is of fixed size. This window slides over complete image pixel by pixel and replaces the center value in the window with the median of all the pixel values in the window. As an example, consider the pixel value of the window in figure, in ascending order is 1, 2, 4, 8, 9, 10, 10, 12, 91. So, the median is (the middle value of the string) 5. When the center value in the window is 87 which is possibly a noise, is replaced with the median value 5, the following new window is next, where the noise is removed.

**Pixel values before filtering:**

1	2	4
10	91	12
8	9	10

**Pixel values after filtering:**

1	2	4
10	9	12
8	9	10

## **iii) Binarization:**

A colour image comprises of three colour plans Red (R), Green (G) and Blue (B). In a colour image, every pixel value is defined by the combination of the values of these three plans. In a gray level image, there is no colour information. The image is defined by the pixel values of a single plan (the intensity plan). However, in a gray level image, the pixel values will have a range which is specified by the number of bits of the image. Eg. for an 8-bit image the pixel values will range from 0 to 255. When the pixel values of a gray level image is assigned with only two values, a binary image is resulted. Thus, image size is greatly reduced in a binary image as compared to its original colour image or the gray level image.

There are two steps to convert a colour (RGB) image into a binary image,

### **1) Conversion of Colour image into gray level image:**

There are several algorithms for converting a colour image into a gray level image. In our project we have used the below method.

#### **Standard NTSC (National Television System Committee) conversion Method:**

Human perception is taken into account in this method. It is weighted average method. This method is a standard method accepted by NTSC and is widely used. MATLAB Image Processing Toolbox uses this method for converting a colour image into a gray level image. Human brain is more sensitive to green colour

than red and it is least sensitive to blue colour. Accordingly in this method, different weights are given to these colours.

$$I = 0.2989 * R + 0.587 * G + 0.114 * B$$

## **2) Conversion of Gray level image into binary image:**

A suitable threshold value (pixel value) is considered to convert a gray level image into a binary image. If a pixel value in the gray level image is greater than the threshold value then the new pixel value assigned is 1 else 0. Thus, the new image will have only two-pixel values '1' (which corresponds to white) and '0' (which corresponds to black).

### **iv) Thinning:**

In thinning, the signature image strokes are made one pixel thick. Thinning is mainly done to reduce the amount of data in the image. This helps to decrease the storage space requirement and also to reduce the computational complexities in successive stages. But during thinning, some information of the signature images such as stroke width may be lost. So, depending on the features to be extracted, thinning may or may not be required.

### **v) Skeletonization:**

Skeletonization is applied on binary images. It preserves the connectivity of the signature segments which were originally connected and removes selected foreground pixels from the image. After skeletonization, the signature image is converted into combination of some thin arcs and curves. One basic way of performing skeletonization is by using a morphological thinning process that successively removes pixels from the boundary. This process continues until no more thinning is possible.

### **(vi) Rotation for Skew Correction:**

Many a times, it is seen that during scanning of the signature images, the images are not properly oriented. This angular tilt in the signature image is called 'skew'. Skew may result poor classification (depending on the classification technique used). Therefore, there may be a need for skew correction of the signature images by rotating them. After skew correction, the final image is made parallel to the horizontal axis.

In one method of skew correction, the bottom pixels of the signature image are used. A straight line is fitted through the bottom pixels by using polynomial curve fitting method. Skew angle of this line is measured and then skew correction is done by rotating the signature image by a negative skew angle.

Following are the steps of another method of skew correction:

1. Move the signature to origin by using the co-ordinates of centre of mass of the signature image.
2. Calculate the minimum Eigen Value of the matrix formed by using the new co-ordinates of the signature image.

3. Calculate the Skew angle using the Eigen vector

$\phi = [ M(1) / M(2) ]$       Where,  $\phi$  = Skew angle,     $M$  = the image matrix

$M(1)$  = the first element in the first row of the image matrix

$M(2)$  = the first element in the second row of the image matrix

4. After skew angle is found, skew correction is performed by applying rotation transformation to every pixel in the signature image.

#### **vii) Cropping:**

When scanned, signature image contains the signature and some white coloured non-signature regions. Those superfluous non-image portions are removed by cropping the image to the bounding rectangle of the signature part. Cropping is an essential preprocessing step for all types of classification techniques.

### **3) Feature Extraction:**

Feature extraction is a process of deriving some characteristic parameters or functions from the patterns (signature images). The extracted characteristic parameters or functions are called 'features'. Function features are functions of time and these can only be derived from online signatures. Characteristic parameters are extracted from offline signatures. The features should effectively represent their parent patterns with a reduced amount of data. Feature extraction helps in decreasing the computation complexities in the subsequent stages of signature verification.

The success of any pattern recognition system significantly depends on feature extraction. Extracted features must minimize the dissimilarity between same class patterns and must maximize the dissimilarity between two patterns from different classes. An ideal feature extraction method in offline signature verification system, should extract a minimum number of features that maximize the distance between the signature examples of other persons (interpersonal distance) but should minimize intrapersonal distance for those belonging to the same person[2].

Extracting features from an offline signature is challenging as compared to an online signature. Because in offline signature, information of dynamic features like pressure, acceleration, stroke order in the signature is lost[50].

#### **Types of features in offline signatures**

Features extracted from an offline signature are basically classified into two categories [20], [51], [52].

(i) Local Features and (ii) Global Features

**(i) Local Features:** Local features are extracted from a small part or a small region of the signature. The critical, distinct parts carrying distinguishing features are selected for this. Local features are very much noise sensitive. Extraction of local features is computationally expensive.



**(ii) Global Features:** Global features are extracted considering the complete signature image as a whole. Global features are easy to extract and these features are least sensitive to noise. But global features are affected by position alignment and they are highly susceptible to signature variations[1].

Two additional types of signature features are also found in literatures. They are (i) Geometrical features and (ii) Statistical features in [1], [53].

**(i) Geometrical Features:** Geometric features are derived from the geometrical parameters of the signature such as the height, width, aspect ratio, signature area etc. These features depict the characteristic geometry and topology of a signature. Both global as well as local properties are preserved in geometric features. This type of features has the ability to withstand distortion and rotation variation [54].

**(ii) Statistical Features:** In many approaches of offline signature verification, researchers have used statistical features of the signature. They are derived from distribution of pixels in the signature image. Some statistical features extracted from offline signatures are mean, centre of gravity of the signature image, global maxima, local maxima, moments etc. Statistical features can tolerate slight variations in signature style and distortion.

### **Features in offline signatures**

In literatures various features are found to be extracted from offline signatures. It has been observed that global features produced better recognition results in offline signature verification. We had tried to extract a greater number of global features with already proposed promising global features and some new global features proposed by us.

We extracted the following features from the signature samples present in our datasets:

**(i) Normalized Signature area (with respect to bounding box):** It is the total number of signature pixels or foreground pixels in the signature image. Signature area gives information about the signature density. If the signature image is skeletonized, signature area represents a measure of the density of the signature traces.

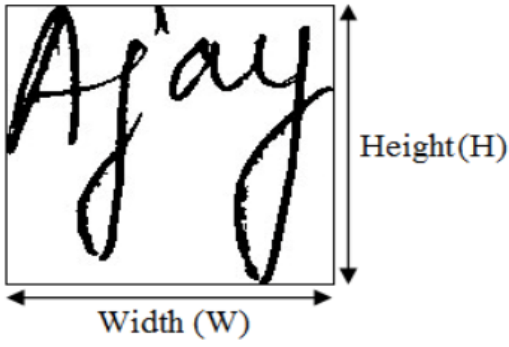
If in a signature image, total number of signature pixels (or black pixels if foreground image is black) = B, and total number of pixels in the whole image = P, then

$$\text{Normalized Signature Area} = B / P$$

Normalized signature area is also called Signature Occupancy Ratio

Signature Occupancy Ratio =

**(ii) Aspect Ratio (Signature width to height ratio):** This is the ratio of signature width to signature height of a cropped signature. It is seen that the aspect ratio of the signatures of a person fairly remains constant.



If signature height is H and signature width is W, then Aspect Ratio is given by

$$\text{Aspect Ratio} = \frac{W}{H} \quad (2.9)$$

**Horizontal and vertical center of the signature:** These two measurements indicate about the Horizontal and Vertical location of the signature image. The horizontal center ( $C_x$ ) is given by

$$C_x = \frac{\sum_{y=1}^{x_{\max}} x \sum_{x=1}^{y_{\max}} b[x, y]}{\sum_{x=1}^{x_{\max}} \sum_{y=1}^{y_{\max}} b[x, y]}$$

The vertical center ( $C_y$ ) is given by

$$C_y = \frac{\sum_{y=1}^{y_{\max}} y \sum_{x=1}^{x_{\max}} b[x, y]}{\sum_{x=1}^{x_{\max}} \sum_{y=1}^{y_{\max}} b[x, y]}$$

$B[x, y]$  indicates signature pixel (black pixel)

**(iii) Horizontal and Vertical Projections (Horizontal and Vertical Histograms):** Horizontal projection or histogram is found by counting the number of signature image pixels in each row in a signature image and plotting it horizontally with a line. The row with maximum value gives the maximum horizontal projection. Similarly, counting and plotting signature pixels in vertical direction for every column, maximum vertical projection or histogram is found.

Horizontal and Vertical Projections can be calculated as follows:

$$P_h[y] = \sum_{x=1}^n \text{black pixel}(x, y)$$

$$P_v[x] = \sum_{y=1}^m \text{black pixel}(x, y)$$

Where, m = width of the signature image

n = height of the signature image

pixels that constitute the stroke of the signature image are called the ON pixels. In a binary image, ON pixels are the black pixels.

**(viii) Centre of Gravity or Centroid:** In a binary signature image with black signature pixels, Centre of Gravity (CG) or Centroid is the average coordinate point of all black pixels.

The CG of a signature image is calculated by the following equations:

$$X = \frac{1}{N} \left( \sum_{i=1}^n x_i \right)$$

$$Y = \frac{1}{N} \left( \sum_{i=1}^n y_i \right)$$

Where,  $X_i$  is the column number of ON pixels and  $y_i$  is the row number of ON pixels. The pixels that constitute the stroke of the signature image are called the ON pixels. In a binary image, ON pixels are the black pixels.

**Slope of line obtained from curve fitting of centre of gravity of each column:**

Center of gravity of each column of the signature image is found. Then the linear polynomial curve was obtained using least square curve fitting technique. Slope of this line is the signature feature [58].

The general expression for curve fitting using a least square curve fitting method is:

$$err = (d_i)^2 = (y_1 - f(x_1))^2 + (y_2 - f(x_2))^2 + (y_3 - f(x_3))^2 + (y_4 - f(x_4))^2$$

Here,  $err$  = The error function

$(x_i, y_i)$  = Data point (CG value),  $(x_i, f(x_i))$  = Point on the fitted line

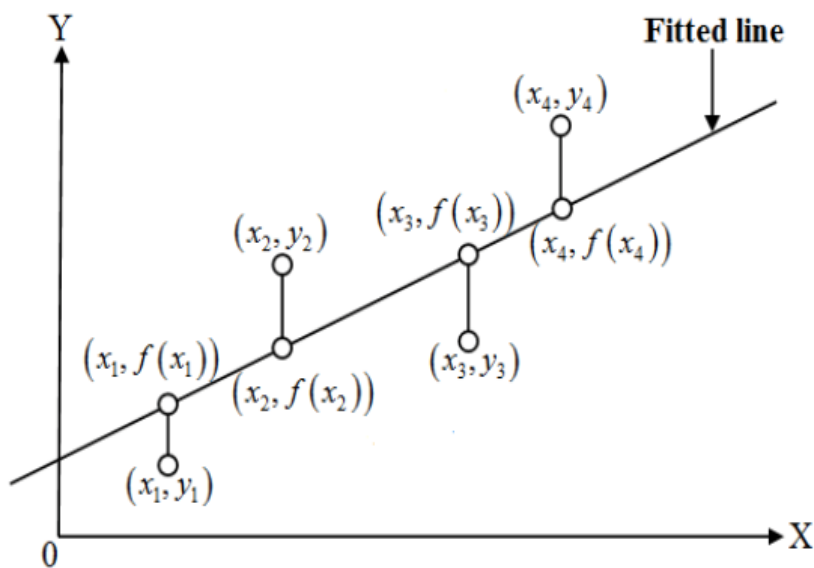


Fig. 2.9. Curve fitting using least square curve fitting method

**(iv) Number of cross points/End Points:** In a skeletonized signature image, Cross point is a signature pixel or point that has more than two 8-neighbours. (When pixels are considered on a grid, a pixel is surrounded by 8-neighbouring pixels; they are called its 8-neighbours).

**(v) Center of Gravities of the vertically divided images:** The Signature image is vertically divided into two halves. Then for both the halves, the centre of gravities  $CG_1$  and  $CG_2$  are found. It is the pair of centroids of the 2 images formed by vertically dividing the main image.

**Skew Angle:** If  $CG_1$  and  $CG_2$  are the centre of gravities of the two halves of the vertically divided images, then the angle between the horizontal axis and the line joining  $CG_1$  and  $CG_2$  is called the 'Skew angle'.

**(vi) Slope of Center of Gravity of two equal halves of signature image:** If CGs of the first and second halves of the signature image are  $(x_1, y_1)$  and  $(x_2, y_2)$ , then slope of the line joining the two CGs is  $m = \frac{(y_2 - y_1)}{(x_2 - x_1)}$ .

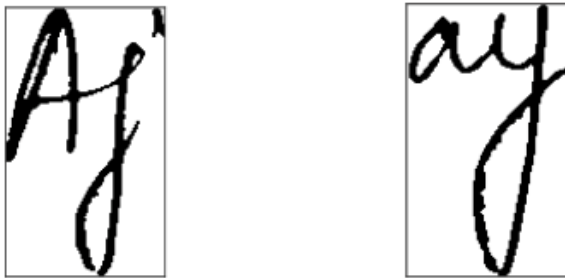


Fig. 2.10. Signature image vertically divided into two parts

**Baseline shift or Orientation of signature:** To calculate this parameter, two centroids (say  $CG_1$  and  $CG_2$ ) of the vertically equally divided signature image are found. If

$CG_1: (x_1, y_1)$  and  $CG_2: (x_2, y_2)$ , then  
Baseline shift or Orientation of signature =  $y_2 - y_1$

### Code for Pre-processing and Feature Extraction:

```
function [Feat_Val,im_processed] = featureExtraction(I)

%FEATUREEXTRACTION Summary of this function goes here

% Detailed explanation goes here

%I=imread('image71.jpeg '); % Load the image file and store it as the variable I.

i=0

figure(),imshow(I);

% pause

I2=imresize(I,[512 ,512]);
```

```

figure(),imshow(I2);

% pause

I3=rgb2gray(I2);

%I3=medfilt2(I3]);

I3=im2double(I3);

I3=im2bw(I3); %converting image to black and white

I3 = bwmorph(~I3, 'thin', inf); %thining the image

I3=~I3;

im1 = I3;

figure(),imshow(I3);

% pause

%extracting the black pixels

k=1;

for i=1:512
for j=1:512
if(I3(i,j)==0)

u(k)=i;

v(k)=j;

k=k+1;

I3(i,j)=1;

end

end

end

C=[u;v];

N=k-1;%the number of pixels in the signature

oub=sum(C(1,:))/N; %the original x co-ordinate center of mass of the image

ovb=sum(C(2,:))/N; %the original y co-ordinate center of mass of the image

%%%%%%%%%%%%%*****ROTATE*****%%%%%%%%%%%%%

%moving the signature to the origin

for i=1:N

u(i)=u(i)-oub+1;

v(i)=v(i)-ovb+1;

end

% the new curve of the signature

C=[u;v];

ub=sum(C(1,:))/N;

vb=sum(C(2,:))/N;

```

```

ubSq=sum((C(1,:)-ub).^2)/N;

vbSq=sum((C(2,:)-vb).^2)/N;

for i=1:N

uv(i)=u(i)*v(i);

end

uvb=sum(uv)/N;

M=[ubSq uvb;uvb vbSq];

%calculating minimum igen value of the matrix

minIgen=min(abs(eig(M)));

%the eigen vector

MI=[ubSq-minIgen uvb;uvb vbSq-minIgen];

theta=(atan((-MI(1))/MI(2))*180)/pi;

thetaRad=(theta*pi)/180;

rotMat=[cos(thetaRad) -sin(thetaRad);sin(thetaRad) cos(thetaRad)];

%% rotating the signature and passing the new co-ordinates

for i=1:N

v(i)=(C(2,i)*cos(thetaRad))-C(1,i)*sin(thetaRad));

u(i)=(C(2,i)*sin(thetaRad))+C(1,i)*cos(thetaRad));

end

C=[u;v];

%moving the signature to its original position

for i=1:N

u(i)=round(u(i)+oub-1);

v(i)=round(v(i)+ovb-1);

end

%after rotating the image the signature might go out of the boundry (128x128) therefore

%we have to move the signature curve

mx=0;%the moving x co-ordinate

my=0;%the moving y co-ordinate

if (min(u)<0)

mx=-min(u);

for i=1:N

u(i)=u(i)+mx+1;

end

end

if (min(v)<0)

my=-min(v);

```

```

for i=1:N

v(i)=v(i)+my+1;

end

end

C=[u;v];

for i=1:N

I3((u(i)),(v(i)))=0;

end

figure(),imshow(I3);

% pause

% removing extra white space in sides

xstart=512;

xend=1;

ystart=512;

yend=1;

for r=1:512

for c=1:512

if((I3(r,c)==0))

if (r<ystart)

ystart=r;

end

if((r>yend))

yend=r;

end

if (c<xstart)

xstart=c;

end

if (c>xend)

xend=c;

end

end

end

%cutting the image and copying it to another matrix

for i=ystart:yend

for j=xstart:xend

im((i-ystart+1),(j-xstart+1))=I3(i,j);

```

```

end

end

figure(),imshow(im); %cropped image

im_processed = im;

% Feature Extraction - NSA -----

PixelB = 0;

PixelA = 0;

for i=ystart:yend

for j=xstart:xend

if (im(i-ystart+1,j-xstart+1)== 0)

PixelB = PixelB + 1;

end

PixelA = PixelA + 1;

end

end

disp('Normalised Signature Area')

%disp([PixelB,PixelA]);

NSA = PixelB/PixelA;

disp(NSA);

% Feature Extraction - Aspect Ratio -----

height_sign = yend-ystart;

length_sign = xend-xstart;

aspect_ratio = length_sign/height_sign;

disp('Aspect Ratio')

disp(aspect_ratio)

% Feature Extraction - Maximum Horizontal and Vertical Projection

max=0;

for i=ystart:yend

summ=0;

for j=xstart:xend

if(im((i-ystart+1),(j-xstart+1))==0)

summ=summ+1;

end

end

if (summ>max)

max=summ;

end

```



```

end

max;

max1=0;

for i=xstart:xend

    summ=0;

    for j=ystart:yend

        if(im((j-ystart+1),(i-xstart+1))==0)

            summ=summ+1;

        end

    end

    if (summ>max1)

        max1=summ;

    end

end

max1;

xdiff=xend-xstart;

ydiff=yend-ystart;

%disp(xdiff)

%disp(ydiff)

Hor_Proj = max/xdiff;

Ver_Proj = max1/ydiff;

disp('Maximum Horizontal')

disp(Hor_Proj)

% Feature Extraction End Points -----

i1 = im1;

[row, col, depth] = size(i1);

%add row%

addrow = ones(1, col);

i1 = [addrow; addrow; i1; addrow];

[row, col, depth] = size(i1);

%add column%

addcol = ones(row, 1);

i1 = horzcat(addcol, i1, addcol, addcol);

[row, col, depth] = size(i1);

i1=~i1;

crosspoints=0;

for r = 3:row-1

```

```

for c = 2:col-2

if(i1(r,c)==1)

if (i1(r-1,c-1)+i1(r-1,c)+i1(r-1,c+1)+i1(r,c-1)+i1(r,c+1)+i1(r+1,c-1)+i1(r+1,c)+i1(r+1,c+1)==1)

crosspoints=crosspoints+1;

%disp(i1(r,c))

end

end

end

end

disp('Feature Extraction End Points')

disp(crosspoints)

% Feature Extraction Center of Gravities of the vertically divided images--

n1 = im(:, 1: xdiff/2); %splitting images

n2 = im(:, xdiff/2+1:xdiff);

%figure(6),imshow(n1);

%figure(7),imshow(n2);

sum1=0;

pix_total=0;

%for the first half

for i=1:ydiff

pix_sum=0;

for j=1:xdiff/2

if(n1(i,j)==0)

pix_sum=pix_sum+1;

pix_total=pix_total+1;

end

end

sum1=sum1+(pix_sum*i);

end

Y1=sum1/pix_total;

RY1=Y1/ydiff;

sum1=0;

for i=1:xdiff/2

pix_sum=0;

for j=1:ydiff

if(n1(j,i)==0)

pix_sum=pix_sum+1;

```

```

end

end

sum1=sum1+(pix_sum*i);

end

X1=sum1/pix_total;

RX1=2*X1/xdiff;

%for the secong half

sum1=0;

pix_total=0;

for i=1:ydiff

    pix_sum=0;

    for j=1:xdiff/2

        if(n2(i,j)==0)

            pix_sum=pix_sum+1;

            pix_total=pix_total+1;

        end

    end

    sum1=sum1+(pix_sum*i);

end

Y2=sum1/pix_total;

RY2=Y2/ydiff;

sum1=0;

for i=1:xdiff/2

    pix_sum=0;

    for j=1:ydiff

        if(n2(j,i)==0)

            pix_sum=pix_sum+1;

        end

    end

    sum1=sum1+(pix_sum*i);

end

X2=sum1/pix_total;

RX2=2*X2/xdiff;

centroid = [ [RX1 RY1] [RX2 RY2] ];

disp('Feature Extraction Center of Gravities of the vertically divided images')

disp(centroid)

% Feature Extraction - Slope -----

```

```

m=xdiff;

m=m/2;

k=m+X2;

slope=(Y2-Y1)/(k-X1);

disp('Slope')





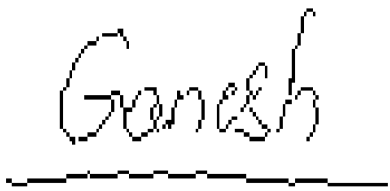
disp(slope)

Feat_Val = [ NSA aspect_ratio Hor_Proj crosspoints centroid slope];




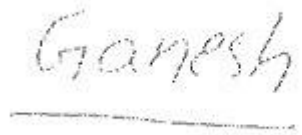
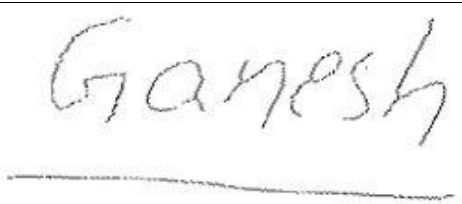
end

```

### **Pre-processed Signatures(signature-1):**

	<b>Original image</b>
	<b>Resized image</b>
	<b>Binarized image or binary image, thinned image</b>
	<b>Skew corrected image</b>
	<b>Cropped image</b>

**Pre-processed Signatures(signature-2):**

	<b>Original image</b>
	<b>Resized image</b>
	<b>Binarized image or binary image, thinned image</b>
	<b>Skew corrected image</b>
	<b>Cropped image</b>

### Feature Extraction(signature-1):

featureExtraction.m	GUI.m	Figure 1	Figure 2	Figure 3	Figure 4	Figure 5
Command Window						
Normalised Signature Area						
0.0205						
Aspect Ratio						
2.8902						
Maximum Horizontal						
0.1540						
Feature Extraction End Points						
27						
Feature Extraction Center of Gravities of the vertically divided images						
0.6036    0.6423    0.4025    0.6332						
Slope						
-0.0078						

### Feature Extraction(signature-2):

featureExtraction.m	GUI.m	Figure 1	Figure 2	Figure 3	Figure 4	Figure 5
Command Window						
Normalised Signature Area						
0.0256						
Aspect Ratio						
1.7826						
Maximum Horizontal						
0.1423						
Feature Extraction End Points						
21						
Feature Extraction Center of Gravities of the vertically divided images						
0.5213    0.5167    0.5243    0.5091						
Slope						
-0.0086						

### **4) Verification using Extracted Features:**

- Once the Features are extracted, they are stored in a Vector. Now we need to fix minimum error threshold values for each feature implemented using which it classifies genuine signatures.
- Conventionally finding the error threshold values should be done by using an Machine Learning Algorithm that will minimize the error threshold.
- But since we have not been taught Machine Learning, we have done this step manually by generating the tables of data for input and selecting error threshold values. The more images this step is applied to the more accurate our algorithm becomes.

```
Feat_Val = [ NSA aspect_ratio Hor_Proj crosspoints centroid slope];
```

#### Command Window

```
Feat_Err =  
    0.0060    0.8000    0.0400    6.0000    0.0750    0.0750    0.0750    0.0750    0.0065  
  
Feat1 =  
    0.0205    2.8902    0.1540   27.0000    0.6036    0.6423    0.4025    0.6332   -0.0078  
  
Feat2 =  
    0.0256    1.7826    0.1423   21.0000    0.5213    0.5167    0.5243    0.5091   -0.0086  
  
Dif =  
    0.0051    1.1076    0.0117    6.0000    0.0823    0.1255    0.1218    0.1242    0.0007  
  
>> |
```

If any value Dif[i] is greater than array Feat\_Err[i] then we can say that signature is invalid.

### **RESULTS:**

We have written a program in MATLAB that identified and distinguished 50 signatures out of 60 signatures, So the accuracy of this project is nearly 80-85%.

### **SCOPE FOR IMPROVEMENT:**

The program we have written is very primitive compared to what should be implemented for live security systems and can be improved by:

- Making a machine learning neural network and training it to generate more accurate error threshold values that can improve the accuracy of the program.
- By increasing the database, we can have more data to train our algorithm which will also increase the accuracy of our program.

### **REFERENCES:**

- i) V. K. Madasu and B. C. Lovell, "An Automatic Off-Line Signature Verification and Forgery Detection System" In: Verma, B., Blumenstein, M. (eds.) Pattern Recognition Technologies and Applications: Recent Advances, 1st ed. IGI Global, Hershey, 2007, pp. 63-89.
- ii) L. Batista, D. Rivard, R. Sabourin, E. Granger and P. Maupin, "State of the Art in Off-Line Signature Verification" In: Verma B., Blumenstein M. (eds.), Pattern Recognition Technologies and Applications: Recent Advances, (1e). IGI Global, Hershey, 2007, pp. 39-62.
- iii) J. J. Brault and R. Plamondon, "How to Detect Problematic Signers for Automatic Signature Verification," In the Proceedings of International Carnahan Conference on Security Technology, Zurich, Switzerland, 3-5 Oct. 1989, pp. 127-132.