Master's Thesis

FINGERPRINT ENHANCEMENT USING FUZZY LOGIC AND DEEP NEURAL NETWORKS

Sridevi Sarraju

Department of Computer Science and Engineering

Graduate School of UNIST

2019

FINGERPRINT ENHANCEMENT USING FUZZY LOGIC AND DEEP NEURAL NETWORKS

Sridevi Sarraju

Department of Computer Science and Engineering

Graduate School of UNIST

FINGERPRINT ENHANCEMENT USING FUZZY LOGIC AND DEEP NEURAL NETWORKS

A thesis/dissertation
submitted to the Graduate School of UNIST
in partial fulfillment of the
requirements for the degree of
Master of Science

Sridevi Sarraju

09/10/2019 of submission
Approved by

_____

Advisor
Dr. Franklin Bien

FINGERPRINT ENHANCEMENT USING FUZZY LOGIC AND DEEP NEURAL NETWORKS

Gil-Dong Hong

This certifies that the thesis/dissertation of Sridevi Sarraju is approved.

09/10/2019 of submission

signature

_____

Advisor: Dr. Franklin Bien

signature

_____

Dr. Jin Ho Chung: Thesis Committee Member #1

signature

_____

Dr. Jaesik Choi: Thesis Committee Member #2

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Fingerprint recognition analysis is one of the most leading preferred prodigious biometric advancement which has drawn generous consideration in biometrics. In this work, fingerprint Intensification is performed which is defined by Fuzzy logic technique and recognize the matching image with its unique characteristics extracted and classify the features extracted from a fuzzy enhanced image along with three major types of Neural Networks which are Feed Forward Artificial Neural Network, Neural Network, Recurrent Neural Network in order to classify the unique features extracted from a fingerprint image. This work efficiently expresses the results with Fuzzy logic enhancement and Neural Networks classifiers. Its principle goal is to improve the image using Fuzzy and extricate the spurious minutiae detected and classify the different features generated using GLCM and DWT. This work displays a framework of unique finger impression classification based on particular characteristics for extricating different features and three types of Neural Network for classification. Fuzzy technique is used for the fuzzy based image enhancement to urge the clear see of the unique finger impression.

Fingerprint Image Intensification is the procedure to enhance the distorted images to encourage the recognizable proof. The motivation behind the work is to enrich the quality of the distorted condition image generated from any fingerprint sensor, as Images can be corrupted due to various conditions and one of the principle issues is the resolution of the fingerprint sensor generating noisy images. High-quality pictures are vital for exact coordinating of unique finger impression pictures. But unique mark pictures are seldom of idealize refinement. As it may be corrupted or debased due to varieties of the skin, impression state and condition. In this way, unique finger impression images must be improved before utilized. The idea behind this work fingerprint image intensification process is to improve the quality of a distorted and noisy fingerprint images generated from a low-cost fingerprint sensor. Execution of current fingerprint acknowledgment frameworks is vigorously influenced by the precision of their characteristic's extraction evaluation. These days, there are more ways to deal with fingerprint analysis with worthy outcomes. Issues begin to emerge in low-quality conditions where the dominant part of the conventional strategies dependent on examining the surface of fingerprint can't handle this issue so effectively as Neural Networks.

Fuzzy logic technique is implemented first to remediate the distorted picture and enhance it with the implementation of GLCM and DWT2 algorithm features of an image is extracted, post to which three types of Neural Network Classification is performed to analyze the accuracy of the image generated

from the extracted feature parameters and match the test and trained result with the implementation of Neural Networks and classify the outcome results. The three Neural Network used are Artificial Neural Network (ANN), Neural Network (NN), Recurrent Neural Network (RNN). This algorithm works efficiently to identify the fingerprint matching from the predefined trained images from the fuzzy enhanced image generated.

Experiments are performed (in MATLAB 2019 student version) to make sure the extraction process should not get the false minutiae and preserve the true extracted features Fuzzy based Image Enhancement method makes sure the feature traits of the image is intensified. Better improvement proves the quality improvement further incrementing the highest accuracy determined in the classification further.

This work can be used in wide area of applications in biometrics as it is a combined work of distorted fingerprints enhancement, false feature removal, true feature extraction, matching of the images for identification purpose and classification using Neural Networks. Experiments show results which are quite promising and gives a direction of the subsequent further analysis in future work.

CHAPTER 1. INTRODUCTION

There are different types of sensors available in Biometrics. Fingerprint's unique image acknowledgment is one of the first and unique biometric advancements that have been assembled freely under digital forensics legal sciences. A matching image will generate improved reconnaissance and conceivable further activity. For the framework to be compelling, the coordinating database ought to be as wide and thorough as would be prudent. There are numerous ways to deal with a biometric-based client fingerprints verification system and all it requires some type of equipment input gadget to accumulate the necessary data about the client to be confirmed. For instance, unique finger impression acknowledgment requires a finger impression scanner. These equipment gadgets must be an indispensable piece of the CR stage over a protected channel. Huge numbers of these gadgets are highly used and very frequently used in Biometrics. The channel interfacing the equipment hardware and programming must be equipped for supporting the information move prerequisites without an undue exhibition sway on the gadget's centre usefulness.

1.1 Fingerprint Sensing

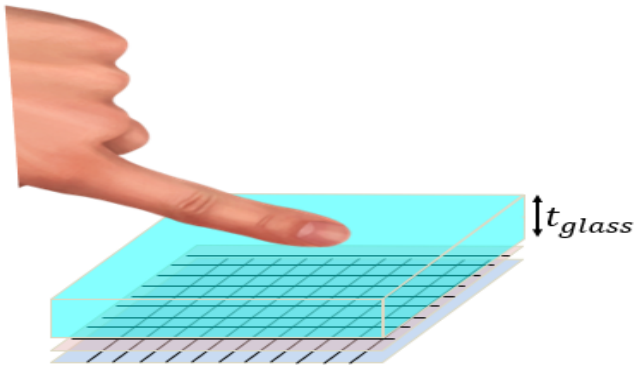
The thickness of the tempered glass is one of the main reasons behind the accuracy of the fingerprint images generated. There could be various other factors as well leading to distorted image generation such as internal circuitry of the sensors or climatic conditions causing wet or dry images, wounds on surface of the skin, dust, moistness, and uneven contact with the scanner. One of the fundamental causes of the complication is in implementing a touch screen, as the glass thickness in thick tempered glass used for internal protection can cause reduction in fingerprint detection accuracy. The integrated touch screens are one of the main factors leading un- improvised image. Hence, the more the thickness of the glass, the less the accuracy of the images generated.

Figure 1 Biometric fingerprint Scanners examples

1.1.1 Fingerprint Scanners

There are variety of fingerprint identification scanners available in the market let it be a smart phone which is very common, smart watch, home security system or any other biometric security systems. The filtering method starts when the finger is subjected on the sensors glass screen. A gadget known as charge coupled device is responsible in taking the photocopy of the image. In order to edify the features of the fingerprint structure truly makes a changed image of the fingerprint. Now the identification is done by the darker part addressing dynamically reflected light and lighter part addressing the less reflection of light.

Prior to standing out the print from set away data, the scanner processor guarantees the CCD has gotten a sensible picture. It examines the ordinary pixel haziness, or the widespread characteristics in a little model, and dismiss the scope if the imprecise picture if unreasonably dull or too much light, the scanner alters the acquaintance time with let in practically light and thereafter endeavors the yield again.



Glass thickness	Sensing Result
t_{glass} → thin	
t_{glass} → thick	

Figure 2 Sensing glass of the Fingerprint sensors

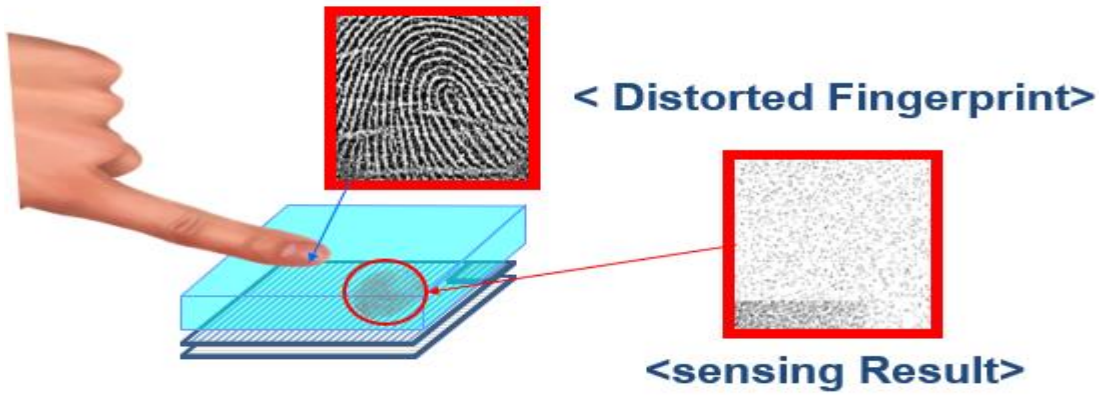


Figure 3 Thickness of the sensing glass

1.2 Scope of The Thesis

Therefore, sustainable algorithm is experimented and implemented to check the accuracy of the image determines which is improvised first using Fuzzy Logic and with the implementation of morphological operations and feature extraction algorithms like GLCM and DWT 2 dimensional algorithms. The challenge is to obtain and wrest the true elements of the image and eliminate the false features extracted in regulation to improvise the standard of the image. Neural Network assumes an significant function to classify the different accuracies obtained after which matching of the features is accomplished.

CHAPTER 2. FINGERPRINT FEATURES

Fingerprints are the most unique creations having different characteristics. Even two identical twins have different fingerprint features. Fingerprints are scaled and identified by some minute form of information which is specified as minutiae. The primarily essential known minutiae features are defined as ridges and bifurcations. These minute features on the fingertips make it less difficult for hatching the hands to get a handle any object.

2.1 Fingerprints

The limit of fingerprints is a hard and fast event. Similar to the remaining organs in the humans, these edges structure through a blend of genetic and common segments. The innate code in DNA gives general demands in the skin traits should be configured in making the embryo, yet the form it constructs is an aftereffect of unpredictable events. The accurate situation of the premature infant in the womb at a moment and the unmistakable association and thickness of enveloping amniotic fluid picks how every individual edge will outline. Along these lines despite the unlimited things that go into picking our inherited make-up regardless, there are limitless biological factors affecting the advancement of the finger’s attributes. Quite equivalent to the atmosphere conditions that structure fogs or any equivalent whether, the entire improvement process is turbulent to such an extent that, in the entire course of humanity's history, there is essentially no chance to get of the equal cautious model confining double time.

Thus, fingerprints are one of a kind marks for an individual, Indeed, even a vague twin. Furthermore, remembering that two prints may give off an impression of being indistinguishable at first, a readied operator or a moved piece of programming can pick clear, described complexities.
This is the central idea of one of a kind imprint assessment, in both crime assessment and surveillance. A one of a kind imprint scanner's obligation is to supplant a human analyst by social affair a print test and standing out it from various models on record.

2.2 Minutiae

Ridges and Bifurcations are the two main principle components characteristics of the minutiae. Ridge is characterized to be a solitary bent fragment though a valley is the region between

two neighboring edges. So, the dark or black regions are considered ridges and white zone that exists between them is known as valleys. Bifurcation alludes to the point where a ridge partition to shape two edges.

Figure 4 Example Image of ridges and bifurcations

2.2.1 Classification: Minutiae Features

Significant highlights of a fingerprint and are utilized in the coordinating of fingerprints. These details focus is utilized to decide the uniqueness of a unique finger impression picture. A decent quality unique fingerprint picture can have 40 to 90 minutes relying upon the unique finger impression scanner goals and the situation of the finger on the sensor. Minutiae characteristics can be characterized as the junctions where the edge lines end or fork. Therefore, the minutiae features are defined as the nearby edge discontinuities and can be of numerous kinds as described below;

1. **Ridge Ending** is described as the point where an edge closes unexpectedly.
2. **Bifurcation** is characterized as the point where an edge forks or wanders into branch edges.
3. **Double Bifurcation** is the dot when one grating edge isolates into two abrading ridges.
4. **Dot** is sporadically, an edge unit may remain secluded that resembles a spot between ordinary edges.
5. **Island** is line-type which remains solitary. (i.e., doesn't contact a different line-type and is completely contained in the example territory of intrigue.

6. **Bridge** is an interfacing grating bridge between parallel running edges, for the most part, right points.
7. **Lake** is an enclosure solitary grating edge that bifurcates and re-joins after short course and proceeds as a solitary grating edge.
8. **Hook Spur** is a separated line with one short edge fanning out a more extended edge.
9. **Trifurcation** is the junction when one rubbing edge partitions into three erosion edges.

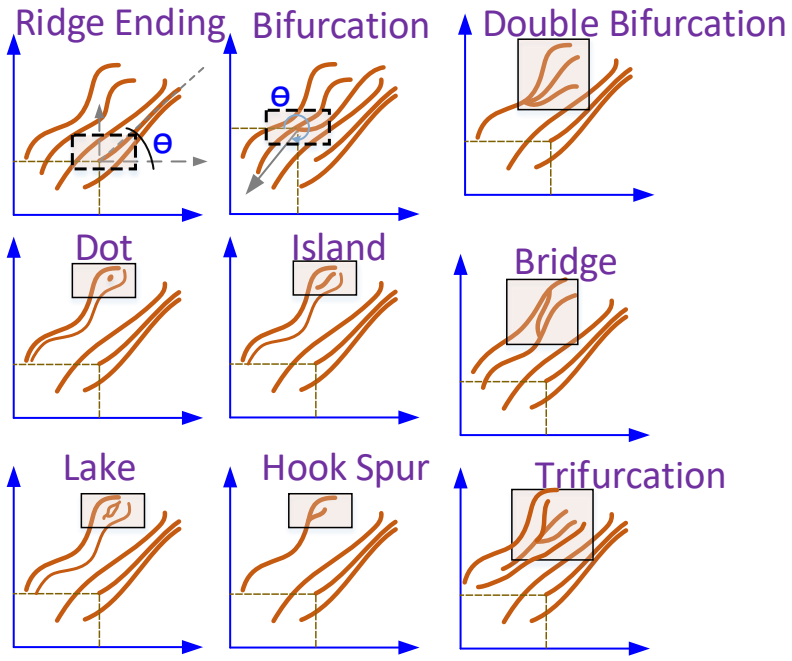


Figure 5 Various Minutiae Features

CHAPTER 3. FINGERPRINT ENHANCEMENT

3.1 Traditional Method: Gabor Filter and Gaussian

Gabor Filter method is one the most famous and traditional technique in fingerprint enhancement. Fingerprint analysis is into effect with the end goal of foremost security. With the purpose of edge detection, Gabor is a filter named after a famous scientist Dennis Gabor. The evaluation of the filter is based on the frequency and other parameters like orientation which is somewhat matching to the human visual framework as they have been especially proper for surface portrayal and separation. Be that as it may, some unique mark pictures trapped in variety applications are poor in quality, which undermines the exactness of fingerprint affirmation.

Hence Fingerprint enhancement is normally the initial step. Also, there have been many other researches based on the same, out of which Gabor Filter along with Gaussian filter technique is been the most common and popular one technique which integrates with the Gaussian process which provides the weights and directional orientation enhances the image. In image processing, Gabor filter it is used particularly because of its ideal characteristics in computer vision.

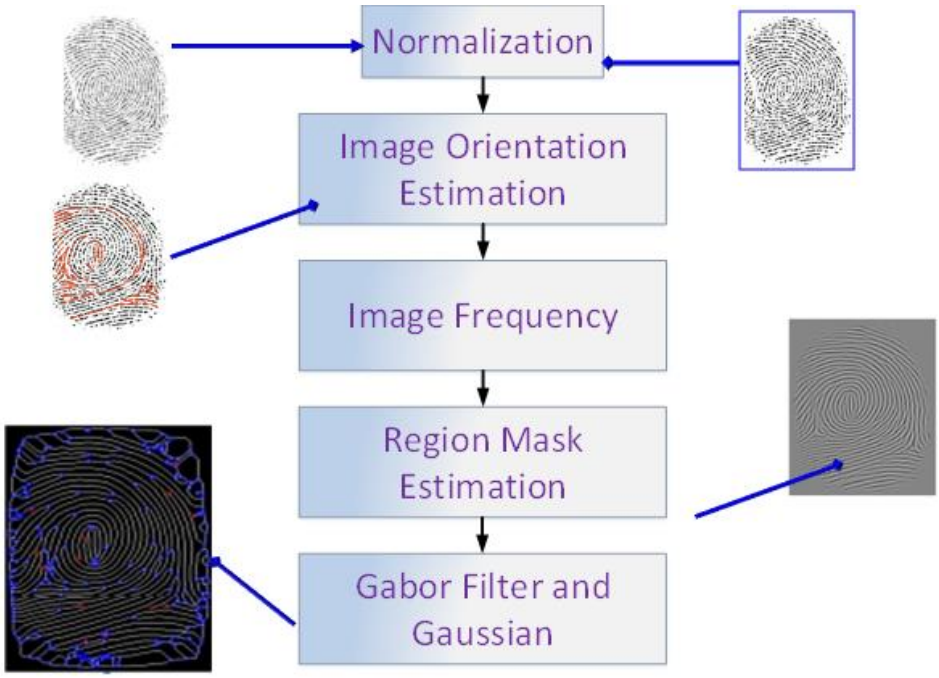


Figure 6 Steps for Gabor and Gaussian Filter method

Figure 6 describes the steps of Gabor and Gaussian filter flow chart, as the initial experiments were carried on the traditional method in Python.

3.1.1 Method: Gabor Filter

It begins with **Normalization** which initializes the image and preprocess it but does not make any changes to the features of the fingerprint like ridges and bifurcations. It is pixel wise operation and the image is normalized first so that it can have a well-defined evaluation of its mean and variance. Ans also it is performed so the dim level exists in the given arrangement of value parameters and standardize the intensity value by adjusting the range of the gray level values.

Figure 7 Normalization Example

Orientation Estimation evaluates the orientation value of the normalized preprocessed image. This direction field shows visibly delineates the fundamental shape, structure and directional element of a unique mark. It comprises of few stages which includes;

1. Preprocessing the normalized image
2. Deciding the essential edge of unique finger impression
3. Evaluation of directional estimates
4. Adjusting the assessed direction field.

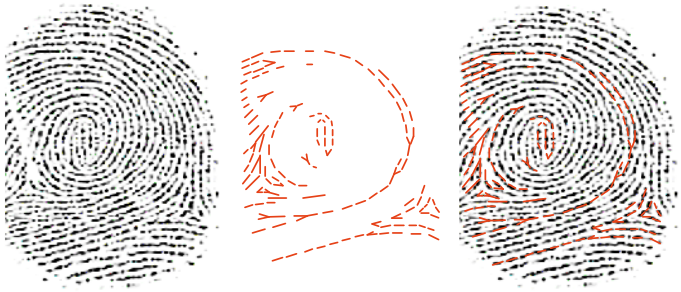


Figure 8 Orientation Estimation Example

Image Frequency calculates the frequency of the image which is calculated from the output of the orientation estimation. It estimates the distance between each point. It is demonstrated that for recurrence the changeability of recurrence esteems in various pictures with different pixel wise resolution, the normal recurrence of zones of unique mark ought to be assessed.

Region Mask Estimation helps in segregating several regions of overlapped fingerprint further utilized to calculate the underlying positioning area. As the image operates on pixel wise evaluation applied to recoverable and non-recoverable regions.
Regions are estimated depending on the appearance obtained by the valleys and ridges

Figure 9 Region Mask Estimation Example

Gaussian Filter is a precise filter used to suppress the boisterous image while keeping the edges of the image remain sharp. The main advantage is that it is quicker with its multiplication and addition functionalities.

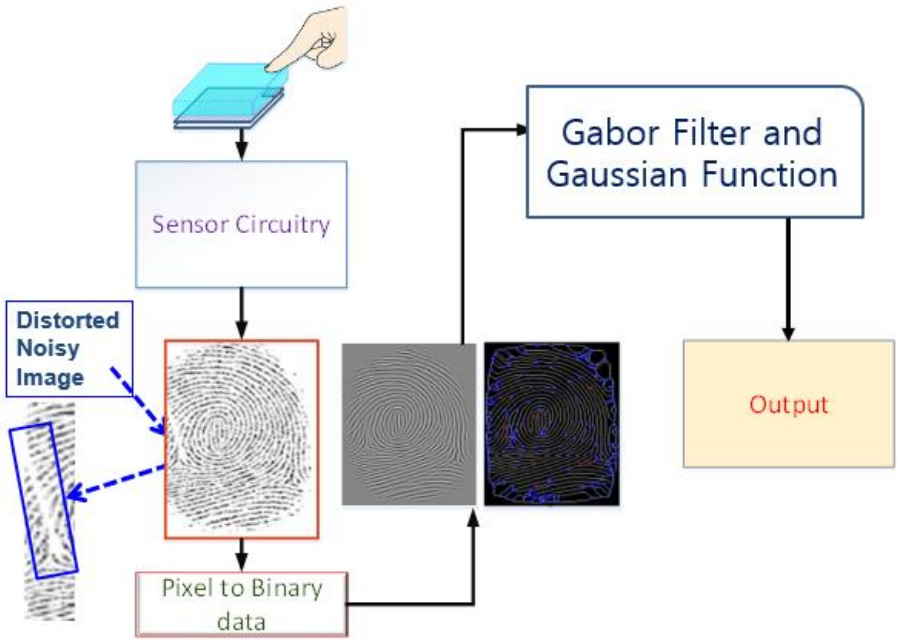


Figure 10 Traditional Method: Gabor Filter and Gaussian with enhanced output and minutiae extracted image

The Fingerprint picture is enhanced first performing the pixel wise operation in the backend and converting it to binary image and from the enhanced image minutiae features are extracted with the implementation of the Gabor and the Gaussian filters.

3.1.2 Binary data conversion

In the Fingerprint Enhancement process the image data is converted to the binary form with respect to 0 and 1, one is considered as the dark or black part and the light or white part is considered as 1. The pixel wise operation is evaluated, and data is converted to binarized value depending upon the threshold value of the image. There are several ways an image is converted to binary. This process starts with an input image preprocessed first the normalized, the orientation of the image is evaluated, upon completion frequency is estimated depending on the pixel value operation then comes the ridge estimation part after performing all these operations image is processed to the next part which is Gabor filtering which again integrated with Gaussian filter which converts the image to binary and shows the enhanced image results and upon further filtering the features in the form of minutiae of the image is extracted. The example of the binary data conversion and its visualization in the binary data format is demonstrated in the figure below.

Distorted image Binary Raw data form

Figure 11 Distorted image data converted to binary data

distorted enhanced minutae

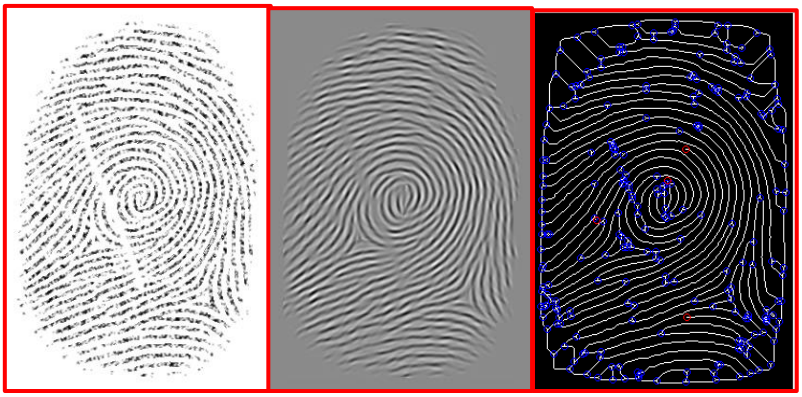
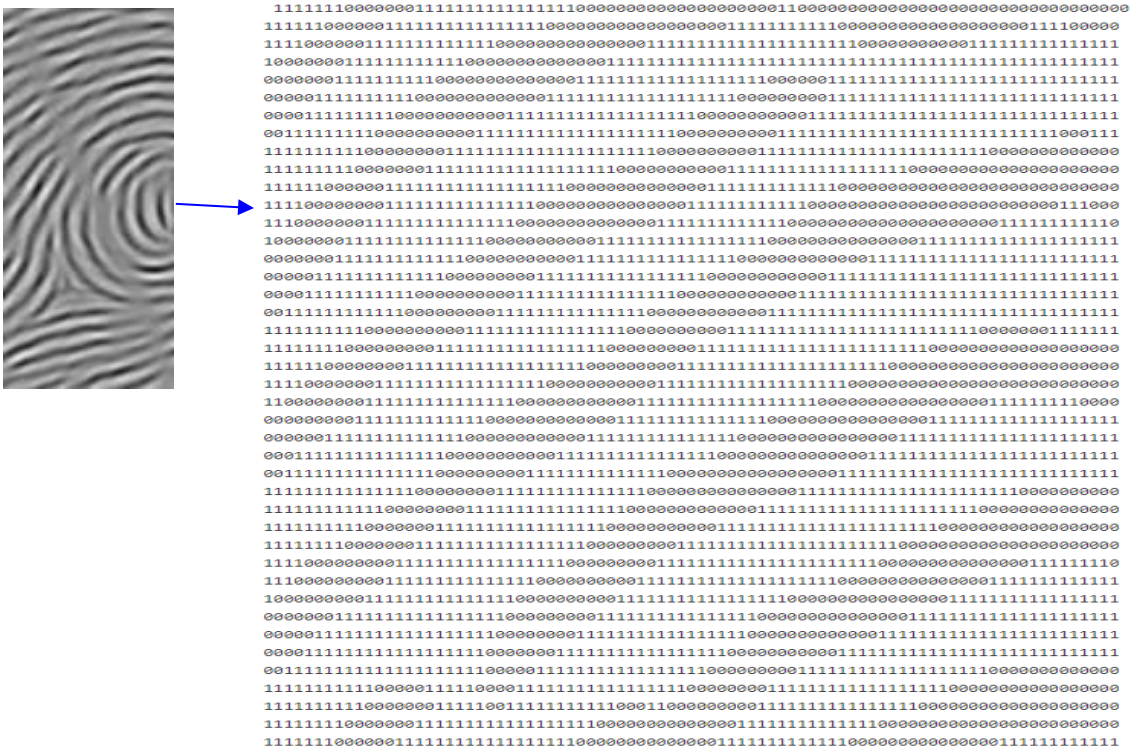


Figure 12 Conversion of the image staring from distorted form to enhanced to the minutae



Enhanced Image Binary Raw Data

Figure 13 Enhanced image data converted to binary data

Minutiae extracted Image Binary Raw Data

Figure 14 Minutiae extracted image data converted to binary data

3.1.3 Result of Gabor and Gaussian Method

Database of fifty images have been tested using the Gabor and Gaussian Method. Several examples of the image and its results is shown below. The varieties of images chosen are with multiple distortions, horizontal distortions, double slant angular distortions and slant angular distortions.

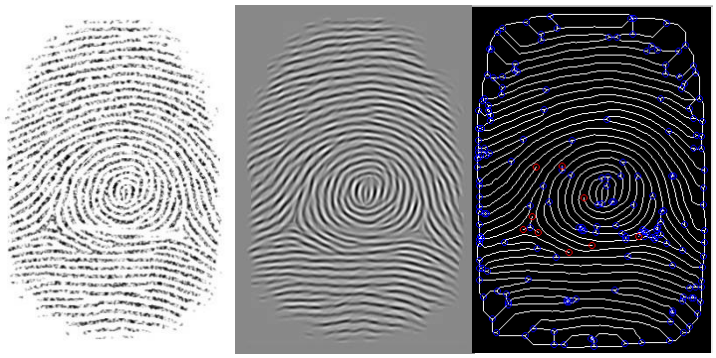


Figure 15 Horizontal distortion

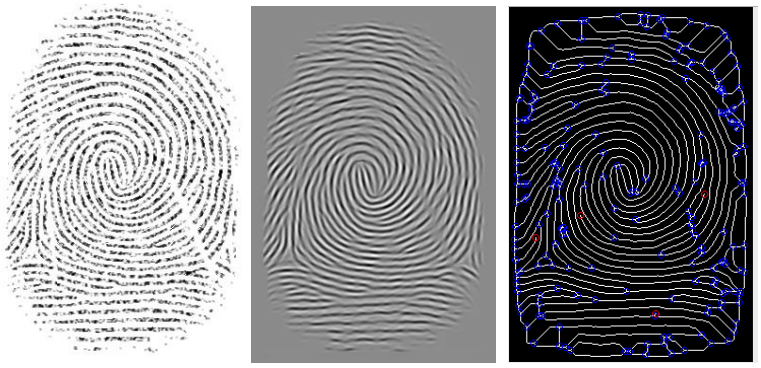


Figure 16 Angular distortion

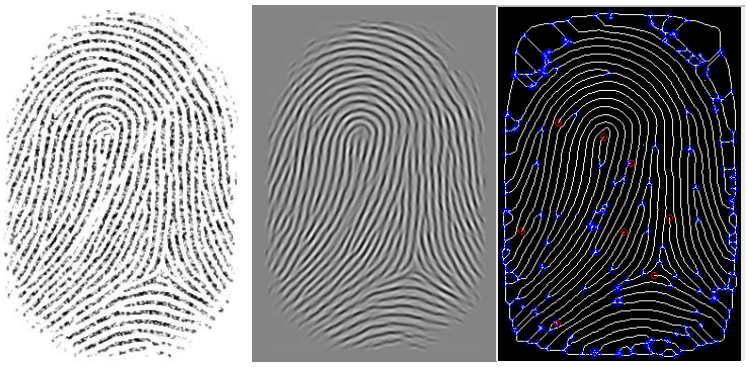


Figure 17 Slant angular distortion

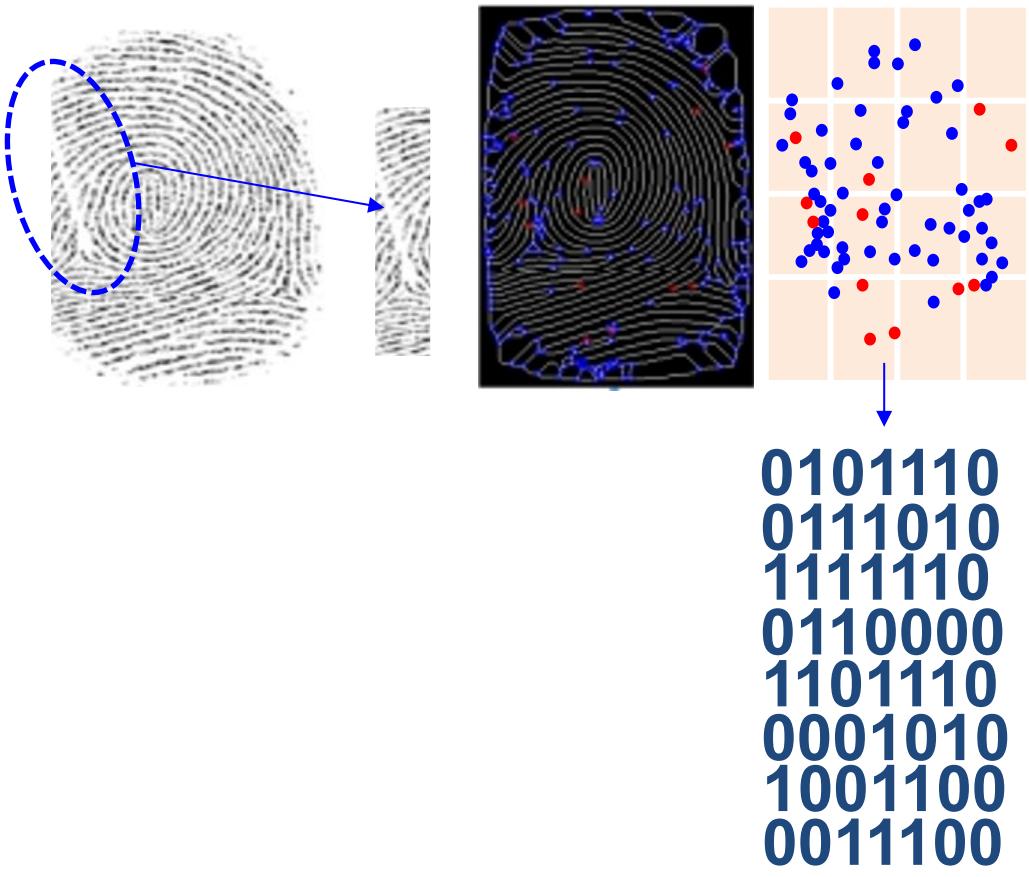


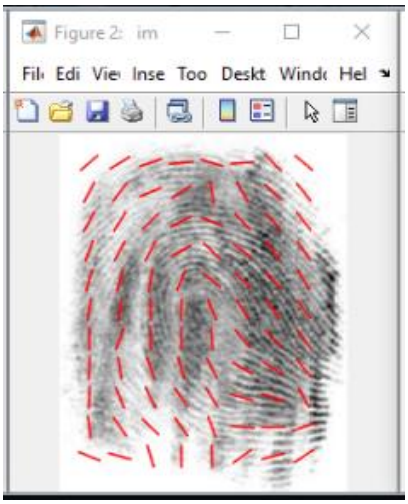
Figure 18 Minutiae to Binary conversion example

3.1.4 Implementation in MATLAB

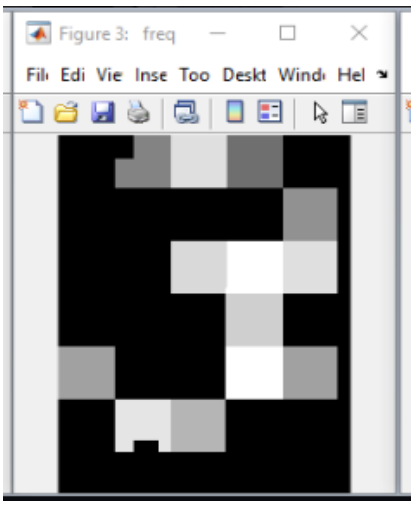
The Gabor and Gaussian method are experimented in MATLAB environment as well and the both the results are compared. It is noticed that when the results are compared from python to MATLAB gave considerable amount of difference in the results including thinning of the image after performing the region mask estimation. MATLAB results are shown more prominent with good visuals of the enhanced binary image. The image examples are shown below.

Original Image

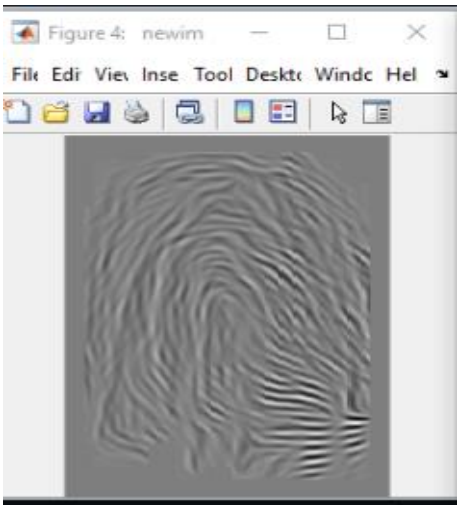
Orientation



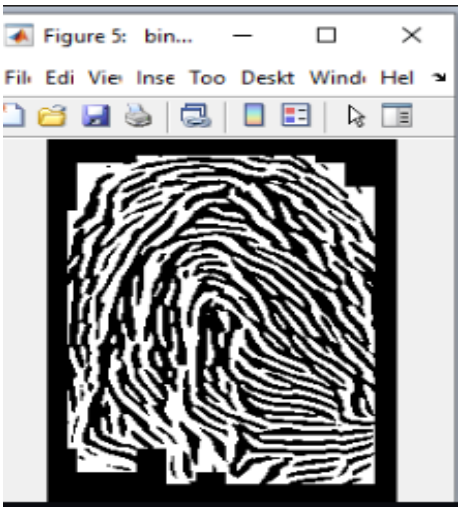
Frequency



Enhancement



Region Estimation



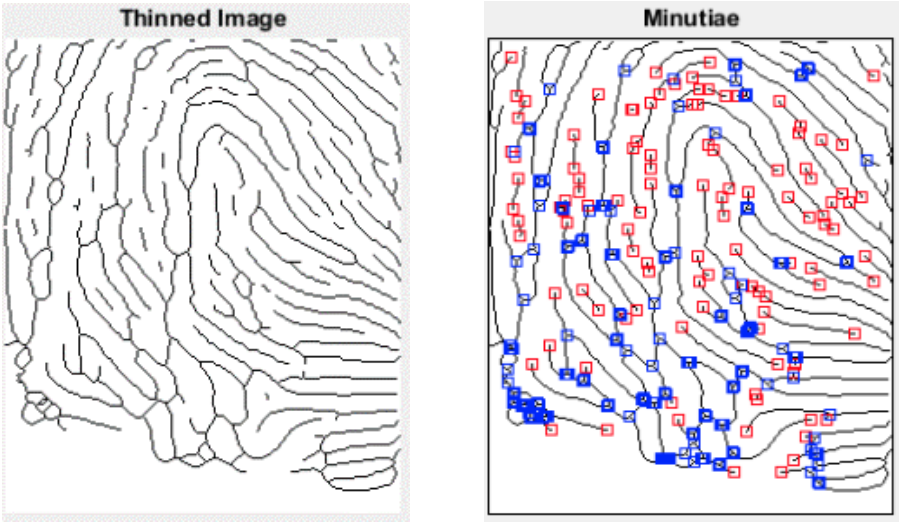


Figure 19 Image modification process

3.2 Technical Challenge

Distortions of the fingerprint image in low resolution, cost effective sensors often lead to the challenge in training the computer interface to reconstruct the noisy pieces of the information which leads to the following problems as stated below.

3.2.1 Problem Statement

1. Spurious minutiae will be generated.
2. True minutiae can be lost in the process.
3. Errors in positioning of the extracted features.
4. The ridge and bifurcations are not always well defined.
5. Classification of the extracted features is quite challenging.

Also there are certain drawbacks related to the traditional Gabor filter method as it is unknown fact that what could be the accuracy of the enhancement and what is the percentage of the evaluation, classification needs to be performed using different classifiers to understand the evaluation of the analysis performed on the enhancement and noise reduction which should also extract the true value and should be able to eliminate the false values.

3.2.2 Objective

So as to improve the nature of the uproarious picture generated from a fingerprint cost effective sensor the objective of the thesis is stated as below.

1. To Enhance the quality of the image.
2. Removal of the false minutiae generated and preserve the true extracted features.
3. Classification of the extracted features to check the accuracy using Neural Networks.
4. Efficient performance evaluation of the proposed method
5. The overall computational time taken should be less.

CHAPTER 4. PROPOSED METHOD

In order to overcome the drawbacks from the traditional Gabor filter method and evaluate the percentage of accuracy associated with the enhancement evaluation there must be some technique implemented to find the answers of improvement. Classification is an important stage to classify the test results to compare between the different classifiers and find out the best possible solution amongst all the classifiers.

The overall outcome of the results should eliminate the noisy part of information and preserve the accurate features and in order to do that we need improved feature extraction techniques to have large part of information which can be evaluated and analyzed.

Matching should be performed between the test and trained data in order to have the fingerprint identification. Out of all the initial step should be the enhancement part and the implementation are based on Fuzzy Logic with GLCM and DWT 2 dimensional as feature extracting algorithms is implemented. Deep Neural Networks has been turned out to be used for the classification purposes and the outcome is estimated measured across different Neural Networks post to which matching is performed between the test and results obtained.

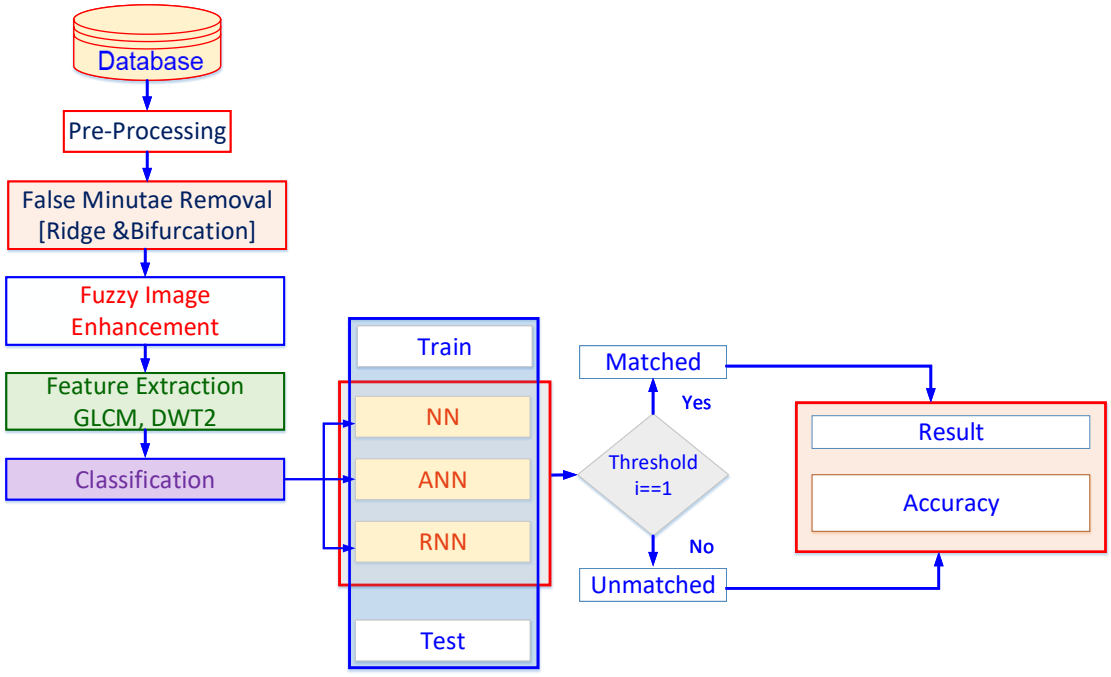


Figure 20 Architecture: Proposed Method

4.1 Pre-processing

The Preprocessing begins with ROI Binary image conversion from the input image upon which morphological operation take place which converts the image to thinned image. Then minutiae and feature extraction are performed, and ridges and bifurcations are determined, and then false detection of these features are detected and removed. Once again true ridge and bifurcation findings are calculated and preserve the true features and false minutiae filtering is applied to the results of feature extraction. After the successful completion of all the preprocessing steps the image is applied to the Fuzzy Logic for the enhancement.

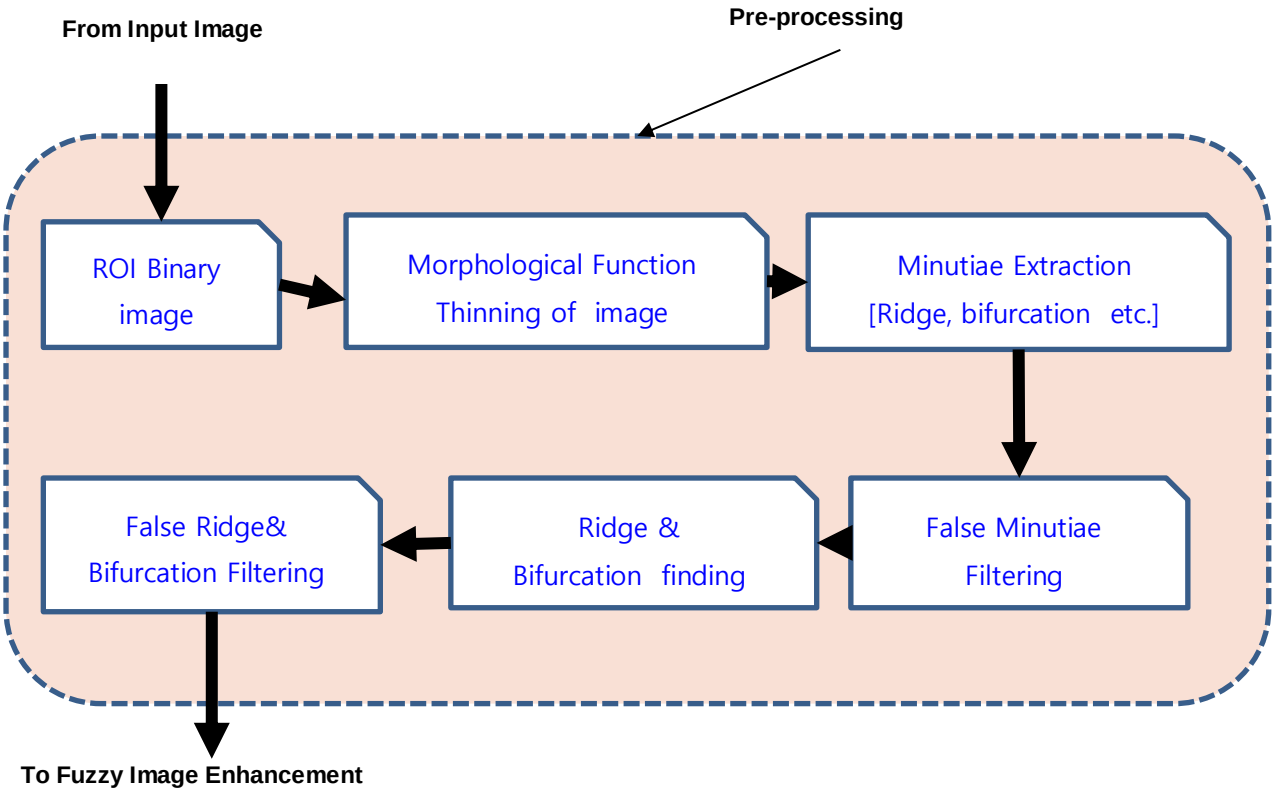


Figure 21 Pre-processing

4.1.1 ROI Binary Image

The typical utilization of an ROI is to make a double cover picture. In the veiled picture, pixel values that have a place with the ROI are set to "one" and pixels outside the ROI are set to "zero". ROI is basically termed as a grayscale image according to plot the region of interest image on the investigating mode. The ROI shape will be recognized in this picture. Used to characterize a locale of enthusiasm for the picture.

A district of intrigue (ROI) is a bit of a picture that we need to channel or work on somehow or another. The tool stash underpins a lot of ROI parts that we can use to make ROIs of numerous shapes, such circles, ovals, polygons, square shapes, and hand-drawn shapes. After creation, we can utilize ROI object properties to modify their appearance and working. Moreover, the ROI items bolster article capacities and occasions that can be used to actualize intuitive conduct. For instance, utilizing occasions, on application can execute custom code at whatever point the ROI changes position. As a comfort, the tool stash incorporates a parallel arrangement of accommodation capacities for ROI creation.

After the fingerprint image is pre-processes a unique reference point is determined to secure a region of interest and for each block there will be standard deviations mean and entropy.

Implementation of the code as below:

```
%% ROI
% binary_image = binary_image(120:400,20:250);
figure(2);imshow(binary_image);title('Input image');
```

4.1.2 Morphological Operations

Morphological operations is a broad positioning of picture managing skills that strategize based on images that are reliant on each of its appearance figures. In a morphological operation, each pixel in the image is adjusted reliant on the estimation of various pixels in its neighborhood. By selecting the area and condition of the region, a morphological action is built that is deliciated to precise orientation in the information image these activities consolidate a picture with an organizing component, frequently. They process a picture pixel wise operation as per the local pixel esteems. Morphological tasks are regularly applied to parallel pictures, in spite of the fact that strategies are accessible for dark level pictures.

Here with the help of the morphological operations thinning of the image is performed as from the thick binary image it gets difficult to evaluate the true features. Thinning is performed in order to decrease

the ridge width which will remain only 1mm wide after performing the thinning of the image. As a sample the initial image is set to 1 and pixelwise logical operation is performed in 3*3 matrix. This is often available for gray level images.

Hence modification is done in a way such that on every image pixelwise logical operation is performed. Minimal lowest significance of the thinned image is easy to compute the effectiveness rather than the original image. Thinning on dilated image is performed to get a dilated image.
Also, often the ridge in a binary image will be found having lot of holes in case if we count the bifurcation values, we will find extra incorrect bifurcation values, hence in order to fill the holes before we can find the minutiae points.

While performing the gray scale of the image is turned into binary image with the morphological functions. The pre-processing steps enhances the quality and it becomes easy to process next.

Code Implementation:

```
%% Morphological function
thin_image=~bwmorph(binary_image,'thin',Inf);
figure(3);imshow(thin_image);title('Thinned Image');
```

4.1.3 Minutiae Extraction

This operation is performed to obtain the ridge and bifurcations etc, minutiae features which are the most important features to identify the different characteristics of any person's fingerprint that's how the difference in the characteristics helps to identify the uniqueness of a person's fingerprint image. The far-reaching arrangement of unique finger impression acknowledgment frameworks in different applications has caused worries that traded off finger impression formats might be utilized to make phony fingers, which could then be utilized to delude all unique mark frameworks a similar individual is joined up with. Once traded off, the grayscale picture is the most in danger. Spillage of a stage picture or skeleton picture is likewise perilous since it is a trifling issue to reproduce a grayscale unique finger impression picture from the stage picture or the skeleton picture. Rather than the over three portrayals, spillage of particulars layouts has been viewed as less genuine as it isn't minor to remake a grayscale picture from the details.

Hence minutiae features are extracted first post to which the fake minutiae is detected and removed

with the help of the current algorithm.

Code Implementation:

```
%% Minutiae extraction
s=size(thin_image);
N=3;%window size
n=(N-1)/2;
r=s(1)+2*n;
c=s(2)+2*n;
double temp(r,c);
temp=zeros(r,c);bifurcation=zeros(r,c);ridge=zeros(r,c);
temp((n+1):(end-n),(n+1):(end-n))=thin_image(:,:);
outImg=zeros(r,c,3);%For Display
outImg(:,:,1) = temp .* 255;
outImg(:,:,2) = temp .* 255;
outImg(:,:,3) = temp .* 255;
for x=(n+1+10):(s(1)+n-10)
    for y=(n+1+10):(s(2)+n-10)
        e=1;
        for k=x-n:x+n
            f=1;
            for l=y-n:y+n
                mat(e,f)=temp(k,l);
                f=f+1;
            end
            e=e+1;
        end
         if(mat(2,2)==0)
            ridge(x,y)=sum(sum(~mat));
            bifurcation(x,y)=sum(sum(~mat));
         end
    end
```

4.1.4 False Ridge and Bifurcation filtering

The initial step after the minutiae extraction if removal of the fake ridge and bifurcation post to which the remaining operations are operated so as to remove the fake part and preserve the true details of the image.

Code Implementation:

```
% RIDGE END FINDING -removal of Minutiae
[ridge_x ridge_y]=find(ridge==2);
len=length(ridge_x);

for i=1:len
    outImg((ridge_x(i)-3):(ridge_x(i)+3),(ridge_y(i)-3),2:3)=0;
    outImg((ridge_x(i)-3):(ridge_x(i)+3),(ridge_y(i)+3),2:3)=0;
    outImg((ridge_x(i)-3),(ridge_y(i)-3):(ridge_y(i)+3),2:3)=0;
    outImg((ridge_x(i)+3),(ridge_y(i)-3):(ridge_y(i)+3),2:3)=0;

    outImg((ridge_x(i)-3):(ridge_x(i)+3),(ridge_y(i)-3),1)=255;
    outImg((ridge_x(i)-3):(ridge_x(i)+3),(ridge_y(i)+3),1)=255;
    outImg((ridge_x(i)-3),(ridge_y(i)-3):(ridge_y(i)+3),1)=255;
    outImg((ridge_x(i)+3),(ridge_y(i)-3):(ridge_y(i)+3),1)=255;
end
```

4.1.5 Ridge and Bifurcation finding

Filtering has been performed multiple times in the algorithm in order to have the true features retreated. So after the feature findings once again the filter is applied to preserve the true ridges and bifurcations.

This process also helps to improve the accuracy to higher values. The image is about prepared for the procedure of highlight extraction. Yet there are little examples in the unique mark that should be evacuated to expand the exactness. This is finished by computing the quantity of pixels in each edge and any edge that has few pixels will be expelled from the unique finger impression.

Code Implementation:

```
%BIFURCATION FINDING
[bifurcation_x bifurcation_y]=find(bifurcation==4);
len=length(bifurcation_x);
%For Display
for i=1:len
    outImg((bifurcation_x(i)-3):(bifurcation_x(i)+3),(bifurcation_y(i)-
3),1:2)=0;
    outImg((bifurcation_x(i)-
3):(bifurcation_x(i)+3),(bifurcation_y(i)+3),1:2)=0;
    outImg((bifurcation_x(i)-3),(bifurcation_y(i)-
3):(bifurcation_y(i)+3),1:2)=0;
    outImg((bifurcation_x(i)+3),(bifurcation_y(i)-
3):(bifurcation_y(i)+3),1:2)=0;

    outImg((bifurcation_x(i)-3):(bifurcation_x(i)+3),(bifurcation_y(i)-
3),3)=255;
    outImg((bifurcation_x(i)-
3):(bifurcation_x(i)+3),(bifurcation_y(i)+3),3)=255;
    outImg((bifurcation_x(i)-3),(bifurcation_y(i)-
3):(bifurcation_y(i)+3),3)=255;
    outImg((bifurcation_x(i)+3),(bifurcation_y(i)-
3):(bifurcation_y(i)+3),3)=255;
end
figure(4);imshow(outImg);title('Minutiae');
```

4.1.6 False Minutiae Filtering

After the completion of double filtering using the morphological operations at the end the false minutiae filtering is performed. The final aim emphasizes on the double filtration technique and obtaining the real features, consequently, it is important to apply evaluations for expelling all these false details. Separating misleading particulars from authentic details in the post-handling stage is significant for exact unique finger impression acknowledgment. The more the fake particulars are wiped out, the better the coordinating exhibition will be. This is taken care by estimating the Euclidean distance across the two centroids of the different minutiae points to remove the majority spurious minutiae. The implementation of the code is given below.

Code Implementation:

```
%% removal
LTermLab=bwlabel(thin_image);
propTerm=regionprops(LTermLab,'Centroid');
CentroidTerm=round(cat(1,propTerm(:).Centroid));
figure(5);
imshow(~thin_image)
set(gcf,'position',[1 1 600 600]);
hold on
plot(CentroidTerm(:,1),CentroidTerm(:,2),'ro')
LBif=(thin_image==3);
LBifLab=bwlabel(LBif);
propBif=regionprops(LBifLab,'Centroid','Image');
CentroidBif=round(cat(1,propBif(:).Centroid))
% imshow(~K)
D=6;
Distance=DistEuclidian(CentroidBif,CentroidTerm);
SpuriousMinutae=Distance<D;
[i,j]=find(SpuriousMinutae);
CentroidBif(i,:)=[];
CentroidTerm(j,:)=[];
Distance=DistEuclidian(CentroidBif);
SpuriousMinutae=Distance<D;
[i,j]=find(SpuriousMinutae);
CentroidBif(i,:)=[];

Distance=DistEuclidian(CentroidTerm);
SpuriousMinutae=Distance<D;
[i,j]=find(SpuriousMinutae);
CentroidTerm(i,:)=[];
hold off
figure(6)
imshow(~thin_image)
hold on
```

4.1.7 Results of Pre-process

Input Binarized Thinned

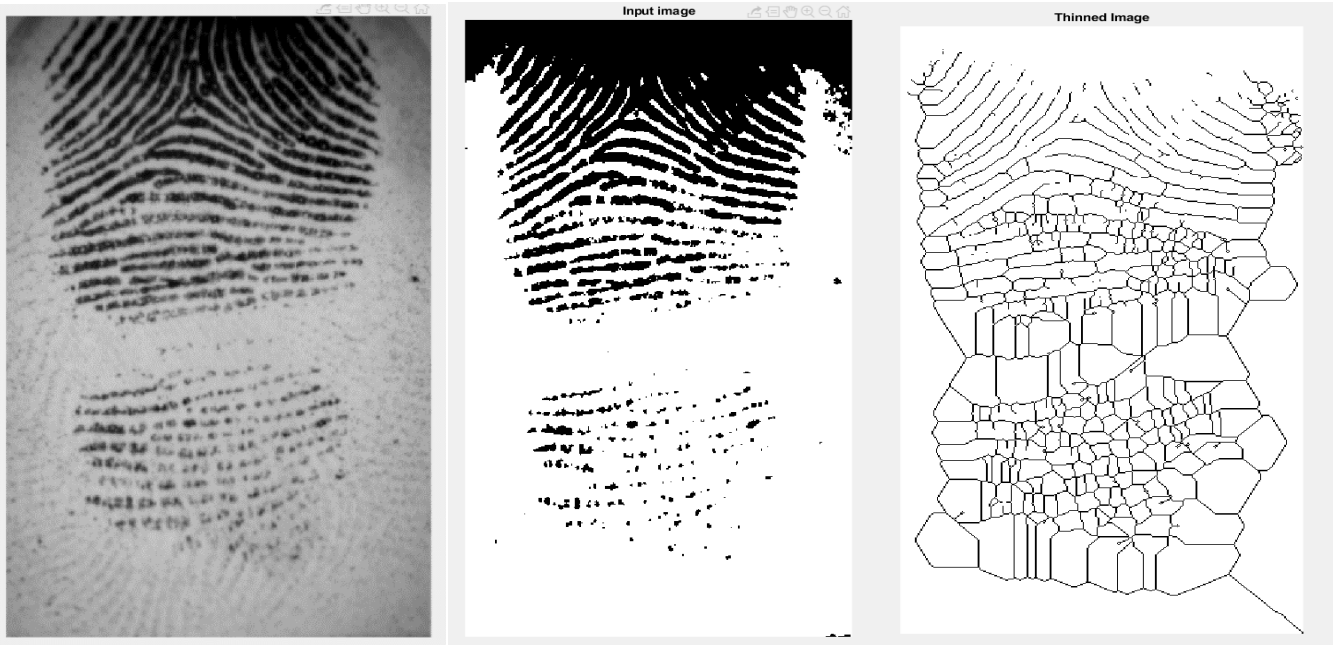


Figure 22 Preprocess

Minutae Feature Filtering

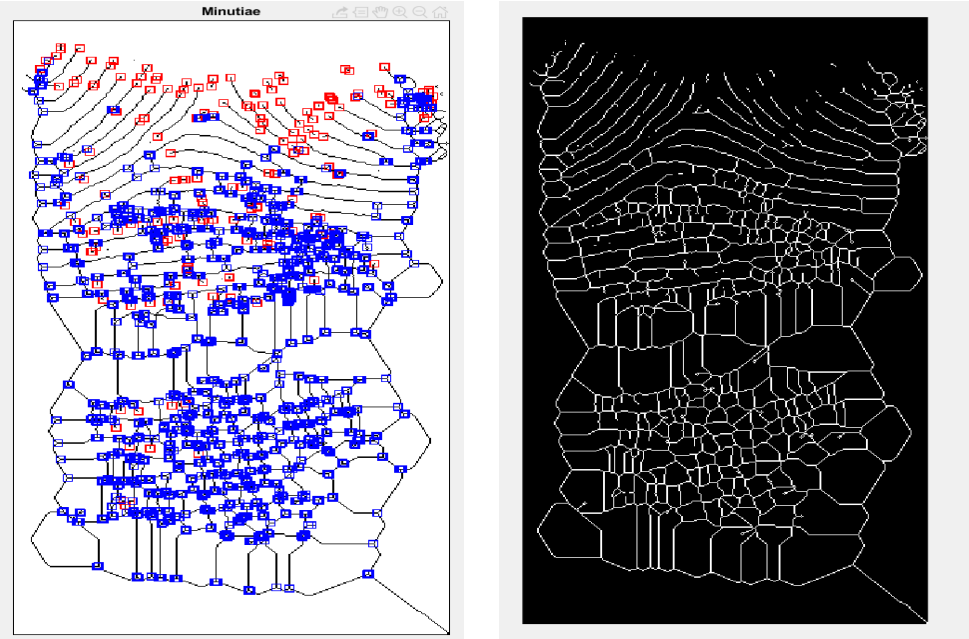


Figure 23 Preprocess after filtering

4.2 Fuzzy Logic

Fuzzy is a rationale or control arrangement of a n-esteemed rationale framework which uses the degrees of state "degrees of truth of the information sources and delivers yields which rely upon the conditions of the data sources and pace of progress of these states as opposed to the standard thing genuine or fake. It fundamentally gives establishments to inexact thinking utilizing loose and off base choices and permits utilizing semantic factors. It acknowledges the deliberate factors as info and changes over the numerical qualities to semantic factors. It changes the physical qualities just as the error sign to a standardized fluffy subset which comprises of an interim for the information esteems range and participation works that portray the likelihood of condition of the input factors.

Fuzzy activity includes utilization of fuzzy sets and participation capacities. Each set is a portrayal of a semantic variable that characterizes the conceivable condition of the output. Participation capacity is the capacity of a nonexclusive incentive in a fuzzy set, with the end goal that both the conventional worth and the fluffy set have a place with an all-inclusive set. With the application of Fuzzy logic technique, the image is preprocessed and enhanced first, and features of the minutiae are extracted with morphological operations in order to identify the matching true minutiae and make comparison between trained and test fingerprint image from the database and classification is performed using three types of the Neural Network.

4.2.1 Fuzzy Operation

In Pre-processing original image is taken of a particular size suitable for further pre-processing.

1. Pre-processed picture is exposed to the proposed strategy for fuzzy based upgrade.
2. Fuzzy upgrade the nature of the given image by wiping out clamor and improving the magnitude fragments.
3. It works on minute information locales known as tiles, as opposed to working on the whole picture.
4. Each tile differentiation is improved with the goal that the histogram of the yield area coordinates around to the predetermined histogram.
5. The adjacent tiles are then joined utilizing bilinear insertion so as to dispose of misleadingly initiated limits.

Algorithm

I. First the input image is specified to a binary range [0 255]

II. Discover the histogram of the image

III. Divide the outcome by number of pixels

IV. Calculate cumulative sum

V. Convert the image into the int value

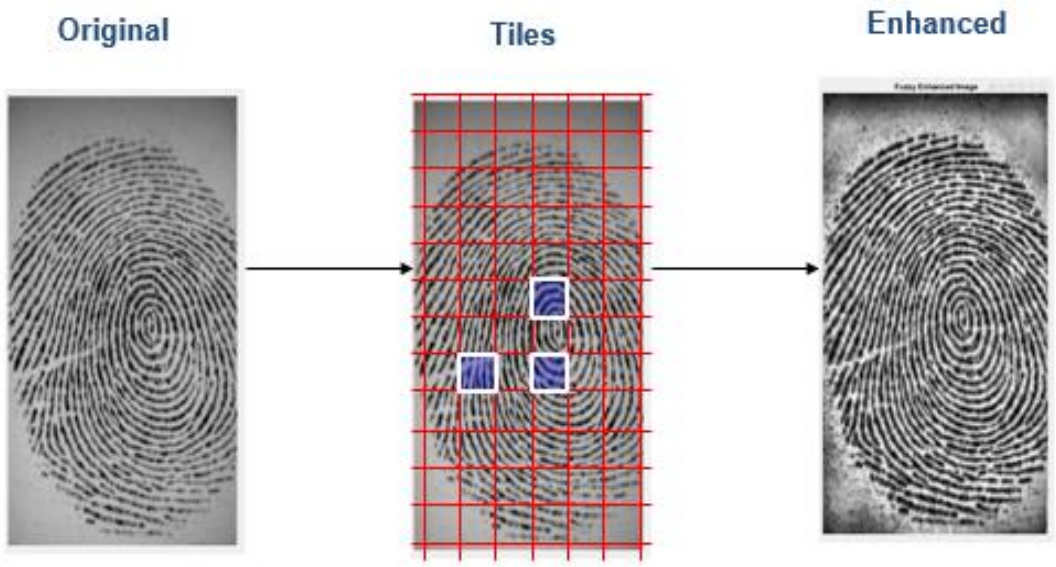


Figure 24 Fuzzy tile operation

Code Implementation:

```
%% Fuzzy based Image Enhancement
A=input_image;
tic
%Specify the bin range[0 255]
bin=255;
%Find the histogram of the image.
Val=reshape(A,[],1);
Val=double (Val);
I=hist(Val,0:bin);
%Divide the result by number of pixels
Output=I/numel(A);
%Calculate the Cumlative sum
CSum=cumsum(Output);
%Perform the transformation S=T(R) where S and R in the range [ 0 1]
HIm=CSum(A+1);
%Convert the image into uint8
HIm=uint8(HIm*bin);
I=input_image;
disp('Parameters for Fuzzy Enhanced Image')
tic
K1=I(:,:,1);
II=I;
J1=I(:,:,1);
K1 = Fuzzy_opt(K1);
J3 = Fuzzy_opt(K1);
toc;
figure(7);
imshow(J3)
title(' Fuzzy Enhanced Image')
```

4.2.2 Results of Fuzzy Enhanced images

Figure 25 Fuzzy Enhanced Result Image Example 1

In the Figure 25 Example 1 it is noticed that from the original input image the Fuzzy enhanced image has intensified the image and the features which are hidden, and unseen are clearly visible in the final output and the intensity of the image is also improved. Missing ridges and bifurcations are also visible and there is a noticeable difference seen. In all the images the intensity of the image is improved and brighter.

In the Figure 25 Example 2 lot of features which are partially seen are newly built and clearly seen in the fuzzy enhanced image.

In the Figure 25 Example 3 darker and unseen lines are formed in the enhanced fuzzy image which is visibly clearer as compared to the initial image.

In the Figure 25 Example 4 image intensification is clearly improved.

In the Figure 25 Example 5 the initial image is very lighter and most of the portions are unseen and not clear and in fuzzy improved image shows a better quality of the image.

As it's a known fact that fuzzy logic does not improvise the result up to 100 percent but shows the logical improvement, hence the noticeable and considerable amount of changes with the help of current algorithm is shown in this work.

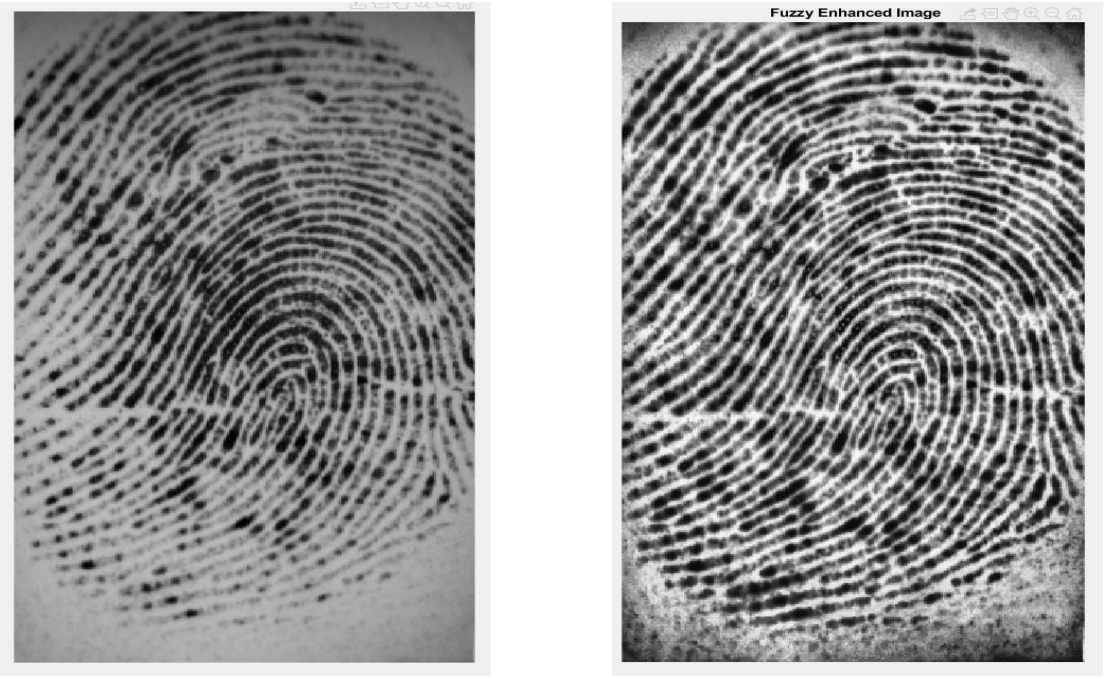


Figure 26 Fuzzy Enhanced Result Image Example 2

Figure 27 Fuzzy Enhanced Result Image Example 3

Figure 28 Fuzzy Enhanced Result Image Example 4

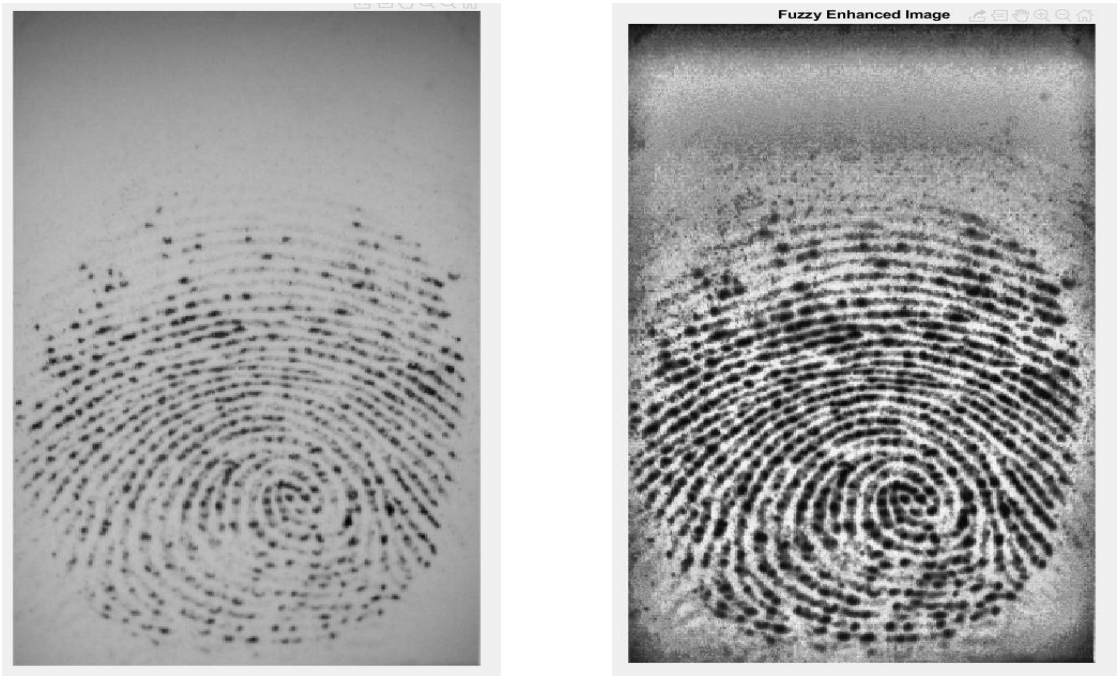


Figure 29 Fuzzy Enhanced Result Image Example 5

4.3 Feature Extraction GLCM and DWT2

4.3.1 GLCM

Gray Level Co-Occurrence Matrix (GLCM) has been used for strong and exact classification. This efficient method helps to evaluate the extracted features and removal of spurious minutiae. One the most important reasons for using this algorithm is because of its distinctive way of summarizing fundamental characteristics.

Co-occurrence matrix are often used for extraction as because they are composed of regular texture and its strong, effective and distinctive way of characterizing the features. Fingerprint analysis is an important and challenging problem and even though with the existence of many methods, there has been noticeable study that a better feature extraction technique may lead to very good result. Appropriation of pixel dark levels can be depicted by second-request insights like the likelihood of two pixels having specific dim levels at specific spatial connections. This data can be abridged in two-dimensional dim level co-event grids, which can be processed for different separations and directions. To utilize data contained in the GLCM. It calculates the Euclidean distance across the two centroids of the image and evaluate the features.

GLCM is a factual methodology that can depict second-request insights of a textured image. GLCM is fundamentally a 2-dimensional histogram in which each component is the recurrence that co-happens with the occasion component. A co-event lattice is indicated by the relative frequencies. The most important part is to remove the fake features extracted from the fingerprint image. GLCM and DWT is implemented in pursuance to decocture components for the purpose of the classification of the features.

4.3.2 DWT2

Discrete wavelet Transformation extract features using high pass and low pass filter, in this algorithm single level 2-dimensional wavelet transformation is been used along with the Haar wavelet which is a succession of rescaled square molded capacities all together or premise. DWT2 as a feature extraction algorithm is used as due to its n levels of data filtering technique it receives approximation coefficient and detailed coefficients. Features extracted from DWT2 coefficients are considered useful to input to classifier.

DWT2 returns the approximation detailed coefficient matrix such as cA, cH, cV, cD termed as

horizontal, vertical, diagonal and approximation coefficients.

Various features extracted from the image is subjected to preprocessing part first upon which it is subdivided such that there is a division and subdivision of the blocks according to the pixelwise operation.

The Euclidean distance is what relates to regular experience and recognitions. That is, the sort of single or multi-level dimensional direct metric presence where the partition between any two points in space identifies with the length of a straight line drawn between them. In the wake of figuring the separation between two sets of focuses, for which there is least separation those will be coordinated.

Code implementation of GLCM and DWT2

```
%% Feature extraction (GLCM+DWT)

[cA,cH,cV,cD] = dwt2(outImg,'haar');
cA = mean2(cA);
cH = mean2(cH);
cV = mean2(cV);
cD = mean2(cD);

Mean = mean2(outImg);
Standard_Deviation = std2(outImg);
Entropy = entropy(outImg);
Variance = mean2(var(double(outImg)));
a = sum(double(outImg(:)));
Smoothness = 1-(1/(1+a));
Kurtosis = kurtosis(double(outImg(:)));
Skewness = skewness(double(outImg(:)));
patients_with_vomiting=10.5;
patients_without_vomiting=11.5;
% Inverse Difference Movement
m = size(outImg,1);
n = size(outImg,2);
in_diff = 0;
for i = 1:m
    for j = 1:n
        temp = outImg(i,j)./(1+(i-j).^2);
        in_diff = in_diff+temp;
    end
end
IDM = double(in_diff);

feat_ext = [Mean, Standard_Deviation, Entropy,Variance, Smoothness, Kurtosis,
Skewness, IDM, cA, cH,cV,cD];
save feat_ext feat_ext
```

CHAPTER 5.

DEEP NEURAL NETWORKS CLASSIFICATION

Deep learning is an incredible breakthrough in the field of artificial intelligence. Its applications have been widely utilized in every one of the fields. In this work I have utilized different Deep Learning Neural Network Classification technique, which mainly helps to analyze the computational value and compare the results to find the best possible outcome using Feedforward Artificial Neural Network, Neural Network, Recurrent Neural Network.

Deep Learning operate on different types of Neural Networks which functions like human brains nervous system. It carries weights arranged and modified according to the training requirement. NNs works well with the enormous data as it can be trained with extra layers unlike Machine Learning where training can be performed till a specific degree of extent. In NN training the productivity increases with addition of more layers in the training process, as a result of which even the accuracy increases with the more training.

The three types of NNs have been utilized in this work so as to measure the accuracy of the results using different classifiers and find the best possible solution. Experiments have been performed in MATLAB 2019 academic edition. The three types of NNs utilized in this thesis work are Artificial Neural Network (ANN), two-layer Neural Network (NN), and Recurrent Neural Network (RNN). Results show considerable amount of improvement in all the classifiers and also matching of the image is performed to test the accuracy scores of the trained verses test results.

5.1 Neural Networks

The Neural Networks which functions as a human brains nervous system has input layer and output layer where we get the final predicted output. In between there are hidden layers where most of the computational work takes place which leads to the data processing which is known as training. Now the input neurons of one layer is connected to the other layers through channels, each of these channels assigned to a numeric value known as weights. Input value is multiplied to the corresponding weights and their sum is sent as input to the neurons in the hidden layer. Every one of these neurons is related with a numeric worth called as the bias, which is then joined to the info aggregate, through a threshold edge capacity called as activation function.

The outcome of the activation function checks in the event that the specific neuron gets actuated or not and, at that point enacted neuron transmits information to the neuron of the following layer over the channels. The output value obtained is usually termed as a probability value.

Propagation through channels without looking back is known as forward propagation and ANNs operates on the feedforward neural network principle. The backpropagation operation works on the principle where the operation can also go backwards in order to check the previous steps is termed as backpropagation and RNN works on backpropagation principle.

5.1.1 Artificial Neural Networks (ANN)

It is one of the simplest and straightforward NN which does not follow backpropagation. The data moves through several channels known as input nodes in single direction until it arrives to the final destination where it gets the output. It works on the front propagation principle.

The reason of using ANN is especially in pattern recognition since patterns are difficult to characterize. ANNs works well the distorted data and easy to characterize.

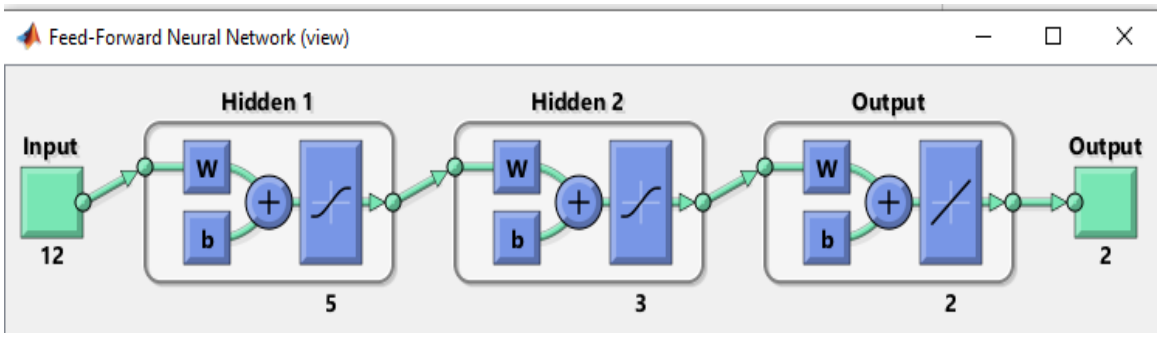


Figure 30 Feed forward NN view

The image of Feedforward NN shows the input, output and two hidden layers distributed over the channels.

5.1.2 Neural Networks (NN)

Neural Network is operated on two hidden layers. NN works on the Feedforward propagation principle.

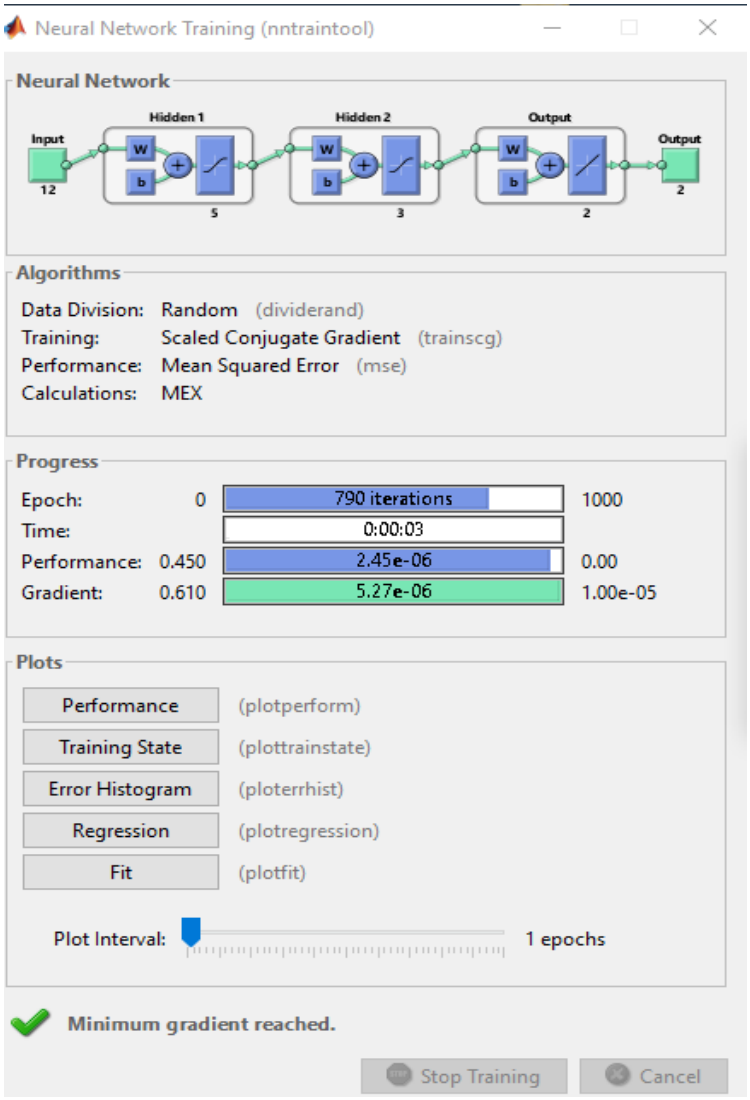


Figure 31 Neural Network Training

Figure 31 shows the Neural Network training example, epoch shows up to 1000 iterations, time taken is less than 3 secs, performance is 0.450 and the gradient is 0.610.

To train the image first input image is read from the data file and then image is resized.

```
*input_image = imread(['DATASETS\',num2str(yi),'.tif']);
```

For training and testing, first a pattern recognition network is created, division is setup for training, validation and testing for the list of all data division function type. The training ratio taken is 80/100 and test ratio is 20/100.

```
net.divideParam.trainRatio = 80/100;
net.divideParam.testRatio = 20/100;
```

Mean Square error is implemented as a Performance function. Training testing and validation is done and result of test worth is contrasted and the trained evaluation and shows whether the true parameter estimation of test parameter after the spurious minutiae removal is matched or not with the trained values.

5.1.3 Neural Networks (NN)

Neural Network is a process of estimations that attempts to see concealed associations of the dataset via operation that copies the way wherein the human intellect functions. In this way NN allude to the framework of neurons either natural or fake in nature. The system adjusts to the evolving input so that the system produces most ideal outcome without expecting to update the yield criteria. The idea of neural systems, which has its underlying foundations in manmade brainpower, is quickly picking up fame in the improvement of exchanging frameworks.
A neural system works likewise to the human mind's neural system. A "neuron" in a neural system is a scientific capacity that collects and orders data as per a particular design. A neural system holds layers of coordinated hubs. Every node act as a perceptron. The perceptron encourages the sign delivered by a different direct relapse into an activation function that might be nonlinear.

The info layer gathers information designs. The yield layer has orders or yield sign to which information examples may outline. The input data is tweaked in the hidden layers till the error rate gets negligible. It is estimated that shrouded layers extrapolate striking highlights in the information that have prescient power with respect to the outcome. This portrays highlight extraction, which achieves a utility like measurable procedures.

There will consistently be informational collections and errand classes that a superior investigated by utilizing recently created algorithms. It isn't so a lot of the algorithm that issues; it is the decidedly ready well managed input information on the focused-on marker that eventually decides the degree of accomplishment of a neural system.

5.1.4 Recurrent Neural Networks (RNN)

RNN is a group of frameworks where relationship between center points structure an organized graph along a transitory course of action. This authorizes it to display transient distinctive comportment. Not under any condition like feedforward neural frameworks, RNNs can use their internal state data to process the information sources.

It is utilized capriciously to insinuate two wide classes of frameworks with a similar general structure, where one is limited motivation and the other. The two classes of frameworks show passing powerful conduct. A constrained drive irregular framework is a planned non-cyclic graph that can be unrolled and displaced with a cautiously feedforward neural framework, while a relentless inspiration dull framework is an organized cyclic outline that can't be unrolled.

In RNNs is quite like the feedforward NN except the fact that it does a backpropagation to rectify the error, is it goes back checks the error and forward propagates to the final node and gives the outcome. Initial outcome is similar to that of feedforward, but it restores the data computed initially at the beginning of the process for future use. It follows the backpropagation rule and corrects the incorrect predictions.

5.2 Results

Table 1 Classifier accuracy

Type of Classifier	**Percentage of Accuracy**
Artificial Neural Network (ANN)	**96%**
Neural Network (NN)	93%
Recurrent Neural Network (RNN)	85%

Results in the Table 1 shows the percentage of the accuracy obtained by all the classifiers, Artificial Neural Network scored the highest possible accuracy scoring 96% when compared to the remaining classifiers, a simple Neural Network scored the second highest scoring 93%, and Recurrent Neural Network got the least accuracy of 85% comparing to the remaining classifiers.

Code Implementation:

```
%% ANN

% create a neural network
net = feedforwardnet([5 3]);
% train net
net.divideParam.trainRatio = 1; % training set [%]
net.divideParam.valRatio = 0; % validation set [%]
net.divideParam.testRatio = 0; % test set [%]
% train a neural network
%  For help on training function 'trainscg' type: help trainscg
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainscg';  % Scaled conjugate gradient

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};

net.efficiency.memoryReduction = 100;
net.trainParam.max_fail = 6;
net.trainParam.min_grad=1e-5;
net.trainParam.show=10;
net.trainParam.lr=0.9;
net.trainParam.epochs=1000;
net.trainParam.goal=0.00;

inputs = Train_feat';
targets = target';

% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)
```

```
% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)

% View the Network
view(net)
disp('after training')
y1 = sim(net,inputs);
y1=abs(y1);
y1=round(y1);

save  net net;

v1=sim(net,p);
disp(v1);
[c I]=max(v1);
disp(I);

if I==1
    disp('Finger print matched');
    msgbox('Finger print matched');
else
    disp('Finger print  not matched');
    msgbox('Finger print  not matched');
end

%% Performance Analysis for ANN
disp('Performance Analysis for ANN=');
load label
load result_ann
cp = classperf(label,result_ann)
accuracy = cp.CorrectRate
sensitivity = cp.Sensitivity
specificity = cp.Specificity
confusionmat2 = confusionmat(label,result_ann)
figure(8);
% plotConfMat2(confusionmat2,result_ann)
plotConfMat2(confusionmat2,result_ann)
eer=1-accuracy
species=label;
meas=result_ann;
load result1
pred = meas(51:end,1:2);
resp = (1:100)'>30;
mdl = fitglm(pred,resp,'Distribution','binomial','Link','logit');
scores = mdl.Fitted.Probability;
[X,Y,T,AUC] = perfcurve(species(51:end,:),scores,'virginica');
figure(9)
plot(X,Y)
xlabel('False positive rate')
ylabel('Sensitivity')
title('ROC for  ANN Classification')
```

Nntraining

```
rng('default');

load('Train_feat.mat');
load('target.mat');
net = feedforwardnet([5 3]);
inputs = Train_feat';
targets = target';


% Create a Pattern Recognition Network
hiddenLayerSize = 39;
net = patternnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 80/100;
net.divideParam.testRatio = 20/100;

% For help on training function 'trainscg' type: help trainscg
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainscg';  % Scaled conjugate gradient

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};

net.efficiency.memoryReduction = 100;
net.trainParam.max_fail = 6;
net.trainParam.min_grad=1e-5;
net.trainParam.show=10;
net.trainParam.lr=0.9;
net.trainParam.epochs=1000;
net.trainParam.goal=0.00;


% Train the Network
[net,tr] = train(net,inputs,targets);
```

```
% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)

% View the Network
view(net)


disp('after training')
y1 = sim(net,inputs);
y1=abs(y1);
y1=round(y1);

save  net net;

%%%%%%
```

Training.m

```
for yi = 1:100
       yi
%% To read a image from file...
input_image = imread(['DATASETS\',num2str(yi),'.tif']);
figure(1);
imshow(input_image);
title('Input image');

%%  Image Resize
input_image_res=imresize(input_image,[256 256]);
figure(2);
imshow(input_image_res);
title('Image Resize');
```

```
%% Performance metrics for nn

disp('Performance Analysis for NN=');
load result_nn

cp = classperf(label,result_nn)
accuracy = cp.CorrectRate
specificity = cp.Sensitivity
specificity = cp.Specificity
confusionmat = confusionmat(label,result_nn)
figure(10);
plotConfMat(confusionmat,result_nn)
eer=1-accuracy
species=label;
meas=result_nn;
load result1
pred = meas(51:end,1:2);
resp = (1:100)'>20;
mdl = fitglm(pred,resp,'Distribution','binomial','Link','logit');
scores = mdl.Fitted.Probability;
[X,Y,T,AUC] = perfcurve(species(51:end,:),scores,'virginica');
figure(11)
plot(X,Y)
xlabel('False positive rate')
ylabel('Sensitivity')
title('ROC for  NN Classification')
```

```
%% RNN
YTrain=target';
XTrain=Train_feat';
XTest=feat_ext';
inputSize = 1;
XValidation=Train_feat';
YValidation=label';
embeddingDimension = 100;
numWords = 500;
numHiddenUnits = 180;

numClasses = 2;
%
layers = [ sequenceInputLayer(inputSize)
    lstmLayer(1,'OutputMode','last')
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer ]

options = trainingOptions('adam', ...
    'MaxEpochs',10, ...
    'GradientThreshold',1, ...
    'InitialLearnRate',0.01, ...
    'ValidationData',{XValidation,YValidation}, ...
    'Plots','training-progress', ...
    'Verbose',false);
% net = trainNetwork(XTrain,YTrain,layers,options);
net=sim(net,XTest)
[D I]=max(net);
disp(I);

if I==1
    disp('Finger print matched');
    msgbox('Finger print matched');
else
    disp('Finger print  not matched');
    msgbox('Finger print  not matched');
end
clear all;
```

```
%% Performance Analysis for RNN
disp('Performance Analysis for RNN=');
load result_rnn
load target
load Train_feat
load feat_ext
load label
cp = classperf(label,result_rnn)
accuracy = cp.CorrectRate
sensitivity = cp.Sensitivity
specificity = cp.Specificity
confusionmat1 = confusionmat(label,result_rnn)
figure(12);
plotConfMat11(confusionmat1,result_rnn)
eer=1-accuracy
species=label;
meas=result_rnn;
load result1
pred = meas(51:end,1:2);
resp = (1:100)'>50;
mdl = fitglm(pred,resp,'Distribution','binomial','Link','logit');
scores = mdl.Fitted.Probability;
[X,Y,T,AUC] = perfcurve(species(51:end,:),scores,'virginica');
figure(13)
plot(X,Y)
xlabel('False positive rate')
ylabel('Sensitivity')
title('ROC for RNN Classification')

%%%%%%%%%%%%%%%%%%%
```

CHAPTER 6.

CONCLUSION AND RESULTS

The algorithm helps in precise mapping and classification of the fingerprint images. Utilizing morphological operations false minutiae is removed and true features are preserved using GLCM and 2-dimensional DWT operational algorithms. Fuzzy logic-based image enhancement technique ensures good quality fingerprint images in succession increase the classification and detection of the highest accuracy. The overall classification is comparatively taken less computational time.

The Process flow begins with Pre-processing first and then filtering which acts as a double filtering technique and operated twice in order to make sure the spurious part is detected properly and eliminated preserving the true values characteristics. After performing the morphological functions image is set to Fuzzy based enhancement. Total 100 images were taken in the dataset for the training purposes. The feature extraction algorithm like GLCM and DWT2 extracts different features as the more the features are the better the classification will be performed. The Classification results gives the highest result after using ANN classification. While performing the classification fingerprint matching is been performed to compare the test verses trained features of the image and a message dialogue box pops up after every operation of each classifier.

Execution of current fingerprint acknowledgment frameworks is vigorously influenced by the precision of their characteristic's extraction evaluation. These days, there are more ways to deal with fingerprint analysis with worthy outcomes. Issues begin to emerge in low-quality conditions where the dominant part of the conventional strategies dependent on examining the surface of fingerprint can't handle this issue so effectively as Neural Networks.

The advantages of the work include Better image enhancement technique ensures high quality fingerprint images which will in turn increases the classification and detection accuracy. Precise mapping and classification of fingerprint images.

6.1 Fingerprint Matching

A feature particulars-based matching is utilized to acquire the matching across the training and the testing images, and a matching dialogue box pops up on particulars highlights likewise processed. At that point, layouts of both examples and test pictures are changed by bringing in the centroids of the separated dabs and incipient in proximity with qualities containing position, heading on the first arrangement is figured. Note dabs will be utilized to set up arrangement when particulars are not accessible in the incomplete prints. To acquire the best execution matcher and profoundly connected, there is no requirement for score standardization.

The fingerprint test verses trained feature results shows whether the image is matched or not with its trained features from the dataset. There are different types of sensors available in Biometrics. Fingerprint's unique image acknowledgment is one of the first and unique biometric advancements that have been assembled freely under digital forensics legal sciences. A matching image will generate improved reconnaissance and conceivable further activity. For the framework to be compelling, the coordinating database ought to be as wide and thorough as would be prudent. There are numerous ways to deal with a biometric-based client fingerprints verification system and all it requires some type of equipment input gadget to accumulate the necessary data about the client to be confirmed. For instance, unique finger impression acknowledgment requires a finger impression scanner. These equipment gadgets must be an indispensable piece of the CR stage over a protected channel. Huge numbers of these gadgets are highly used and very frequently used in Biometrics. The channel interfacing the equipment hardware and programming must be equipped for supporting the information move prerequisites without an undue exhibition sway on the gadget's center usefulness.

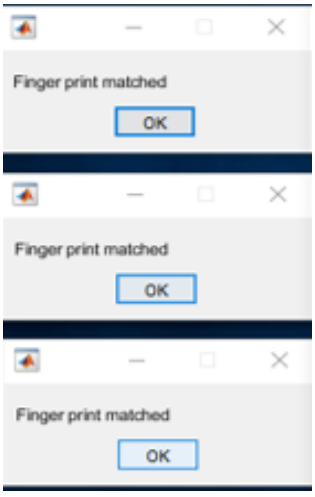


Figure 31 Shows the dialogue box of fingerprint matched for all the three classifiers

6.2 Results and Graphs

Receiver Operating Characteristics known as ROC curve illustrate the effectiveness of the initiated strategy. In classification to evaluate the execution performance visually this curve comes into picture. It is one of the most effective function used to analyze the performance of any Deep Learning or Machine Learning Classification. The purpose of the roc curve ascertains the Receiver Operating Characteristic % bend, which speaks to the 1-particularity and affectability, of two classes % of information, called class_1 and class_2. It is a graphical representation that defines the indicative magnitude of a binary classifier framework as per its threshold value which is fluctuated. The plot is curved with respect to TPR and FPR which is true positive rate and false positive rate respectively at different threshold values.

6.2.1 Receiver Operating Characteristics (ROC Curve)

ROC provides decision to select conceivably perfect models and to discard defective ones uninhibitedly from the previous and deciding the cost setting or the class spread. Its evaluation is associated in a quick and characteristic manner to cost effective analysis of characteristic fundamental administration. The curve is plotted for all the three classifiers showing the best results in all specially ANN. For each image we test in MATLAB ROC curve is plotted for all the classifiers. A good result shows the more value closer to 1, so if the value is close to 1 the higher is the accuracy to be considered.

The formula to measure the False positive rate and Sensitivity is described below.

FPR=FP/(FP+TN)

Sensitivity=TP/(TP+FN)

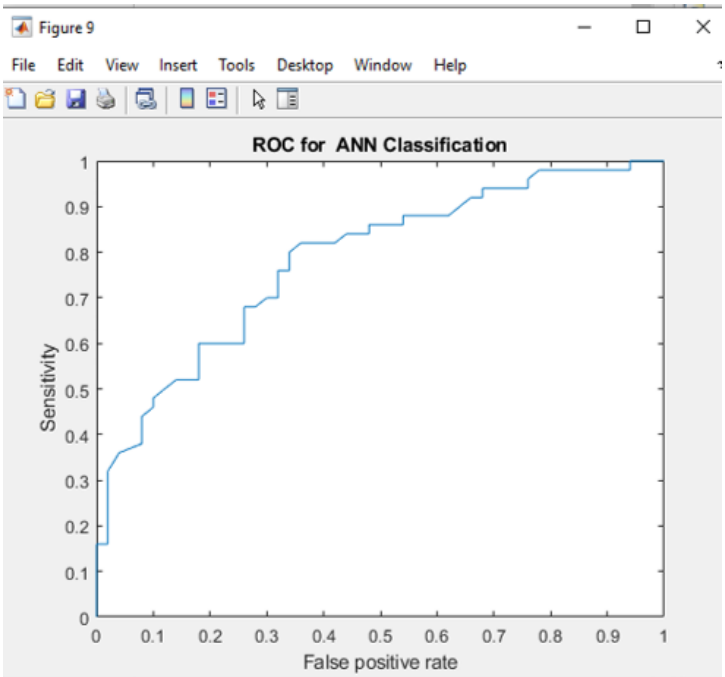


Figure 32 ANN Roc Curve

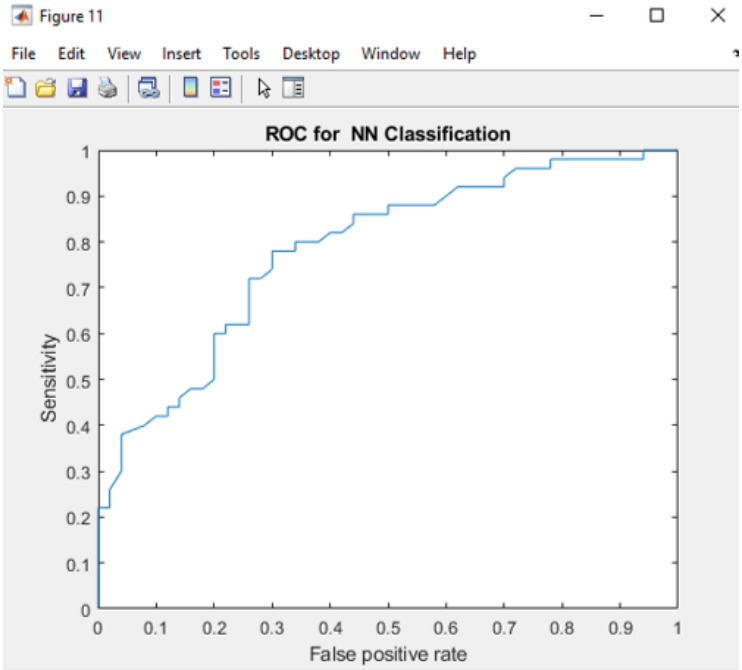


Figure 33 NN Roc Curve

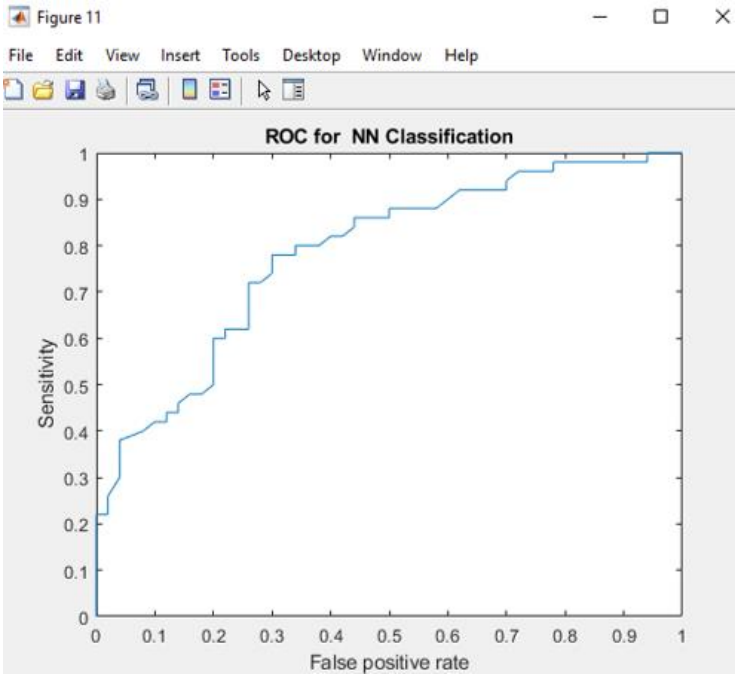


Figure 34 RNN Roc Curve

6.2.2 Equal Error Rate (EER)

Equal Error Rate is a biometric security framework algorithm utilized to evaluate the threshold value parameters between the false acceptance rate and false rejection rate. Here lowers the error rate higher is the accuracy to be considered. The more the value close to 0, the more efficient is the performance.

At the point when the rates are equivalent, the regular worth is alluded to as the equivalent mistake rate. The worth shows that the extent of false acknowledgments is equivalent to the extent of false dismissals. The lower the equivalent mistake rate esteem, the higher the exactness of the biometric framework. The Table 2 below describes the EER metrices of all the classifiers and once again ANN proves to be having the best results when compared to the remaining classifiers.

Table 2 EER

Type Of NN	Equal Error Rate (EER)
Artificial Neural Network (ANN)	0.0400
Neural Network (NN)	0.0700
Recurrent Neural Network (RNN)	1.5000

CODE IMPLEMENTATION

Function: Fuzzy.m

```
function out = fuzzyy(varargin)
%fuzzyy Contrast-limited.
%   fuzzyy enhances the contrast of images by transforming the
%   values in the intensity image I.  Unlike HISTEQ, it operates on small
%   data regions (tiles), rather than the entire image. Each tile's
%   contrast is enhanced, so that the histogram of the output region
%   approximately matches the specified histogram. The neighboring tiles
%   are then combined using bilinear interpolation in order to eliminate
%   artificially induced boundaries.  The contrast, especially
%   in homogeneous areas, can be limited in order to avoid amplifying the
%   noise which might be present in the image.
%
%   J = fuzzyy(I) Performs CLAHE on the intensity image I.
%
%   J = fuzzyy(I,PARAM1,VAL1,PARAM2,VAL2...) sets various parameters.
%   Parameter names can be abbreviated, and case does not matter. Each
%   string parameter is followed by a value as indicated below:
%
%   'NumTiles'     Two-element vector of positive integers: [M N].
%                  [M N] specifies the number of tile rows and
%                  columns.  Both M and N must be at least 2.
%                  The total number of image tiles is equal to M*N.
%
%                  Default: [8 8].
%
%   'ClipLimit'    Real scalar from 0 to 1.
%                  'ClipLimit' limits contrast enhancement. Higher numbers
%                  result in more contrast.
%
%                  Default: 0.01.
%
%   'NBins'        Positive integer scalar.
%                  Sets number of bins for the histogram used in building a
%                  contrast enhancing transformation. Higher values result
%                  in greater dynamic range at the cost of slower processing
%                  speed.
%
%                  Default: 256.
%
%   'Range'        One of the strings: 'original' or 'full'.
%                  Controls the range of the output image data. If 'Range'
%                  is set to 'original', the range is limited to
%                  [min(I(:)) max(I(:))]. Otherwise, by default, or when
%                  'Range' is set to 'full', the full range of the output
%                  image class is used (e.g. [0 255] for uint8).
%
%                  Default: 'full'.
%
%   'Distribution' Distribution can be one of three strings: 'uniform',
%                  'rayleigh', 'exponential'.
%                  Sets desired histogram shape for the image tiles, by
%                  specifying a distribution type.
%
%                  Default: 'uniform'.
```

```
%
%   'Alpha'        Nonnegative real scalar.
%                  'Alpha' is a distribution parameter, which can be supplied
%                  when 'Dist' is set to either 'rayleigh' or 'exponential'.
%
%                  Default: 0.4.
%
%   Notes
%   -----
%   - 'NumTiles' specify the number of rectangular contextual regions (tiles)
%     into which the image is divided. The contrast transform function is
%     calculated for each of these regions individually. The optimal number
of
%     tiles depends on the type of the input image, and it is best determined
%     through experimentation.
%
%   - The 'ClipLimit' is a contrast factor that prevents over-saturation of
the
%     image specifically in homogeneous areas.  These areas are characterized
%     by a high peak in the histogram of the particular image tile due to
many
%     pixels falling inside the same gray level range. Without the clip
limit,
%     the adaptive histogram equalization technique could produce results
that,
%     in some cases, are worse than the original image.
%
%   - fuzzyy can use Uniform, Rayleigh, or Exponential distribution as
%     the basis for creating the contrast transform function. The
distribution
%     that should be used depends on the type of the input image.
%     For example, underwater imagery appears to look more natural when the
%     Rayleigh distribution is used.
%
%   Class Support
%   -------------
%   Intensity image I can be uint8, uint16, int16, double, or single.
%   The output image J has the same class as I.
%
%   Example 1
%   ---------
%   Apply Contrast-Limited Adaptive Histogram Equalization to an
%   image and display the results.
%
%      I = imread('tire.tif');
%      A = fuzzyy(I,'clipLimit',0.02,'Distribution','rayleigh');
%      figure, imshow(I);
%      figure, imshow(A);
%
%   Example 2
%   ---------
%
%   Apply Contrast-Limited Adaptive Histogram Equalization to a color
%   photograph.
%
%      [X MAP] = imread('shadow.tif');
```

```
%       RGB = ind2rgb(X,MAP); % convert indexed image to truecolor format
%       cform2lab = makecform('srgb2lab');
%       LAB = applycform(RGB, cform2lab); %convert image to L*a*b color space
%       L = LAB(:,:,1)/100; % scale the values to range from 0 to 1
%       LAB(:,:,1) = fuzzyy(L,'NumTiles',[8 8],'ClipLimit',0.005)*100;
%       cform2srgb = makecform('lab2srgb');
%       J = applycform(LAB, cform2srgb); %convert back to RGB
%       figure, imshow(RGB); %display the results
%       figure, imshow(J);
%
%   See also HISTEQ, IMHISTMATCH.

%   Copyright 1993-2012 The MathWorks, Inc.
%   $Revision: 1.1.6.13 $  $Date: 2012/03/01 02:21:53 $

%   References:
%      Karel Zuiderveld, "Contrast Limited Adaptive Histogram Equalization",
%      Graphics Gems IV, p. 474-485, code: p. 479-484
%
%      Hanumant Singh, Woods Hole Oceanographic Institution, personal
%      communication

%--------------------------- The algorithm ----------------------------------
%
%  1. Obtain all the inputs:
%    * image
%    * number of regions in row and column directions
%    * number of bins for the histograms used in building image transform
%      function (dynamic range)
%    * clip limit for contrast limiting (normalized from 0 to 1)
%    * other miscellaneous options
%  2. Pre-process the inputs:
%    * determine real clip limit from the normalized value
%    * if necessary, pad the image before splitting it into regions
%  3. Process each contextual region (tile) thus producing gray level
mappings
%    * extract a single image region
%    * make a histogram for this region using the specified number of bins
%    * clip the histogram using clip limit
%    * create a mapping (transformation function) for this region
%  4. Interpolate gray level mappings in order to assemble final CLAHE image
%    * extract cluster of four neighboring mapping functions
%    * process image region partly overlapping each of the mapping tiles
%    * extract a single pixel, apply four mappings to that pixel, and
%      interpolate between the results to obtain the output pixel; repeat
%      over the entire image
%
%  See code for further details.
%
%------------------------------------------------------------------------------
-
```

```
tileMappings = makeTileMappings(I, numTiles, dimTile, numBins, clipLimit, ...
                                selectedRange, fullRange, distribution,
alpha);

%Synthesize the output image based on the individual tile mappings.
out = makeClaheImage(I, tileMappings, numTiles, selectedRange, numBins,...
                     dimTile);

if int16ClassChange
  % Change uint16 back to int16 so output has same class as input.
  out = uint16toint16(out);
end

if ~isempty(noPadRect) %do we need to remove padding?
  out = out(noPadRect.ulRow:noPadRect.lrRow, ...
            noPadRect.ulCol:noPadRect.lrCol);
end

%------------------------------------------------------------------------
-

function tileMappings = ...
    makeTileMappings(I, numTiles, dimTile, numBins, clipLimit,...
                     selectedRange, fullRange, distribution, alpha)

numPixInTile = prod(dimTile);

tileMappings = cell(numTiles);

% extract and process each tile
imgCol = 1;
for col=1:numTiles(2),
  imgRow = 1;
  for row=1:numTiles(1),

    tile = I(imgRow:imgRow+dimTile(1)-1,imgCol:imgCol+dimTile(2)-1);

    % for speed, call MEX file directly thus avoiding costly
    % input parsing of imhist
    tileHist = imhistc(tile, numBins, 1, fullRange(2));

    tileHist = clipHistogram(tileHist, clipLimit, numBins);

    tileMapping = makeMapping(tileHist, selectedRange, fullRange, ...
                              numPixInTile, distribution, alpha);

    % assemble individual tile mappings by storing them in a cell array;
    tileMappings{row,col} = tileMapping;

    imgRow = imgRow + dimTile(1);
  end
  imgCol = imgCol + dimTile(2); % move to the next column of tiles
```

```
end

%-----------------------------------------------------------------------
-
% Calculate the equalized lookup table (mapping) based on cumulating the
input
% histogram.  Note: lookup table is rescaled in the selectedRange [Min..Max].

function mapping = makeMapping(imgHist, selectedRange, fullRange, ...
                               numPixInTile, distribution, alpha)

histSum = cumsum(imgHist);
valSpread  = selectedRange(2) - selectedRange(1);

switch distribution
 case 'uniform',
  scale =  valSpread/numPixInTile;
  mapping = min(selectedRange(1) + histSum*scale,...
                selectedRange(2)); %limit to max

 case 'rayleigh', % suitable for underwater imagery
  % pdf = (t./alpha^2).*exp(-t.^2/(2*alpha^2))*U(t)
  % cdf = 1-exp(-t.^2./(2*alpha^2))
  hconst = 2*alpha^2;
  vmax = 1 - exp(-1/hconst);
  val = vmax*(histSum/numPixInTile);
  val(val>=1) = 1-eps; % avoid log(0)
  temp = sqrt(-hconst*log(1-val));
  mapping = min(selectedRange(1)+temp*valSpread,...
                selectedRange(2)); %limit to max

 case 'exponential',
  % pdf = alpha*exp(-alpha*t)*U(t)
  % cdf = 1-exp(-alpha*t)
  vmax = 1 - exp(-alpha);
  val = (vmax*histSum/numPixInTile);
  val(val>=1) = 1-eps;
  temp = -1/alpha*log(1-val);
  mapping = min(selectedRange(1)+temp*valSpread,selectedRange(2));

 otherwise,
  error(message('images:fuzzyy:distributionType')) %should never get here

end

%rescale the result to be between 0 and 1 for later use by the GRAYXFORM
%private mex function
mapping = mapping/fullRange(2);

%-----------------------------------------------------------------------
-
% This function clips the histogram according to the clipLimit and
% redistributes clipped pixels across bins below the clipLimit
```

```
function imgHist = clipHistogram(imgHist, clipLimit, numBins)

% total number of pixels overflowing clip limit in each bin
totalExcess = sum(max(imgHist - clipLimit,0));

% clip the histogram and redistribute the excess pixels in each bin
avgBinIncr = floor(totalExcess/numBins);
upperLimit = clipLimit - avgBinIncr; % bins larger than this will be
                                     % set to clipLimit

% this loop should speed up the operation by putting multiple pixels
% into the "obvious" places first
for k=1:numBins
  if imgHist(k) > clipLimit
    imgHist(k) = clipLimit;
  else
    if imgHist(k) > upperLimit % high bin count
      totalExcess = totalExcess - (clipLimit - imgHist(k));
      imgHist(k) = clipLimit;
    else
      totalExcess = totalExcess - avgBinIncr;
      imgHist(k) = imgHist(k) + avgBinIncr;
    end
  end
end

% this loops redistributes the remaining pixels, one pixel at a time
k = 1;
while (totalExcess ~= 0)
  %keep increasing the step as fewer and fewer pixels remain for
  %the redistribution (spread them evenly)
  stepSize = max(floor(numBins/totalExcess),1);
  for m=k:stepSize:numBins
    if imgHist(m) < clipLimit
      imgHist(m) = imgHist(m)+1;
      totalExcess = totalExcess - 1; %reduce excess
      if totalExcess == 0
        break;
      end
    end
  end

  k = k+1; %prevent from always placing the pixels in bin #1
  if k > numBins % start over if numBins was reached
    k = 1;
  end
end

%------------------------------------------------------------------------
-
% This function interpolates between neighboring tile mappings to produce a
% new mapping in order to remove artificially induced tile borders.
% Otherwise, these borders would become quite visible.  The resulting
```

```
% mapping is applied to the input image thus producing a CLAHE processed
% image.

function claheI = makeClaheImage(I, tileMappings, numTiles, selectedRange,...
                                 numBins, dimTile)

%initialize the output image to zeros (preserve the class of the input image)
claheI = I;
claheI(:) = 0;

%compute the LUT for looking up original image values in the tile mappings,
%which we created earlier
if ~(isa(I,'double') || isa(I,'single'))
  k = selectedRange(1)+1 : selectedRange(2)+1;
  aLut = zeros(length(k),1);
  aLut(k) = (k-1)-selectedRange(1);
  aLut = aLut/(selectedRange(2)-selectedRange(1));
else
  % remap from 0..1 to 0..numBins-1
  if numBins ~= 1
    binStep = 1/(numBins-1);
    start = ceil(selectedRange(1)/binStep);
    stop  = floor(selectedRange(2)/binStep);
    k = start+1:stop+1;
    aLut(k) = 0:1/(length(k)-1):1;
  else
    aLut(1) = 0; %in case someone specifies numBins = 1, which is just silly
  end
end

imgTileRow=1;
for k=1:numTiles(1)+1
  if k == 1  %special case: top row
    imgTileNumRows = dimTile(1)/2; %always divisible by 2 because of padding
    mapTileRows = [1 1];
  else
    if k == numTiles(1)+1 %special case: bottom row
      imgTileNumRows = dimTile(1)/2;
      mapTileRows = [numTiles(1) numTiles(1)];
    else %default values
      imgTileNumRows = dimTile(1);
      mapTileRows = [k-1, k]; %[upperRow lowerRow]
    end
  end

  % loop over columns of the tileMappings cell array
  imgTileCol=1;
  for l=1:numTiles(2)+1
    if l == 1 %special case: left column
      imgTileNumCols = dimTile(2)/2;
      mapTileCols = [1, 1];
    else
      if l == numTiles(2)+1 % special case: right column
        imgTileNumCols = dimTile(2)/2;
        mapTileCols = [numTiles(2), numTiles(2)];
```

```
      else %default values
        imgTileNumCols = dimTile(2);
        mapTileCols = [1-1, 1]; % right left
      end
    end

    % Extract four tile mappings
    ulMapTile = tileMappings{mapTileRows(1), mapTileCols(1)};
    urMapTile = tileMappings{mapTileRows(1), mapTileCols(2)};
    blMapTile = tileMappings{mapTileRows(2), mapTileCols(1)};
    brMapTile = tileMappings{mapTileRows(2), mapTileCols(2)};

    % Calculate the new greylevel assignments of pixels
    % within a submatrix of the image specified by imgTileIdx. This
    % is done by a bilinear interpolation between four different mappings
    % in order to eliminate boundary artifacts.

    normFactor = imgTileNumRows*imgTileNumCols; %normalization factor
    imgTileIdx = {imgTileRow:imgTileRow+imgTileNumRows-1, ...
                  imgTileCol:imgTileCol+imgTileNumCols-1};

    imgPixVals = grayxform(I(imgTileIdx{1},imgTileIdx{2}), aLut);

    % calculate the weights used for linear interpolation between the
    % four mappings
    rowW = repmat((0:imgTileNumRows-1)',1,imgTileNumCols);
    colW = repmat(0:imgTileNumCols-1,imgTileNumRows,1);
    rowRevW = repmat((imgTileNumRows:-1:1)',1,imgTileNumCols);
    colRevW = repmat(imgTileNumCols:-1:1,imgTileNumRows,1);

    claheI(imgTileIdx{1}, imgTileIdx{2}) = ...
        (rowRevW .* (colRevW .* double(grayxform(imgPixVals,ulMapTile)) + ...
                     colW    .* double(grayxform(imgPixVals,urMapTile)))+ ...
         rowW    .* (colRevW .* double(grayxform(imgPixVals,blMapTile)) + ...
                     colW    .* double(grayxform(imgPixVals,brMapTile))))...
        /normFactor;

    imgTileCol = imgTileCol + imgTileNumCols;
  end %over tile cols
  imgTileRow = imgTileRow + imgTileNumRows;
end %over tile rows

%-----------------------------------------------------------------------------
-

function [I, selectedRange, fullRange, numTiles, dimTile, clipLimit,...
          numBins, noPadRect, distribution, alpha, ...
          int16ClassChange] = parseInputs(varargin)

narginchk(1,13);

I = varargin{1};
validateattributes(I, {'uint8', 'uint16', 'double', 'int16', 'single'}, ...
```

```
                {'real', '2d', 'nonsparse', 'nonempty'}, ...
                mfilename, 'I', 1);

% convert int16 to uint16
if isa(I,'int16')
  I = int16touint16(I);
  int16ClassChange = true;
else
  int16ClassChange = false;
end

if any(size(I) < 2)
  error(message('images:fuzzyy:inputImageTooSmall'))
end

%Other options
%%%%%%%%%%%%%%%

%Set the defaults
distribution = 'uniform';
alpha    = 0.4;

if isa(I, 'double') || isa(I,'single')
  fullRange = [0 1];
else
  fullRange(1) = I(1);          %copy class of the input image
  fullRange(1:2) = [-Inf Inf]; %will be clipped to min and max
  fullRange = double(fullRange);
end

selectedRange    = fullRange;

%Set the default to 256 bins regardless of the data type;
%the user can override this value at any time
numBins =128;
normClipLimit = 0.01;
numTiles = [8 8];

checkAlpha = false;

validStrings = {'NumTiles','ClipLimit','NBins','Distribution',...
                'Alpha','Range'};

if nargin > 1
  done = false;

  idx = 2;
  while ~done
    input = varargin{idx};
    inputStr = validatestring(input, validStrings,mfilename,'PARAM',idx);

    idx = idx+1; %advance index to point to the VAL portion of the input
```

```
    if idx > nargin
      error(message('images:fuzzyy:missingValue', inputStr))
    end

    switch inputStr

     case 'NumTiles'
       numTiles = varargin{idx};
       validateattributes(numTiles, {'double'}, {'real', 'vector', ...
                          'integer', 'finite','positive'},...
                     mfilename, inputStr, idx);

       if (any(size(numTiles) ~= [1,2]))
         error(message('images:fuzzyy:invalidNumTilesVector', inputStr))
       end

       if any(numTiles < 2)
         error(message('images:fuzzyy:invalidNumTilesValue', inputStr))
       end

     case 'ClipLimit'
      normClipLimit = varargin{idx};
      validateattributes(normClipLimit, {'double'}, ...
                    {'scalar','real','nonnegative'},...
                    mfilename, inputStr, idx);

      if normClipLimit > 1
        error(message('images:fuzzyy:invalidClipLimit', inputStr))
      end

     case 'NBins'
      numBins = varargin{idx};
      validateattributes(numBins, {'double'}, {'scalar','real','integer',...
                         'positive'}, mfilename, 'NBins', idx);

     case 'Distribution'
      validDist = {'rayleigh','exponential','uniform'};
      distribution = validatestring(varargin{idx}, validDist, mfilename,...
                                    'Distribution', idx);

     case 'Alpha'
      alpha = varargin{idx};
      validateattributes(alpha, {'double'},{'scalar','real',...
                         'nonnan','positive','finite'},...
                    mfilename, 'Alpha',idx);
      checkAlpha = true;

     case 'Range'
      validRangeStrings = {'original','full'};
      rangeStr = validatestring(varargin{idx},
validRangeStrings,mfilename,...
                                'Range',idx);
```

```
      if strmatch(rangeStr,'original')
        selectedRange = double([min(I(:)), max(I(:))]);
      end

     otherwise
      error(message('images:fuzzyy:inputString')) %should never get here
    end

    if idx >= nargin
      done = true;
    end

    idx=idx+1;
  end
end


%% Pre-process the inputs
%%%%%%%%%%%%%%%%%%%%%%%%%%

dimI = size(I);
dimTile = dimI ./ numTiles;

%check if tile size is reasonable
if any(dimTile < 1)
  error(message('images:fuzzyy:inputImageTooSmallToSplit',
num2str( numTiles )))
end

if checkAlpha
  if strcmp(distribution,'uniform')
    error(message('images:fuzzyy:alphaShouldNotBeSpecified', distribution))
  end
end

%check if the image needs to be padded; pad if necessary;
%padding occurs if any dimension of a single tile is an odd number
%and/or when image dimensions are not divisible by the selected
%number of tiles
rowDiv  = mod(dimI(1),numTiles(1)) == 0;
colDiv  = mod(dimI(2),numTiles(2)) == 0;

if rowDiv && colDiv
  rowEven = mod(dimTile(1),2) == 0;
  colEven = mod(dimTile(2),2) == 0;
end

noPadRect = [];
if  ~(rowDiv && colDiv && rowEven && colEven)
  padRow = 0;
  padCol = 0;

  if ~rowDiv
```

```
    rowTileDim = floor(dimI(1)/numTiles(1)) + 1;
    padRow = rowTileDim*numTiles(1) - dimI(1);
  else
    rowTileDim = dimI(1)/numTiles(1);
  end

  if ~colDiv
    colTileDim = floor(dimI(2)/numTiles(2)) + 1;
    padCol = colTileDim*numTiles(2) - dimI(2);
  else
    colTileDim = dimI(2)/numTiles(2);
  end

  %check if tile dimensions are even numbers
  rowEven = mod(rowTileDim,2) == 0;
  colEven = mod(colTileDim,2) == 0;

  if ~rowEven
    padRow = padRow+numTiles(1);
  end

  if ~colEven
    padCol = padCol+numTiles(2);
  end

  padRowPre  = floor(padRow/2);
  padRowPost = ceil(padRow/2);
  padColPre  = floor(padCol/2);
  padColPost = ceil(padCol/2);

  I = padarray(I,[padRowPre  padColPre ],'symmetric','pre');
  I = padarray(I,[padRowPost padColPost],'symmetric','post');

  %UL corner (Row, Col), LR corner (Row, Col)
  noPadRect.ulRow = padRowPre+1;
  noPadRect.ulCol = padColPre+1;
  noPadRect.lrRow = padRowPre+dimI(1);
  noPadRect.lrCol = padColPre+dimI(2);
end

%redefine this variable to include the padding
dimI = size(I);

%size of the single tile
dimTile = dimI ./ numTiles;

%compute actual clip limit from the normalized value entered by the user
%maximum value of normClipLimit=1 results in standard AHE, i.e. no clipping;
%the minimum value minClipLimit would uniformly distribute the image pixels
%across the entire histogram, which would result in the lowest possible
%contrast value
numPixInTile = prod(dimTile)
minClipLimit = ceil(numPixInTile/numBins);
clipLimit = minClipLimit + round(normClipLimit*(numPixInTile-minClipLimit));
```

```
%-------------------------------------------------------------------------
-
```

Fuzzy_opt.m

```
function out = Fuzzy_opt(im)
out=adapthisteq(im);

[m,n]=size(out);
for i=1:m
    for j=1:n
        if (out(i,j)<= 80)
            out(i,j)=out(i,j)+10;
        end
        if (out(i,j)>=160)
            out(i,j)=out(i,j)-10;
        end
        if (out(i,j)>80) || (out(i,j)<160)
            out(i,j)=out(i,j);
        end

    end
end

%%%%%%%%%
```

Fus.m

```
can you mail me function out = fus(varargin)
%fus Contrast-limited Adaptive Histogram Equalization (CLAHE).
%   fus enhances the contrast of images by transforming the
%   values in the intensity image I.  Unlike HISTEQ, it operates on small
%   data regions (tiles), rather than the entire image. Each tile's
%   contrast is enhanced, so that the histogram of the output region
%   approximately matches the specified histogram. The neighboring tiles
%   are then combined using bilinear interpolation in order to eliminate
%   artificially induced boundaries.  The contrast, especially
%   in homogeneous areas, can be limited in order to avoid amplifying the
%   noise which might be present in the image.
%
%   J = fus(I) Performs CLAHE on the intensity image I.
%
%   J = fus(I,PARAM1,VAL1,PARAM2,VAL2...) sets various parameters.
%   Parameter names can be abbreviated, and case does not matter. Each
%   string parameter is followed by a value as indicated below:
%
%   'NumTiles'     Two-element vector of positive integers: [M N].
%                  [M N] specifies the number of tile rows and
%                  columns.  Both M and N must be at least 2.
%                  The total number of image tiles is equal to M*N.
%
%                  Default: [8 8].
%
%   'ClipLimit'    Real scalar from 0 to 1.
%                  'ClipLimit' limits contrast enhancement. Higher numbers
%                  result in more contrast.
%
%                  Default: 0.01.
%
%   'NBins'        Positive integer scalar.
%                  Sets number of bins for the histogram used in building a
%                  contrast enhancing transformation. Higher values result
%                  in greater dynamic range at the cost of slower processing
%                  speed.
%
%                  Default: 256.
%
%   'Range'        One of the strings: 'original' or 'full'.
%                  Controls the range of the output image data. If 'Range'
%                  is set to 'original', the range is limited to
%                  [min(I(:)) max(I(:))]. Otherwise, by default, or when
%                  'Range' is set to 'full', the full range of the output
%                  image class is used (e.g. [0 255] for uint8).
%
%                  Default: 'full'.
%
%   'Distribution' Distribution can be one of three strings: 'uniform',
%                  'rayleigh', 'exponential'.
%                  Sets desired histogram shape for the image tiles, by
%                  specifying a distribution type.
```

```
%                   Default: 'uniform'.
%
%   'Alpha'         Nonnegative real scalar.
%                   'Alpha' is a distribution parameter, which can be supplied
%                   when 'Dist' is set to either 'rayleigh' or 'exponential'.
%
%                   Default: 0.4.
%
%   Notes
%   -----
%   - 'NumTiles' specify the number of rectangular contextual regions (tiles)
%     into which the image is divided. The contrast transform function is
%     calculated for each of these regions individually. The optimal number
of
%     tiles depends on the type of the input image, and it is best determined
%     through experimentation.
%
%   - The 'ClipLimit' is a contrast factor that prevents over-saturation of
the
%     image specifically in homogeneous areas.  These areas are characterized
%     by a high peak in the histogram of the particular image tile due to
many
%     pixels falling inside the same gray level range. Without the clip
limit,
%     the adaptive histogram equalization technique could produce results
that,
%     in some cases, are worse than the original image.
%
%   - fus can use Uniform, Rayleigh, or Exponential distribution as
%     the basis for creating the contrast transform function. The
distribution
%     that should be used depends on the type of the input image.
%     For example, underwater imagery appears to look more natural when the
%     Rayleigh distribution is used.
%
%   Class Support
%   -------------
%   Intensity image I can be uint8, uint16, int16, double, or single.
%   The output image J has the same class as I.
%
%   Example 1
%   ---------
%   Apply Contrast-Limited Adaptive Histogram Equalization to an
%   image and display the results.
%
%      I = imread('tire.tif');
%      A = fus(I,'clipLimit',0.02,'Distribution','rayleigh');
%      figure, imshow(I);
%      figure, imshow(A);
%
%   Example 2
%   ---------
%
%   Apply Contrast-Limited Adaptive Histogram Equalization to a color
%   photograph.
```

```
%
%       [X MAP] = imread('shadow.tif');
%       RGB = ind2rgb(X,MAP); % convert indexed image to truecolor format
%       cform2lab = makecform('srgb2lab');
%       LAB = applycform(RGB, cform2lab); %convert image to L*a*b color space
%       L = LAB(:,:,1)/100; % scale the values to range from 0 to 1
%       LAB(:,:,1) = fus(L,'NumTiles',[8 8],'ClipLimit',0.005)*100;
%       cform2srgb = makecform('lab2srgb');
%       J = applycform(LAB, cform2srgb); %convert back to RGB
%       figure, imshow(RGB); %display the results
%       figure, imshow(J);
%
%   See also HISTEQ, IMHISTMATCH.

%   Copyright 1993-2012 The MathWorks, Inc.
%   $Revision: 1.1.6.13 $  $Date: 2012/03/01 02:21:53 $

%   References:
%      Karel Zuiderveld, "Contrast Limited Adaptive Histogram Equalization",
%      Graphics Gems IV, p. 474-485, code: p. 479-484
%
%      Hanumant Singh, Woods Hole Oceanographic Institution, personal
%      communication

%--------------------------- The algorithm -----------------------------------
%
%  1. Obtain all the inputs:
%    * image
%    * number of regions in row and column directions
%    * number of bins for the histograms used in building image transform
%      function (dynamic range)
%    * clip limit for contrast limiting (normalized from 0 to 1)
%    * other miscellaneous options
%  2. Pre-process the inputs:
%    * determine real clip limit from the normalized value
%    * if necessary, pad the image before splitting it into regions
%  3. Process each contextual region (tile) thus producing gray level
mappings
%    * extract a single image region
%    * make a histogram for this region using the specified number of bins
%    * clip the histogram using clip limit
%    * create a mapping (transformation function) for this region
%  4. Interpolate gray level mappings in order to assemble final CLAHE image
%    * extract cluster of four neighboring mapping functions
%    * process image region partly overlapping each of the mapping tiles
%    * extract a single pixel, apply four mappings to that pixel, and
%      interpolate between the results to obtain the output pixel; repeat
%      over the entire image
%
%  See code for further details.
%
%-------------------------------------------------------------------------------
-

[I, selectedRange, fullRange, numTiles, dimTile, clipLimit, numBins, ...
```

```
 noPadRect, distribution, alpha, int16ClassChange] =
parseInputs(varargin{:});

tileMappings = makeTileMappings(I, numTiles, dimTile, numBins, clipLimit, ...
                                selectedRange, fullRange, distribution,
alpha);

%Synthesize the output image based on the individual tile mappings.
out = makeClaheImage(I, tileMappings, numTiles, selectedRange, numBins,...
                     dimTile);

if int16ClassChange
  % Change uint16 back to int16 so output has same class as input.
  out = uint16toint16(out);
end

if ~isempty(noPadRect) %do we need to remove padding?
  out = out(noPadRect.ulRow:noPadRect.lrRow, ...
            noPadRect.ulCol:noPadRect.lrCol);
end

%-----------------------------------------------------------------------------
-

function tileMappings = ...
    makeTileMappings(I, numTiles, dimTile, numBins, clipLimit,...
                     selectedRange, fullRange, distribution, alpha)

numPixInTile = prod(dimTile);

tileMappings = cell(numTiles);

% extract and process each tile
imgCol = 1;
for col=1:numTiles(2),
  imgRow = 1;
  for row=1:numTiles(1),

    tile = I(imgRow:imgRow+dimTile(1)-1,imgCol:imgCol+dimTile(2)-1);

    % for speed, call MEX file directly thus avoiding costly
    % input parsing of imhist
    tileHist = imhistc(tile, numBins, 1, fullRange(2));

    tileHist = clipHistogram(tileHist, clipLimit, numBins);

    tileMapping = makeMapping(tileHist, selectedRange, fullRange, ...
                              numPixInTile, distribution, alpha);

    % assemble individual tile mappings by storing them in a cell array;
    tileMappings{row,col} = tileMapping;

    imgRow = imgRow + dimTile(1);
```

```
  end
  imgCol = imgCol + dimTile(2); % move to the next column of tiles
end

%---------------------------------------------------------------------------
-
% Calculate the equalized lookup table (mapping) based on cumulating the
input
% histogram.  Note: lookup table is rescaled in the selectedRange [Min..Max].

function mapping = makeMapping(imgHist, selectedRange, fullRange, ...
                               numPixInTile, distribution, alpha)

histSum = cumsum(imgHist);
valSpread  = selectedRange(2) - selectedRange(1);

switch distribution
 case 'uniform',
  scale =  valSpread/numPixInTile;
  mapping = min(selectedRange(1) + histSum*scale,...
                selectedRange(2)); %limit to max

 case 'rayleigh', % suitable for underwater imagery
  % pdf = (t./alpha^2).*exp(-t.^2/(2*alpha^2))*U(t)
  % cdf = 1-exp(-t.^2./(2*alpha^2))
  hconst = 2*alpha^2;
  vmax = 1 - exp(-1/hconst);
  val = vmax*(histSum/numPixInTile);
  val(val>=1) = 1-eps; % avoid log(0)
  temp = sqrt(-hconst*log(1-val));
  mapping = min(selectedRange(1)+temp*valSpread,...
                selectedRange(2)); %limit to max

 case 'exponential',
  % pdf = alpha*exp(-alpha*t)*U(t)
  % cdf = 1-exp(-alpha*t)
  vmax = 1 - exp(-alpha);
  val = (vmax*histSum/numPixInTile);
  val(val>=1) = 1-eps;
  temp = -1/alpha*log(1-val);
  mapping = min(selectedRange(1)+temp*valSpread,selectedRange(2));

 otherwise,
  error(message('images:fus:distributionType')) %should never get here

end

%rescale the result to be between 0 and 1 for later use by the GRAYXFORM
%private mex function
mapping = mapping/fullRange(2);

%---------------------------------------------------------------------------
-
% This function clips the histogram according to the clipLimit and
```

```
% redistributes clipped pixels across bins below the clipLimit

function imgHist = clipHistogram(imgHist, clipLimit, numBins)

% total number of pixels overflowing clip limit in each bin
totalExcess = sum(max(imgHist - clipLimit,0));

% clip the histogram and redistribute the excess pixels in each bin
avgBinIncr = floor(totalExcess/numBins);
upperLimit = clipLimit - avgBinIncr; % bins larger than this will be
                                     % set to clipLimit

% this loop should speed up the operation by putting multiple pixels
% into the "obvious" places first
for k=1:numBins
  if imgHist(k) > clipLimit
    imgHist(k) = clipLimit;
  else
    if imgHist(k) > upperLimit % high bin count
      totalExcess = totalExcess - (clipLimit - imgHist(k));
      imgHist(k) = clipLimit;
    else
      totalExcess = totalExcess - avgBinIncr;
      imgHist(k) = imgHist(k) + avgBinIncr;
    end
  end
end

% this loops redistributes the remaining pixels, one pixel at a time
k = 1;
while (totalExcess ~= 0)
  %keep increasing the step as fewer and fewer pixels remain for
  %the redistribution (spread them evenly)
  stepSize = max(floor(numBins/totalExcess),1);
  for m=k:stepSize:numBins
    if imgHist(m) < clipLimit
      imgHist(m) = imgHist(m)+1;
      totalExcess = totalExcess - 1; %reduce excess
      if totalExcess == 0
        break;
      end
    end
  end

  k = k+1; %prevent from always placing the pixels in bin #1
  if k > numBins % start over if numBins was reached
    k = 1;
  end
end

%------------------------------------------------------------------------
-
% This function interpolates between neighboring tile mappings to produce a
% new mapping in order to remove artificially induced tile borders.
```

```
% Otherwise, these borders would become quite visible.  The resulting
% mapping is applied to the input image thus producing a CLAHE processed
% image.

function claheI = makeClaheImage(I, tileMappings, numTiles, selectedRange,...
                                 numBins, dimTile)

%initialize the output image to zeros (preserve the class of the input image)
claheI = I;
claheI(:) = 0;

%compute the LUT for looking up original image values in the tile mappings,
%which we created earlier
if ~(isa(I,'double') || isa(I,'single'))
  k = selectedRange(1)+1 : selectedRange(2)+1;
  aLut = zeros(length(k),1);
  aLut(k) = (k-1)-selectedRange(1);
  aLut = aLut/(selectedRange(2)-selectedRange(1));
else
  % remap from 0..1 to 0..numBins-1
  if numBins ~= 1
    binStep = 1/(numBins-1);
    start = ceil(selectedRange(1)/binStep);
    stop  = floor(selectedRange(2)/binStep);
    k = start+1:stop+1;
    aLut(k) = 0:1/(length(k)-1):1;
  else
    aLut(1) = 0; %in case someone specifies numBins = 1, which is just silly
  end
end

imgTileRow=1;
for k=1:numTiles(1)+1
  if k == 1  %special case: top row
    imgTileNumRows = dimTile(1)/2; %always divisible by 2 because of padding
    mapTileRows = [1 1];
  else
    if k == numTiles(1)+1 %special case: bottom row
      imgTileNumRows = dimTile(1)/2;
      mapTileRows = [numTiles(1) numTiles(1)];
    else %default values
      imgTileNumRows = dimTile(1);
      mapTileRows = [k-1, k]; %[upperRow lowerRow]
    end
  end

  % loop over columns of the tileMappings cell array
  imgTileCol=1;
  for l=1:numTiles(2)+1
    if l == 1 %special case: left column
      imgTileNumCols = dimTile(2)/2;
      mapTileCols = [1, 1];
    else
      if l == numTiles(2)+1 % special case: right column
        imgTileNumCols = dimTile(2)/2;
```

```
        mapTileCols = [numTiles(2), numTiles(2)];
      else %default values
        imgTileNumCols = dimTile(2);
        mapTileCols = [1-1, 1]; % right left
      end
    end

    % Extract four tile mappings
    ulMapTile = tileMappings{mapTileRows(1), mapTileCols(1)};
    urMapTile = tileMappings{mapTileRows(1), mapTileCols(2)};
    blMapTile = tileMappings{mapTileRows(2), mapTileCols(1)};
    brMapTile = tileMappings{mapTileRows(2), mapTileCols(2)};

    % Calculate the new greylevel assignments of pixels
    % within a submatrix of the image specified by imgTileIdx. This
    % is done by a bilinear interpolation between four different mappings
    % in order to eliminate boundary artifacts.

    normFactor = imgTileNumRows*imgTileNumCols; %normalization factor
    imgTileIdx = {imgTileRow:imgTileRow+imgTileNumRows-1, ...
                  imgTileCol:imgTileCol+imgTileNumCols-1};

    imgPixVals = grayxform(I(imgTileIdx{1},imgTileIdx{2}), aLut);

    % calculate the weights used for linear interpolation between the
    % four mappings
    rowW = repmat((0:imgTileNumRows-1)',1,imgTileNumCols);
    colW = repmat(0:imgTileNumCols-1,imgTileNumRows,1);
    rowRevW = repmat((imgTileNumRows:-1:1)',1,imgTileNumCols);
    colRevW = repmat(imgTileNumCols:-1:1,imgTileNumRows,1);

    claheI(imgTileIdx{1}, imgTileIdx{2}) = ...
        (rowRevW .* (colRevW .* double(grayxform(imgPixVals,ulMapTile)) + ...
                     colW    .* double(grayxform(imgPixVals,urMapTile)))+ ...
         rowW    .* (colRevW .* double(grayxform(imgPixVals,blMapTile)) + ...
                     colW    .* double(grayxform(imgPixVals,brMapTile))))...
        /normFactor;

    imgTileCol = imgTileCol + imgTileNumCols;
  end %over tile cols
  imgTileRow = imgTileRow + imgTileNumRows;
end %over tile rows

%---------------------------------------------------------------------------
-

function [I, selectedRange, fullRange, numTiles, dimTile, clipLimit,...
          numBins, noPadRect, distribution, alpha, ...
          int16ClassChange] = parseInputs(varargin)

narginchk(1,13);

I = varargin{1};
```

```
validateattributes(I, {'uint8', 'uint16', 'double', 'int16', 'single'}, ...
              {'real', '2d', 'nonsparse', 'nonempty'}, ...
              mfilename, 'I', 1);

% convert int16 to uint16
if isa(I,'int16')
  I = int16touint16(I);
  int16ClassChange = true;
else
  int16ClassChange = false;
end

if any(size(I) < 2)
  error(message('images:fus:inputImageTooSmall'))
end

%Other options
%%%%%%%%%%%%%%%

%Set the defaults
distribution = 'uniform';
alpha    = 0.4;

if isa(I, 'double') || isa(I,'single')
  fullRange = [0 1];
else
  fullRange(1) = I(1);          %copy class of the input image
  fullRange(1:2) = [-Inf Inf]; %will be clipped to min and max
  fullRange = double(fullRange);
end

selectedRange    = fullRange;

%Set the default to 256 bins regardless of the data type;
%the user can override this value at any time
numBins =128;
normClipLimit = 0.01;
numTiles = [8 8];

checkAlpha = false;

validStrings = {'NumTiles','ClipLimit','NBins','Distribution',...
                'Alpha','Range'};

if nargin > 1
  done = false;

  idx = 2;
  while ~done
    input = varargin{idx};
    inputStr = validatestring(input, validStrings,mfilename,'PARAM',idx);

    idx = idx+1; %advance index to point to the VAL portion of the input
```

```
    if idx > nargin
      error(message('images:fus:missingValue', inputStr))
    end

    switch inputStr

     case 'NumTiles'
       numTiles = varargin{idx};
       validateattributes(numTiles, {'double'}, {'real', 'vector', ...
                           'integer', 'finite','positive'},...
                    mfilename, inputStr, idx);

       if (any(size(numTiles) ~= [1,2]))
         error(message('images:fus:invalidNumTilesVector', inputStr))
       end

       if any(numTiles < 2)
         error(message('images:fus:invalidNumTilesValue', inputStr))
       end

     case 'ClipLimit'
      normClipLimit = varargin{idx};
      validateattributes(normClipLimit, {'double'}, ...
                   {'scalar','real','nonnegative'},...
                   mfilename, inputStr, idx);

      if normClipLimit > 1
        error(message('images:fus:invalidClipLimit', inputStr))
      end

     case 'NBins'
      numBins = varargin{idx};
      validateattributes(numBins, {'double'}, {'scalar','real','integer',...
                          'positive'}, mfilename, 'NBins', idx);

     case 'Distribution'
      validDist = {'rayleigh','exponential','uniform'};
      distribution = validatestring(varargin{idx}, validDist, mfilename,...
                                    'Distribution', idx);

     case 'Alpha'
      alpha = varargin{idx};
      validateattributes(alpha, {'double'},{'scalar','real',...
                          'nonnan','positive','finite'},...
                   mfilename, 'Alpha',idx);
      checkAlpha = true;

     case 'Range'
      validRangeStrings = {'original','full'};
      rangeStr = validatestring(varargin{idx},
validRangeStrings,mfilename,...
                                'Range',idx);
```

```
    if idx > nargin
      error(message('images:fus:missingValue', inputStr))
    end

    switch inputStr

     case 'NumTiles'
       numTiles = varargin{idx};
       validateattributes(numTiles, {'double'}, {'real', 'vector', ...
                           'integer', 'finite','positive'},...
                    mfilename, inputStr, idx);

       if (any(size(numTiles) ~= [1,2]))
         error(message('images:fus:invalidNumTilesVector', inputStr))
       end

       if any(numTiles < 2)
         error(message('images:fus:invalidNumTilesValue', inputStr))
       end

     case 'ClipLimit'
      normClipLimit = varargin{idx};
      validateattributes(normClipLimit, {'double'}, ...
                   {'scalar','real','nonnegative'},...
                   mfilename, inputStr, idx);

      if normClipLimit > 1
        error(message('images:fus:invalidClipLimit', inputStr))
      end

     case 'NBins'
      numBins = varargin{idx};
      validateattributes(numBins, {'double'}, {'scalar','real','integer',...
                          'positive'}, mfilename, 'NBins', idx);

     case 'Distribution'
      validDist = {'rayleigh','exponential','uniform'};
      distribution = validatestring(varargin{idx}, validDist, mfilename,...
                                    'Distribution', idx);

     case 'Alpha'
      alpha = varargin{idx};
      validateattributes(alpha, {'double'},{'scalar','real',...
                          'nonnan','positive','finite'},...
                   mfilename, 'Alpha',idx);
      checkAlpha = true;

     case 'Range'
      validRangeStrings = {'original','full'};
      rangeStr = validatestring(varargin{idx},
validRangeStrings,mfilename,...
                                'Range',idx);
```

```
      if strmatch(rangeStr,'original')
        selectedRange = double([min(I(:)), max(I(:))]);
      end

     otherwise
      error(message('images:fus:inputString')) %should never get here
    end

    if idx >= nargin
      done = true;
    end

    idx=idx+1;
  end
end


%% Pre-process the inputs
%%%%%%%%%%%%%%%%%%%%%%%%%%

dimI = size(I);
dimTile = dimI ./ numTiles;

%check if tile size is reasonable
if any(dimTile < 1)
  error(message('images:fus:inputImageTooSmallToSplit', num2str( numTiles )))
end

if checkAlpha
  if strcmp(distribution,'uniform')
    error(message('images:fus:alphaShouldNotBeSpecified', distribution))
  end
end

%check if the image needs to be padded; pad if necessary;
%padding occurs if any dimension of a single tile is an odd number
%and/or when image dimensions are not divisible by the selected
%number of tiles
rowDiv  = mod(dimI(1),numTiles(1)) == 0;
colDiv  = mod(dimI(2),numTiles(2)) == 0;

if rowDiv && colDiv
  rowEven = mod(dimTile(1),2) == 0;
  colEven = mod(dimTile(2),2) == 0;
end

noPadRect = [];
if  ~(rowDiv && colDiv && rowEven && colEven)
  padRow = 0;
  padCol = 0;

  if ~rowDiv
```

```
    rowTileDim = floor(dimI(1)/numTiles(1)) + 1;
    padRow = rowTileDim*numTiles(1) - dimI(1);
  else
    rowTileDim = dimI(1)/numTiles(1);
  end

  if ~colDiv
    colTileDim = floor(dimI(2)/numTiles(2)) + 1;
    padCol = colTileDim*numTiles(2) - dimI(2);
  else
    colTileDim = dimI(2)/numTiles(2);
  end

  %check if tile dimensions are even numbers
  rowEven = mod(rowTileDim,2) == 0;
  colEven = mod(colTileDim,2) == 0;

  if ~rowEven
    padRow = padRow+numTiles(1);
  end

  if ~colEven
    padCol = padCol+numTiles(2);
  end

  padRowPre  = floor(padRow/2);
  padRowPost = ceil(padRow/2);
  padColPre  = floor(padCol/2);
  padColPost = ceil(padCol/2);

  I = padarray(I,[padRowPre  padColPre ],'symmetric','pre');
  I = padarray(I,[padRowPost padColPost],'symmetric','post');

  %UL corner (Row, Col), LR corner (Row, Col)
  noPadRect.ulRow = padRowPre+1;
  noPadRect.ulCol = padColPre+1;
  noPadRect.lrRow = padRowPre+dimI(1);
  noPadRect.lrCol = padColPre+dimI(2);
end

%redefine this variable to include the padding
dimI = size(I);

%size of the single tile
dimTile = dimI ./ numTiles;

%compute actual clip limit from the normalized value entered by the user
%maximum value of normClipLimit=1 results in standard AHE, i.e. no clipping;
%the minimum value minClipLimit would uniformly distribute the image pixels
%across the entire histogram, which would result in the lowest possible
%contrast value
numPixInTile = prod(dimTile);
minClipLimit = ceil(numPixInTile/numBins);
clipLimit = minClipLimit + round(normClipLimit*(numPixInTile-minClipLimit));
```

Nntraining

```
rng('default');

load('Train_feat.mat');
load('target.mat');
net = feedforwardnet([5 3]);
inputs = Train_feat';
targets = target';


% Create a Pattern Recognition Network
hiddenLayerSize = 39;
net = patternnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 80/100;
net.divideParam.testRatio = 20/100;

% For help on training function 'trainscg' type: help trainscg
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainscg';  % Scaled conjugate gradient

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};

net.efficiency.memoryReduction = 100;
net.trainParam.max_fail = 6;
net.trainParam.min_grad=1e-5;
net.trainParam.show=10;
net.trainParam.lr=0.9;
net.trainParam.epochs=1000;
net.trainParam.goal=0.00;


% Train the Network
[net,tr] = train(net,inputs,targets);
```

```
% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)

% View the Network
view(net)


disp('after training')
y1 = sim(net,inputs);
y1=abs(y1);
y1=round(y1);

save  net net;

%%%%%%
```

Training.m

```
for yi = 1:100
       yi
%% To read a image from file...
input_image = imread(['DATASETS\',num2str(yi),'.tif']);
figure(1);
imshow(input_image);
title('Input image');

%%  Image Resize
input_image_res=imresize(input_image,[256 256]);
figure(2);
imshow(input_image_res);
title('Image Resize');

%% Fuzzy based Image Enhancement
A=input_image_res;
tic
%Specify the bin range[0 255]
bin=255;
%Find the histogram of the image.
Val=reshape(A,[],1);
Val=double(Val);
I=hist(Val,0:bin);
%Divide the result by number of pixels
Output=I/numel(A);
%Calculate the Cumlative sum
CSum=cumsum(Output);
%Perform the transformation S=T(R) where S and R in the range [ 0 1]
HIm=CSum(A+1);
%Convert the image into uint8
HIm=uint8(HIm*bin);
I=input_image_res;
disp('Parameters for  Fuzzy Enhanced Image')
tic
K1=I(:,:,1);
II=I;
figure(3),imshow(I);
J1=I(:,:,1);
K1 = Fuzzy_opt(K1);
J3 = Fuzzy_opt(K1);
toc;
figure(4);
imshow(J3)
title(' Fuzzy Enhanced Image')
seg_img=J3;
%% Feature extraction (GLCM+DWT)
[cA,cH,cV,cD] = dwt2(seg_img,'haar');
cA = mean2(cA);
cH = mean2(cH);
cV = mean2(cV);
cD = mean2(cD);

Mean = mean2(seg_img);
```

```
Standard_Deviation = std2(seg_img);
Entropy = entropy(seg_img);
Variance = mean2(var(double(seg_img)));
a = sum(double(seg_img(:)));
Smoothness = 1-(1/(1+a));
Kurtosis = kurtosis(double(seg_img(:)));
Skewness = skewness(double(seg_img(:)));
patients_with_vomiting=10.5;
patients_without_vomiting=11.5;
% Inverse Difference Movement
m = size(seg_img,1);
n = size(seg_img,2);
in_diff = 0;
for i = 1:m
    for j = 1:n
        temp = seg_img(i,j)./(1+(i-j).^2);
        in_diff = in_diff+temp;
    end
end
IDM = double(in_diff);
feat_ext = [Mean, Standard_Deviation, Entropy,Variance, Smoothness, Kurtosis,
Skewness, IDM, cA, cH,cV,cD];
Train_feat(yi,:) = feat_ext;
end
save Train_feat Train_feat
%%
```

Roc_curve.m

```
% ------------------------------------------------------------------------ %
%                          R O C    C U R V E                              %
% ------------------------------------------------------------------------ %
% Function "roc_curve" calculates the Receiver Operating Characteristic   %
% curve, which represents the 1-specificity and sensitivity, of two classes
% of data, called class_1 and class_2.                                     %
%                                                                          %
%   Input parameters                                                       %
%       * class_1:  Column vector that represents the data of the first    %
%                   class.                                                 %
%       * class_2:  Column vector that represents the data of the second   %
%                   class.                                                 %
%       * dispp:    (Optional) If dispp is 1, the ROC Curve will be disp- %
%                   ayed inside the active figure. If dispp is 0, no figure
%                   will be displayed.                                     %
%       * dispt:    (Optional) If dispt is 1, the optimum threshold para- %
%                   meters obtained will be displayed on the MATLAB log.   %
%                   Otherwise, if dispt is 0, no parameters will be disp- %
%                   ayed there.                                            %
%                                                                          %
%   Output variables                                                       %
%       * ROC_data: Struct that contains all the curve parameters.         %
%           - param:    Struct that contains the cuantitative parameters   %
%                       of the obtained curve, which are:                  %
%               + Threshold:Optimum threshold calculated in order to maxi-%
%                           mice the sensitivity and specificity values,   %
%                           which is colocated in the nearest point to     %
%                           (0,1).                                         %
%               + AROC:     Area under ROC curve.                          %
%               + Accuracy: Accuracy.                                      %
%               + Sensi:    Sensitivity (i.e., recall, hit rate, or true   %
%                           positive rate).                                %
%               + Speci:    Specificity (i.e., selectivity, or true        %
%                           negative rate).                                %
%               + PPV:      Positive predicted value (i.e., precision).    %
%               + NPV:      Negative predicted value.                      %
%               + FNR:      False negative rate (i.e., miss rate).         %
%               + FPR:      False positive rate (i.e., fall-out).          %
%               + FDR:      False discovery rate.                          %
%               + FOR:      False omission rate.                           %
%               + F1_score: F1 score (harmonic mean of precision and       %
%                           sensitivity).                                  %
%               + MCC:      Matthews correlation coefficient.              %
%               + BM:       Bookmaker informedness.                        %
%               + MK:       Markedness.                                    %
%           - curve:    Matrix that contains the specificity and specifi- %
%                       city of each threshold point in columns.           %
% ------------------------------------------------------------------------ %
%   Example of use:                                                        %
%       class_1 = 0.5*randn(100,1);                                        %
%       class_2 = 0.5+0.5*randn(100,1);                                    %
%       roc_curve(class_1, class_2);                                       %
% ------------------------------------------------------------------------ %
```

```
%   Version log:                                                            %
%       - 1.0: Original script (02/09/2015).                                %
%       - 2.0: Different sizes of data_1 and data_2 are allowed as long as%
%              one column is filled with NaNs (26/06/2018).                 %
%       - 2.1: Classes are now indicated separately (10/07/2018).           %
%       - 3.0: More parameters are indicated as an output (13/12/2018).     %
%       - 3.1: All parameters are now displayed in CMD (13/12/2018).        %
% ------------------------------------------------------------------------- %
                      %
%                                                                           %
%   Sources: Fawcett (2006), Powers (2011), and Ting (2011)                 %
% ------------------------------------------------------------------------- %
function ROC_data = roc_curve(class_1, class_2, dispp, dispt)

    % Setting default parameters and detecting errors
    if(nargin<4), dispt = 1;    end
    if(nargin<3), dispp = 1;    end
    if(nargin<2), error('Params "class_1" or "class_2" are not indicated.');
end
    class_1 = class_1(:);
    class_2 = class_2(:);

    % Calculating the threshold values between the data points
    s_data = unique(sort([class_1; class_2]));          % Sorted data points
    s_data(isnan(s_data)) = [];                 % Delete NaN values
    d_data = diff(s_data);                      % Difference between
consecutive points
    if(isempty(d_data)), error('Both class data are the same!'); end
    d_data(length(d_data)+1,1) = d_data(length(d_data));% Last point
    thres(1,1) = s_data(1) - d_data(1);                 % First point
    thres(2:length(s_data)+1,1) = s_data + d_data./2;   % Threshold values

    % Calculating the sensibility and specificity of each threshold
    curve = zeros(size(thres,1),2);
    distance = zeros(size(thres,1),1);
    for id_t = 1:1:length(thres)
        TP = length(find(class_2 >= thres(id_t)));    % True positives
        FP = length(find(class_1 >= thres(id_t)));    % False positives
        FN = length(find(class_2 < thres(id_t)));     % False negatives
        TN = length(find(class_1 < thres(id_t)));     % True negatives

        curve(id_t,1) = TP/(TP + FN);   % Sensitivity
        curve(id_t,2) = TN/(TN + FP);   % Specificity

        % Distance between each point and the optimum point (0,1)
        distance(id_t)= sqrt((1-curve(id_t,1))^2+(curve(id_t,2)-1)^2);
    end

    % Optimum threshold and parameters
    [~, opt] = min(distance);
    TP = length(find(class_2 >= thres(opt)));    % No. true positives
    FP = length(find(class_1 >= thres(opt)));    % No. false positives
    FN = length(find(class_2 < thres(opt)));     % No. false negatives
    TN = length(find(class_1 < thres(opt)));     % No. true negatives
```

```
    % Output parameters
    param.Threshold = thres(opt);               % Optimum threshold position
    param.Sensi = curve(opt,1);                 % Sensitivity
    param.Speci = curve(opt,2);                 % Specificity
    param.AROC  = abs(trapz(1-curve(:,2),curve(:,1))); % Area under curve
    param.Accuracy = (TP+TN)/(TP+TN+FP+FN);     % Aaccuracy
    param.PPV   = TP/(TP+FP);                   % Positive predictive value
    param.NPV   = TN/(TN+FN);                   % Negative predictive value
    param.FNR   = FN/(FN+TP);                   % False negative rate
    param.FPR   = FP/(FP+TN);                   % False positive rate
    param.FDR   = FP/(FP+TP);                   % False discovery rate
    param.FOR   = FN/(FN+TN);                   % False omission rate
    param.F1_score = 2*TP/(2*TP+FP+FN);         % F1 score
    param.MCC   = (TP*TN-FP*FN)/sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN));  %
Matthews correlation coefficient
    param.BM    = param.Sensi+param.Speci-1;    % Informedness
    param.MK    = param.PPV+param.NPV-1;        % Markedness

    param.TP = TP;    % No. true positives
    param.FP = FP;    % No. false positives
    param.FN = FN;    % No. false negatives
    param.TN = TN;    % No. true negatives

    % Plotting if required
    if(dispp == 1)
        fill_color = [11/255, 208/255, 217/255];
        fill([1-curve(:,2); 1], [curve(:,1); 0], fill_color,'FaceAlpha',0.5);
        hold on; plot(1-curve(:,2), curve(:,1), '-b', 'LineWidth', 2);
        hold on; plot(1-curve(opt,2), curve(opt,1), 'or', 'MarkerSize', 10);
        hold on; plot(1-curve(opt,2), curve(opt,1), 'xr', 'MarkerSize', 12);
        hold off; axis square; grid on; xlabel('1 - specificity');
ylabel('sensibility');
        title(['AROC = ' num2str(param.AROC)]);
    end

    % AROC warning
    if param.AROC < 0.5
        warning('Since AROC is less than 0.5, you should swap the classes:
roc_curve(class_2,class_1).');
    end

    % Log screen parameters if required
    if(dispt == 1)
        fprintf('\n ROC CURVE PARAMETERS\n');
        fprintf(' ----------------------------\n');
        fprintf('  - Distance:     %.4f\n', distance(opt));
        fprintf('  - Threshold:    %.4f\n', param.Threshold);
        fprintf('  - Sensitivity:  %.4f\n', param.Sensi);
        fprintf('  - Specificity:  %.4f\n', param.Speci);
        fprintf('  - AROC:         %.4f\n', param.AROC);
        fprintf('  - Accuracy:     %.4f\n', param.Accuracy);
        fprintf('  - PPV:          %.4f\n', param.PPV);
        fprintf('  - NPV:          %.4f\n', param.NPV);
        fprintf('  - FNR:          %.4f\n', param.FNR);
```

```
        fprintf('   - FPR:           %.4f\n', param.FPR);
        fprintf('   - FDR:           %.4f\n', param.FDR);
        fprintf('   - FOR:           %.4f\n', param.FOR);
        fprintf('   - F1 score:      %.4f\n', param.F1_score);
        fprintf('   - MCC:           %.4f\n', param.MCC);
        fprintf('   - BM:            %.4f\n', param.BM);
        fprintf('   - MK:            %.4f\n', param.MK);
        fprintf(' \n');
    end

    % Assinging parameters and curve data
    ROC_data.param = param;
    ROC_data.curve = curve;
end
```

REFERENCES

[1.1] D. Maltoni, D. Maio, K. Jain, S. Prabhakar, Handbook of Fingerprint Recognition, Springer. British Library of Cataloguing in Publication Data, ISBN: 978-1-84882-253-5, 2009.

[1.2] J. BO, T. Hua, P. Xing LAN, Fingerprint Singular Point Detection Algorithm by Poincare Index, la collection des ingenieursd'Electronique et d'informatique, CHINA, 2008.

[1.3] M. KAWAGOE, A. TOKO, Fingerprint Pattern Classification. PatternRecognition, vol.17, pp. 295-303, 1984.

[1.4] Cover, T.M., Hart, P.E, Nearestneighbor pattern classification, Institute of Electrical and Electronics Engineers Transactions on Information Theory,13, 21-27 (1967).

[1.5] K.-B. Duan, S.S Keerthi, Which is the best multiclass SVM method? An empirical study. Technical Report CD-0312, Control Division, Department of Mechanical Engineering, National University of Singapore, 2003.

[1.6] Jelena Godjevac, Apprentissage des neurones flous.

[1.7] Leila Boutella, Amina Serir, Farida Benabbas et Zehira Haddad, Classification des images empreintes digitales par la transformée en Ridgelets par blocs.

[1.8] Dr.H.B.Kekre, Dr.SudeepThepade, Dimple Parekh, -Fingerprint classification using KFCG Algorithm, International Journal of Computer Scienceand Information Security(IJCSIS), Vol 9, 2012.

[1.9] Dimple Parekh, RekhaVig 2011 Review of Fingerprint Classification methods based on Algorithmic Flow Journal of biometrics, Vol 2, Issue1, 2011.

[1.10] N. Yager and A. Amin, Fingerprint classification: A review,PatternAnal. Applic.7.1 (2004), 77-93. M. Tico, P. Kuosmanen, Neuro-Fuzzy Modelling and Control, Proceedings of the IEEE, Vol.83, No 3, 1995, pp.378-405.

[1.11] M. Tico, P. Kuosmanen, Neuro-Fuzzy Modeling and Control, Proceedings of the IEEE, Vol.83, No 3, 1995, pp. 378-405.

[1.12] V.Vapnik,the nature of statistical learning, theory.Spring-Verlag, NewYork, USA, 1995.

[1.13] O. Lezoray. Segmentation d'images par morphologie mathématique et classification de donnes par réseaux de neurones: Application la classification de cellules en cytologie des séreuses. Thèse de doctorates, Universitéde Caen 2000.

[1.14] ROSENBLATT, F. The Perceptron - A theory of Statistical Separabilityin Cognitive Systems, Cornell Aeronautical Laboratory Report VG-1198-G-1, January, 1958.

[1.15] J.-S.R. Jang, ANFIS: Adaptive-Network-Based Fuzzy Inference Systems, IEEE Trans. Systems, Man& Cybernetics 23 (1993), pp: 665685.

[1.16] S. U. Maheswari, E. Chandra, A Review Study on Fingerprint Classification Algorithm used for Fingerprint Identification and Recognition, International Journal of Computer Science and Technology IJCST Vol.Issue 1, JanMarch 2012.
[1.17] M. Hanmandlu, S. N. Tandon, and A. H. Mir, "A new fuzzy logic based image enhancement," Biomed. Sci. Instrum, vol. 34, pp. 590–595.

[1.18] M. Natchegael , E. E. Kerre Fuzzy techniques in image processing , Springer Verlag , 2000.
[11] M. Russo, G. Ramponi , " A fuzzy operator for blurred and noisy images", IEEE Trans. Image Process., vol 4, no.8, pp 1169-1171, Aug 1995.

[1.19] M. T. Yildrim, A. Basturk, " A Detail Preerving type-2 Fuzzy Logic Filter for Impulse Noise Removal from Digital Images", Fuzzy Systems Conference, FUZZ-IEEEE, 2007.

[1.20] M.Hanmandlu, D. Jha, "An Optimal Fuzzy System for Color Image Enhancement" IEEE Transactions on Image Processing, Vol. 15 ,No. 10 p.p 2956-2966,October 2006.

[1.21] M.Hanmandlu, J. See, S. Vasikarla, "Fuzzy Edge Detector Using Entropy Optimization",International Conference on Information Technology: Coding and Computing,2004.

[1.22] N.N. Karnik, J. M. Mendel, "Introduction to Type-2 Fuzzy Logic Systems", 1998 IEEE World Congress on Computational Intelligence, vol.2, p 915-935, pt. 2, 1998.

[1.23] R. Krishnapuram and Y.S. Choi , " A robust approach to image enhancement based on fuzzy logic", IEEE Trans. Image Process., vol 6, no. 6, pp 808-825, Jun 1997.

[2.1] B.G. Sherlock, D. M. Monro, K. Millard, " Fingerprint Enhancement by Directional Fourier Filtering", IEE Proc. Image Signal Process, Vol. 141, No. 2, April 1994.

[2.2] H. R. Tizhoosh, "Image Thresholding using Type-II Fuzzy Sets", Pattern Recognition vol. 38, pp 2363-2372, 2005.

[2.3] J.M. Mendel, R.I. John, "Type-2 Fuzzy Sets made Simple ", IEEE Transactions on Fuzzy Systems, vol. 10, no. 2, April 2002.

[2.4] J. M. Mendel, R. I. John, F. Liu, "Interval Type-II Fuzzy Logic Systems Made Simple", IEEE Transactions on Fuzzy Systems, vol. 14, Dec., 2006.

[2.5] L.A.Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes" IEEE Trans. Sys. Man Cybern. SMC-3, 2914, 1973.

[2.6] L.A. Zadeh, "The Concept of A Linguistic Variable and Its Application to Approximate Reasoning - 1", Information Sciences 8, pp. 199-249, 1975.

[2.7] L.Hong, Y. Wan, A. jain "Fingerprint Image Enhancement: Algorithm and Performance Evaluation" IEEE Transaction on pattern analysis and machine intelligence, Vol 20 No. 8, p.p. 777-789, 1998.

[2.8] M. Hanmandlu, D. Jha, and R. Sharma, "Color image enhancement using fuzzy intensification", Pattern Recognition Letters, Vol. 24(1-3) 2003, pp. 81-87.

[2.9] R. Bansal, P. Sehgal, P. Bedi, "A novel framework for enhancing images corrupted by impulse noise using type-II fuzzy sets", IEEE International Conference on Fuzzy Systems and Knowledge Discovery(FSKD'2008), vol. 3, pp 266-271, October 2008.

[2.10] R. C. Gonzalez, R. E. Wood, "Digital Image Processing", Second Edition, Prentice Hall,2006.

[2.11] S. Greenberg, M. Aladjem, D. Kogan and I. Dimitrov "Fingerprint Image Enhancement using Filtering Techniques" ,Electrical and Computer Engineering Department, BenGurion University of the Negev, Beer-Sheva, Israel, 2000.

[2.12] S. K. Pal and R. A. King, "Image enhancement using smoothing with fuzzy sets," IEEE Trans. Syst. Man. Cybern, vol. SMC-11, no. 7, pp.494–501, Jul. 1981.

[2.13] S.K. Oh, J.J. Lee, C.H. Park, B.S. Kim, K.H. Park, "New Fingerprint Image Enhancement Using Directional Filter Bank", School of Electrical Engineering, Kyungpook National University SEOUL, Daegu, Korea, School of Internet Engineering, Dongseo University SEOUL , Daegu, Korea. 417

[2.14] Gonzalez R. C. and Woods R. E., Digital Image Processing, 3'd ed.Upper Saddle River, NJ: Prentice- Hall, 2006.

[2.15] Yanxia Wang, QiuqiRuan, "An Improved UnsharpMasking method for Palmprint Image Enhancement", Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC'06), 2006 IEEE.

[2.16] Timothy j.Ross, Fuzzy Logic with Engineering Applications, 2'd edition

[2.17] SasiGopalan, Madhu S Nair, and Souriar Sebastian Approximation Studies on "Image Enhancement Using Fuzzy Technique"International Journal of Advanced Science and Technology Vol. 10, September, 2009, pp. 11-26.

[2.18] Reshmalakshmi C, Sasikumar M, "Image Contrast Enhancement using Fuzzy Technique" ,2013 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2013], IEEE, pp. 861-865.

[2.19] Zhou Z, "Cognition and removal of impulse noise with uncertainty". IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 21, NO. 7, JULY 2012, pp. 3157-3167.

[2.20] Hamid R. Tizhoosh, Bernd Michaelis, "Image Enhancement Based on Fuzzy Aggregation techniques", IEEE IMTC'99,vol. 3, 1999 IEEE, pp. 1813-1817.

[2.21] Dr.G.Sudhavani, M.Srilakshmi, S. Sravani, P. Venkateswara Rao K Enhancement of Low Contrast Images Using Fuzzy Techniques SPACES-2015, Dept of ECE, K L university. pp. 286-290.

[2.22] NutanY.Suple, Sudhir M. Kharad, "Basic approach to image contrast enhancement with fuzzy inferencesystem", International Journal of Scientific and Research Publications, Volume 3, Issue 6, June 2013, pp. 1-4.

[2.23] A. A. Altun and N. Allahverdi. Neural network based recognition by using genetic algorithm for feature selection of enhanced fingerprints. In Adaptive and Natural Computing Algorithms, pages 467–476. Springer, 2007.

[2.24] S. Bernard, N. Boujemaa, D. Vitale, and C. Bricot. Fingerprint segmentation using the phase of multiscale gabor wavelets. In The 5th Asian Conference on Computer Vision, Melbourne, Australia. Citeseer, 2002.

[2.25] J. D. Bowen. The home office automatic fingerprint pattern classification project. In Neural Networks for Image Processing Applications, IEE Colloquium on, pages 1/1–1/5, Oct 1992.
[2.26] R. Cappelli, M. Ferrara, and D. Maltoni. Minutia cylinder-code: A new representation and matching technique for fingerprint recognition. IEEE Trans. Pattern Anal. Mach. Intell., pages 2128–2141, 2010.

[3.1] H. Chen and A. F. Murray. Continuous restricted boltzmann machine with an implementable training algorithm. In Vision, Image and Signal Processing, IEE Proceedings, Volume 150, No. 3, 10.1049/ipvis:20030362, 2003.

[3.2] J. Chen, F. Chan, and Y.-S. Moon. Fingerprint matching with minutiae quality score. In Advances in Biometrics, volume 4642 of Lecture Notes in Computer Science. 2007.

[3.3] H. Cheng, J. Tian, and T. Zhang. Fingerprint enhancement with dyadic scale-space. In Pattern Recognition, 2002. Proceedings. 16th International Conference on, volume 1, pages 200–203 vol.1, 2002.

[3.4] K. Chih Lee and S. Prabhakar. Probabilistic orientation field estimation for fingerprint enhancement and verification. In Biometrics Symposium, 2008. BSYM ’08, pages 41–46, Sept 2008.

[3.5] A. Crouzil, L. Massip-Pailhes, and S. Castan. A new correlation criterion based on gradient. fields similarity. In Pattern Recognition, 1996., Proceedings of the 13th International Conference on, volume 1, pages 632– 636 vol.1, Aug 1996.

[3.6] S. Dass. Markov random field models for directional field and singularity extraction in fingerprint images. Image Processing, IEEE Transactions on, 13(10):1358–1367, 2004.

[3.7] G. Desjardins and Y. Bengio. Empirical evaluation of convolutional rbms for vision. DIRO, Universite de ´ Montreal ´ , 2008.

[3.8] A. Erol, U. Halici, and G. Ongun. Intelligent biometric techniques in fingerprint and face recognition. chapter Feature Selective Filtering for Ridge Extraction, pages 195–215. 1999.

[3.9] J. Feng, J. Zhou, and A. Jain. Orientation field estimation for latent fingerprint enhancement. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 35(4):925–940, April 2013.

[3.10] S. Greenberg, M. Aladjem, D. Kogan, and I. Dimitrov. Fingerprint image enhancement using filtering techniques. In Pattern Recognition, 2000. Proceedings. 15th International Conference on, volume 3, pages 322–325. IEEE, 2000.

[3.11] T. Hatano, T. Adachi, S. Shigematsu, H. Morimura, S. Onishi, Y. Okazaki, and H. Kyuragi. A fingerprint verification algorithm using the differential matching rate. In Proceedings of the 16 th International Conference on Pattern Recognition (ICPR’02) Volume 3 - Volume 3, ICPR ’02, 2002.

[3.12] G. E. Hinton. A practical guide to training restricted boltzmann machines. August 2010.

ACKNOWLEDGEMENTS

I would like to sincerely thank my advisor Professor Dr. Franklin Bien, who has helped me enormously deciding a way of research, which has been a wellspring of motivation, his patience support and guidance towards my research was a great encouragement without whom it would have been harder to accomplish the thesis. I thank Professor Dr. Jaesik Choi for helping me in the development of initial ideas during the coursework Advanced Machine Learning and being my guide in the thesis work. I would also like to thank Professor Dr. Jin Ho Chung for accepting my proposal and being a part of my thesis committee member. I thank all my coursework professors in the department of Computer Science specially Professor Dr. Young-ri Choi for the guidance in the field of Cloud Computing and Advanced Operating system. All the knowledge and skills I acquired in the respective fields of Artificial Intelligence and Cloud Computing is due to the assistance of all the respective Professors under the guidance of whom I worked. Also, I thank and admire Professor Changju Oh for having me as teaching assistant (TA) for the subject Artificial Intelligence.

I thank my co-lab members especially Kyeongmin Park with whom I shared ideas related to the fingerprint analysis throughout the master's period and coursework colleagues, with whom I have worked on several projects related to Machine Learning.

Here comes my family without whom I would not have been a part of higher studies once again after long years of working in corporates like IBM and Hewlett Packard, where I worked as Server Engineer in VMware and Wintel for years soon after completing my Bachelor's in Technology (B-Tech from BPUT) India. My Family holds the vital role and stood as a backbone throughout the master's period, they are the ones who showed the path to me in the darkness whenever I used to lose hope. "Thanks" will be a small phrase to describe the unconditional love, care and support I got from my 'Amma and Pinni'. I cherish the unforgettable moments where I got support and guidance from my husband Dr. Sai Kiran, till I completed my Thesis Presentation successfully before time.

I would like to thank everyone in the journey of my Masters in UNIST. South Korea is one of the most beautiful countries I have ever visited which drew my heart with its pleasant and nature's scenic ambience. I admire my advisor Professor Dr. Franklin Bien for giving me an opportunity to be a part of one of the world's top renowned universities UNIST.