Integrated Finger Print Recognition Using Image Morphology and Neural Network

Ravi Lakshmanan
Senior Lecturer, School of Engineering
Asia Pacific University of Technology and Innovation
Kuala Lumpur, Malaysia

Sathish Kumar Selvaperumal
Senior Lecturer, School of Engineering
Asia Pacific University of Technology and Innovation
Kuala Lumpur, Malaysia

Chow Hin Mun
Assistant Engineer, Willowglen Technology Sdn Bhd,
Kuala Lumpur, Malaysia

***Abstract*—Finger print recognition is one among the biometric measures which is commonly used. In this paper, an integrated finger print recognition is proposed using image morphology and neural networks. In the first stage, acquisition of the finger print image is done using finger print reader and then pre-processing is done using 2D Median and Weiner filter, then in the second stage, image Binarization, image normalization, image thinning, and minutiae extraction are done in order to extract the minutiae features. Further these minutiae points are reshaped into 60 feature vectors as post – processing. Further these extracted features are used to configure and train the neural network. The performance of the system developed is evaluated and from the simulated results, it is found that the system achieved an overall FRR of 8% and FAR of 8%. Also, the calculated average time taken to train the neural network is 1.214 seconds only.**

Keywords- Finger print recognition, Median filter, Weiner filter, Image Morphology, Neural Network.

I. INTRODUCTION

Fingerprint is the most commonly used biometric measures for automatic personal identification [1], widely applied in personal identification forensic investigation, bank transaction and access control system. When compared to other biometric measures, human fingerprints have the most uniqueness and invariance [7], and at the same time offer better security and convenience to utilizer [1]. The uniqueness of the fingerprint lies on its characteristic ridge and furrows pattern, which is unique to everyone, even identical twins do not have the same fingerprint pattern [3]. Fingerprint feature extraction algorithm can be classified generally based on minutiae points and correlation-based.

A fingerprint security system applied to encrypt email proposed by Al-Taee et al. [1] is based on minutiae extraction to extract the unique points of the fingerprint. This method is executed using a GrFinger Software Development Kit (SDK), where the entire minutiae extraction algorithm can be easily done in a sequential order of Binarization, thinning and minutiae detection.

Binarization is done to convert the input fingerprint image in the form of grey scale image to binary image, with the purpose of compressing the image, repair broken lines and enhance the image further, so as the subsequent process of thinning can be easily done. Thinning refers to the reduction of the fingerprint ridge pattern thickness into a single line, where the minutiae points can be easily identified. Two types of minutiae points are emphasized, which are ridge termination and ridge bifurcation. Ridge termination marked in red refers to the end point, whereas ridge bifurcation in blue marking refers to the branching point. These minutiae points provide simple yet unique features to represent fingerprint patterns [1].

In a fingerprint recognition system proposed by Zhang et al. [8] using the similar minutiae points feature extraction, with improvements based on Direct Minutia Extraction algorithm, that involves fingerprint image enhancement based on Gabor filtering to remove fake minutiae points and using ridge line tracing algorithm to accurately extract the minutiae points. This method shows an improved recognition performance over traditional minutiae extraction, as justified by the outcomes of simulation test which yield less than 1 second system identifying time and FAR less than 0.95% [8].

A detailed fingerprint feature extraction algorithm based on correlation can be seen in the model proposed by Vasuhi et al. [6]. This methodology is a cross correlation based technique known as Cross Correlation of Field Orientation (CCFO) that combines cross correlation along with the field orientation technique for fingerprint features extraction. Pre-processing of the fingerprint image includes smoothing and edge detection using Gaussian filter and Canny filter respectively. Usually a high frequency image is

difficult to process mainly due to high variation in the pixel intensity. Hence, smoothing is done in which it is converted to a slow varying image thereby reducing the high variation with respect to the spatial coordinates. Edge detection is a

process in which the sharp changes are usually detected due to the variations in intensity. By doing this, the most important changes in an image is captured by obtaining the sharp changes in the same image brightness.

After pre-processing, the image is converted based on Orientation Filed Methodology (OFM) into field orientation image. Field orientation is defined as a technique by which the directional properties of the image are extracted instead of the actual image. This technique has advantages such as high robustness to noise and environmental factors such as moisture and high humidity.

Cross correlation is used as a matching technique to determine the image likeness. The obtained field orientation image in the form of cross correlation coefficient is compared with the images that are stored in the database, based on analysis in the frequency domain using Fourier transform. This methodology achieved 79.67% correct recognition. However, the recognition accuracy decreases with the increase in the size of the database [6].

II. PROPOSED METHOD

The design and implementation of the future extraction algorithm for fingerprint recognition system is done by first writing the source code in MATLAB. Figure 2.1 shows the block diagram of the feature extraction processes for fingerprint recognition system based on fingerprint minutiae points feature extraction algorithm:

Step 1: Fingerprint Acquisition via Fingerprint Reader

Based on Figure 2.1, the processing chain of the fingerprint recognition subsystem begins by first obtaining the fingerprint image of the user via a fingerprint reader and using a Security Development Kit (SDK) known as UniFinger, as a software interface to save the fingerprint image. Using the user interface, the captured fingerprint image can be saved in the computer as a bitmap image file, where subsequent processing can be done by loading the fingerprint image over to MATLAB.

Step 2: Image Enhancement via Filtering

The captured fingerprint image in the form of 360 x 256 bitmap file is exported to the workspace and pre-processing is done to enhance the quality of the image. The fingerprint image will undergo two types of noise removal filters which are 2-D Median filter and 2-D adaptive Wiener filter.

The fingerprint image after applying Median filter contains less dotted points. While the fingerprint image after applying Wiener filter have a smoother edge and better image quality. As a result, the ridge pattern of the fingerprint can be clearly seen and the feature points can be properly extracted in the following processes.

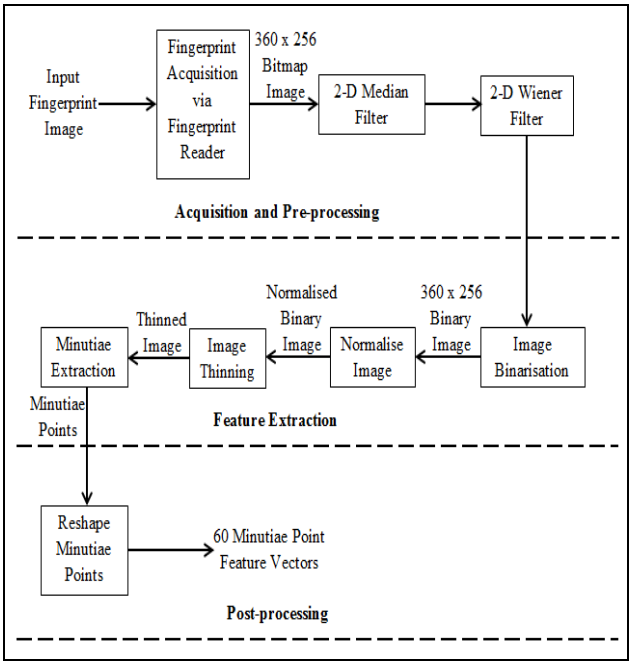

Figure 2.1: Block Diagram of Fingerprint Minutiae Points Feature Extraction Algorithm

Step 3: Image Binarization

Fingerprint feature extraction begins with first converting the pre-processed fingerprint image into binary. The resultant image seen will be in the form of 360 x 256 represented by binary bits of 1 and 0.

Step 4: Image Normalization

The next step in feature extraction is to normalize the dimension of the fingerprint image, to focus on the Region of Interest (ROI). The resultant fingerprint image shown illustrates the ROI of the fingerprint, where the more distinctive ridge pattern of the fingerprint is focused, so that the feature points can be extracted easily and consistently.

Step 5: Image Thinning

Image thinning, also known as skeletonisation is done to thin down the ridge pattern into a single line. Thinning of the fingerprint image shows a single line skeletal view of the fingerprint ridge pattern.

Step 6: Minutiae Extraction

Minutiae extraction algorithm is based on finding the termination and bifurcation points in the ridge pattern of the fingerprint. The resultant extracted minutiae image consists of a combination of red and blue markings. The red markings represent the ridge termination points, while the blue markings represent the ridge bifurcation points.

Step 7: Reshape Minutiae Points

The position of the minutiae points is described by their coordinates in the form of x-axis and y-axis relative to the origin. Hence, these minutiae points will be reshaped into 60 feature vectors. Pythagoras' Theorem is used to determine the exact point of the ridge termination and bifurcation, based on their coordinates on the x-axis and y-axis. Finally, these points are combined into a column vector of 60 feature vectors, which will be used in training the neural network.

Neural Network:

After obtaining and compiling all the extracted features of the finger print, the next procedure is to configure and train them using the neural network toolbox utility in MATLAB. In supervised training of the neural network, the neural network is inputted with the feature vectors, where it will be trained to map the corresponding targets of the inputs. The 100 input data containing the facial feature vectors will be randomly divided into 70% for data training, 15% for data validation and 15% for data testing. Training data constitute the major samples, as they are presented to the network during training, where the value of the weight and bias are adjusted to reduce the error rate in recognizing the pattern. Validation data are used to measure the generalization of the neural network. The training of the neural network will be stopped when generalization stops improving, as the neural network starts to memorize the pattern, instead of learning to recognize. Testing data have no effect on the training, only to provide independent measure on the performance of the network during and after training.

The structure of the neural network is configured as feed forward network and the number of hidden neurons of the hidden layer can also be defined. For the purpose of simulation test and analysis, three different neural networks are configured with 10, 50 and 100 hidden neurons. Table 2.1 shows the number of neuron for each layer of the neural networks for the system:

Table 2.1: Number of Neuron in Each Layer

Recognition system	Input Neuron	Hidden Neuron	Output Neuron
Fingerprint	60	10, 50, 100	5

The number of input neuron will correspond to the number of input facial feature vector. For face recognition system, there are 60 input neurons, and the output layer consists of 5 output neurons which correspond to the number of users enrolled to the system. Hence, the output layer is acting as similar to the database that stores the template of the enrolled users. The transfer function used in these feed forward neural networks is tangent-sigmoid, which is commonly used for pattern recognition. When the neural network is properly configured, training is commenced, with the use of a training algorithm known as scaled conjugate gradient back-propagation. After the training has been done, testing is done.

.

III. EXPERIMENTAL RESULTS

A. Data Base

For fingerprint recognition subsystem, 5 fingerprint samples taken are collected from 5 users, summing up to a total of 25 fingerprint samples, where these fingerprint samples acting as the fingerprint biometric data will undergo minutiae points feature extraction. The extracted minutiae point feature vectors of these fingerprint samples will be used to train the neural network. Figure 3.1 shows the fingerprint images collected from each of the user:

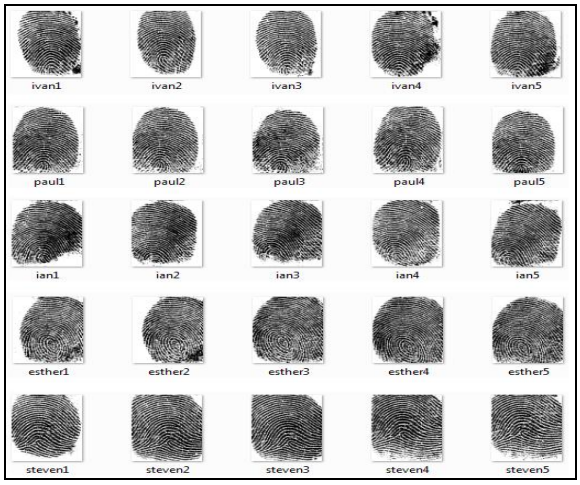

Figure 3.1: Fingerprint Samples for Each User

B. Graphical User Interface

A Graphical User Interface (GUI) is designed to replace the MATLAB command line interaction when operating the system. The GUI maintains the same function and processing flow, but it is designed to provide user-friendly interaction when using the system. The layout of the GUI is design using GUI Development Environment (GUIDE) in MATLAB and a script file is generated containing the source code to implement all the recognition function into the GUI. Figure 3.2 shows the GUI for the

finger print recognition system. Figure 3.3 shows the captured fingerprint image in the UniFinger SDK user interface. Figure 3.3 illustrates the fingerprint images when undergoing the fingerprint feature extraction processes and Figure 3.4 shows the finger print feature extraction process.

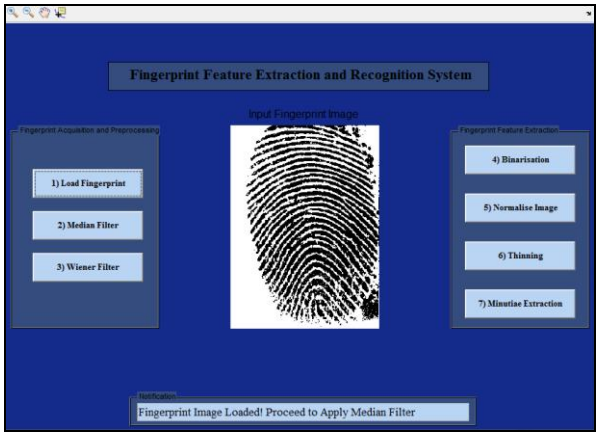

Figure 3.2: GUI for Finger Print Recognition system

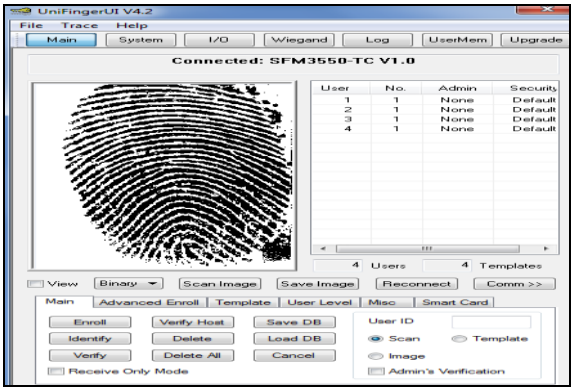

Figure 3.3: UniFinger SDK User Interface

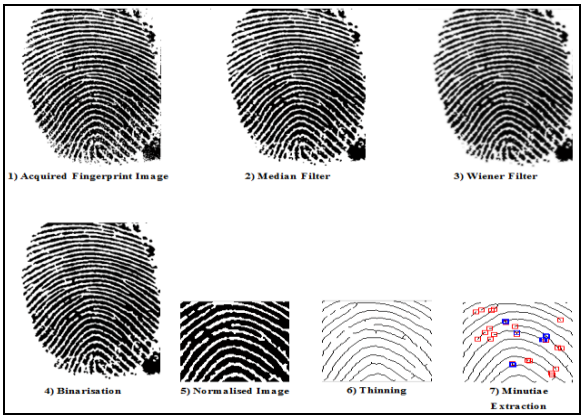

Figure 3.4: Fingerprint Images for Each Stage of Fingerprint Feature Extraction

Figure 3.5 shows the neural network configuration for finger print recognition system and the summary of the neural network training process for finger print recognition system is shown in the Figure 3.6.

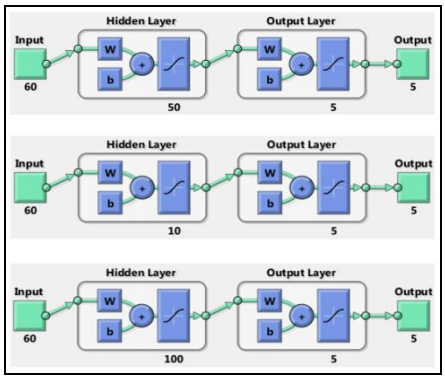

Figure 3.5: Neural Network Configuration for finger print recognition

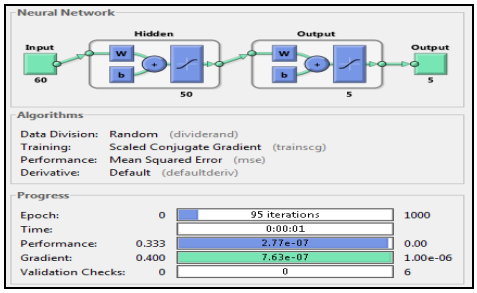

Figure 3.6: Finger Print Neural Network Training Summary

C. Finger Print Simulation Scenario 1:10 hiddenNeurons

Figure 3.7 shows the line plot on training performance of the neural network, with respect to the MSE value and Figure 3.8 shows the MSE values and percentage errors for each type of the training data. Figure 3.9 shows the confusion matrix.

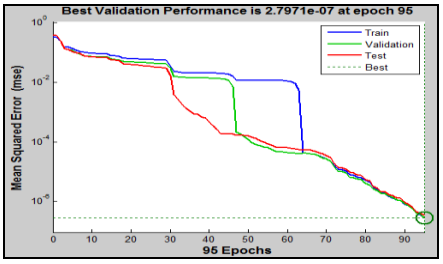

Figure 3.7: Training Performance for Fingerprint Simulation Scenario 1

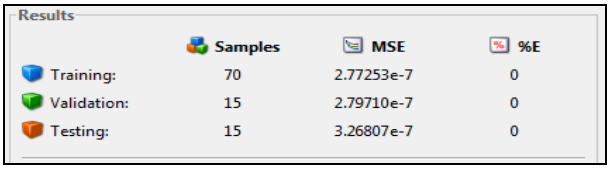

Figure 3.8: MSE Values and Percentage Errors for Fingerprint Simulation Scenario 1

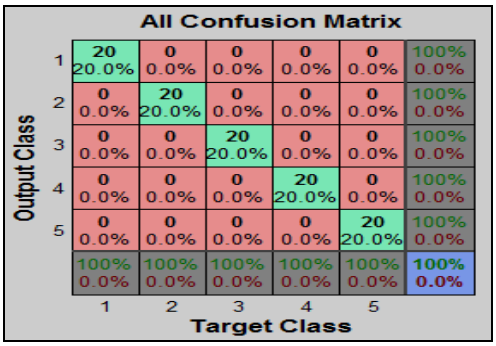

Figure 3.9: Confusion Matrix for Fingerprint Simulation Scenario 1

Table 3.1(a): FRRTest for Fingerprint Simulation Scenario1

	Raw Test Fingerprint Images									
Users	*I1*	*I2*	*I3*	*I4*	*I5*	*P1*	*P2*	*P3*	*P4*	*P5*
Ivan	1	1	1	1	1	0	0	0	0	0
Paul	0	0	0	0	0	1	1	1	1	1
Lan	0	0	0	0	0	0	0	0	0	0
Esther	0	0	0	0	0	0	0	0	0	0
Steven	0	0	0	0	0	0	0	0	0	0
FR	N	N	N	N	N	N	N	N	N	N

Table 3.1(b): FRRTest for Fingerprint Simulation Scenario1

	Raw Test Fingerprint Images									
Users	*L 1*	*L 2*	*L 3*	*L4*	*L5*	*E1*	*E2*	*E3*	*E4*	*E5*
Ivan	0	0	0	0	0	0	0	0	0	0
Paul	0	0	0	0	0	0	0	0	0	0
Lan	1	1	1	1	0	0	0	0	0	0
Esther	0	0	0	0	1	1	1	1	0	1
Steven	0	0	0	0	0	0	0	0	1	0
FR	N	N	N	N	**Y**	N	N	N	**Y**	N

Table 3.1(c): FRRTest for Fingerprint Simulation Scenario1

Based on the simulation results shown in Table 3.1, the FFR is calculated as shown in the following:

$$False\ Rejection\ Rate\ (FRR) = \frac{Number\ of\ False\ Rejection}{Total\ Number\ of\ Test} \times 100\%$$

$$False\ Rejection\ Rate\ (FRR) = \frac{2}{5\ users \times 5\ tests} \times 100\%$$

$$False\ Rejection\ Rate\ (FRR) = 8\%$$

Table 3.2 shows the FAR test on the fingerprint recognition system when each user is asked to access the system as the other users and scan their fingerprint images for authentication:

Table 3.2(a): FARTest for Fingerprint Simulation Scenario1

	Raw Test Fingerprint Images									
Users	*I1*	*I2*	*I3*	*I4*	*I5*	*P1*	*P2*	*P3*	*P4*	*P5*
Ivan	x	x	x	x	x	0	0	0	0	0
Paul	0	0	0	0	0	x	x	x	x	x
Lan	0	0	0	0	0	0	0	0	0	0
Esther	0	0	0	0	0	0	0	0	0	0
Steven	0	0	0	0	0	0	0	0	0	0
FA	N	N	N	N	N	N	N	N	N	N

Table 3.2(b): FARTest for Fingerprint Simulation Scenario1

	Raw Test Fingerprint Images									
Users	*L 1*	*L 2*	*L 3*	*L4*	*L5*	*E1*	*E2*	*E3*	*E4*	*E5*
Ivan	0	0	0	0	1	0	0	0	0	0
Paul	0	0	0	0	0	0	0	0	0	0
Lan	x	x	x	x	x	0	0	0	0	0
Esther	0	0	0	0	1	x	x	x	x	x
Steven	0	0	0	0	0	0	0	0	1	0
FA	N	N	N	N	**Y**	N	N	N	N	N

Table 3.2(c): FARTest for Fingerprint Simulation Scenario1

	Raw Test Fingerprint Images				
Users	*S1*	*S2*	*S3*	*S4*	*S5*
Ivan	0	0	0	0	0
Paul	0	0	0	0	0
Lan	0	0	0	0	0
Esther	0	0	0	0	0
Steven	X	x	x	x	x
FA	N	N	N	N	N

Based on the simulation results shown in Table 3.2, the FAR is calculated as shown in the following:

$$False\ Acceptance\ Rate\ (FAR) = \frac{Number\ of\ False\ Acceptance}{Total\ Number\ of\ Test} \times 100\%$$

$$False\ Acceptance\ Rate\ (FAR) = \frac{1}{5\ users \times 5\ tests} \times 100\%$$

$$False\ Acceptance\ Rate\ (FAR) = 4\%$$

	Raw Test Fingerprint Images				
Users	*S1*	*S2*	*S3*	*S4*	*S5*
Ivan	0	0	0	0	0
Paul	0	0	0	0	0
Lan	0	0	0	0	0
Esther	0	0	0	0	0
Steven	1	1	1	1	1
FR	N	N	N	N	N

D. Finger Print Simulation Scenario 2:50 hiddenNeurons

Figure 3.10 shows the line plot on training performance of the neural network, with respect to the MSE value and Figure 3.11 shows the MSE values and percentage errors for each type of the training data and Figure 3.12 shows the all confusion matrix

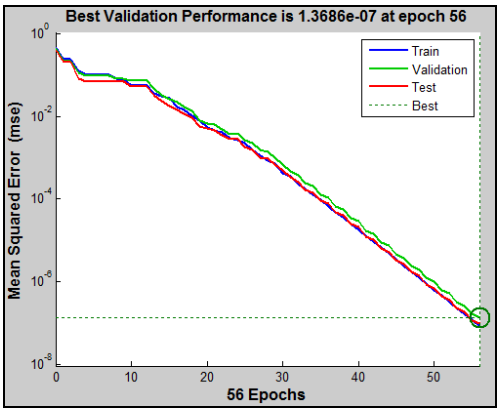

Figure 3.10: Training Performance for Fingerprint Simulation Scenario 2

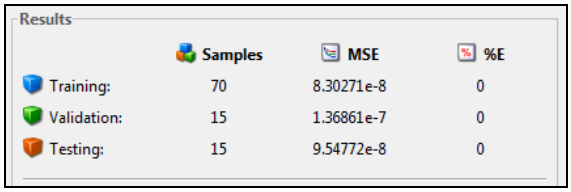

Figure 3.11: MSE Values and Percentage Errors for Fingerprint Simulation Scenario 2

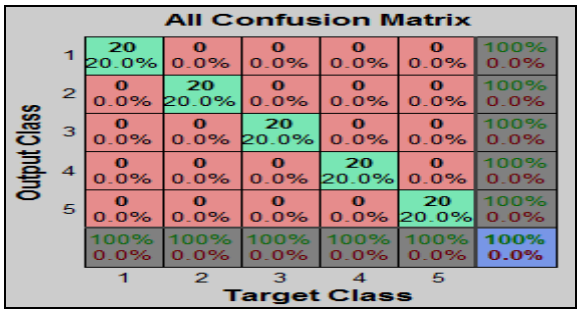

Figure 3.12: Confusion Matrix for Fingerprint Simulation Scenario 2

Table 3.3 shows the FRR test on the fingerprint recognition subsystem when tested with 25 raw fingerprint images from 5 users.

Table 3.3(a): FRRTest for Fingerprint Simulation Scenario2

	Raw Test Fingerprint Images									
Users	*I1*	*I2*	*I3*	*I4*	*I5*	*P1*	*P2*	*P3*	*P4*	*P5*
Ivan	1	1	1	1	1	0	0	0	0	0
Paul	0	0	0	0	0	1	1	1	1	1
Lan	0	0	0	0	0	0	0	0	0	0
Esther	0	0	0	0	0	0	0	0	0	0
Steven	0	0	0	0	0	0	0	0	0	0
FR	N	N	N	N	N	N	N	N	N	N

Table 3.3(b): FRRTest for Fingerprint Simulation Scenario2

	Raw Test Fingerprint Images									
Users	*L 1*	*L 2*	*L 3*	*L4*	*L5*	*E1*	*E2*	*E3*	*E4*	*E5*
Ivan	0	0	0	0	0	0	0	0	0	0
Paul	0	0	0	0	0	0	0	0	0	0
Lan	x	x	x	x	x	0	0	0	0	0
Esther	0	0	0	0	0	x	x	x	x	x
Steven	0	0	0	0	0	0	0	0	1	0
FR	N	N	N	N	N	N	N	N	N	N

Table 3.3(c): FRRTest for Fingerprint Simulation Scenario2

	Raw Test Fingerprint Images				
Users	*S1*	*S2*	*S3*	*S4*	*S5*
Ivan	0	0	0	0	0
Paul	0	0	0	0	0
Lan	0	0	0	0	0
Esther	0	0	0	0	0
Steven	1	1	1	1	1
FR	N	N	N	N	N

Based on the simulation results shown in Table 3.3, the FFR is calculated as shown in the following:

$$False\ Rejection\ Rate\ (FRR) = \frac{Number\ of\ False\ Rejection}{Total\ Number\ of\ Test} \times 100\%$$

$$False\ Rejection\ Rate\ (FRR) = \frac{0}{5\ users \times 5\ tests} \times 100\%$$

$$False\ Rejection\ Rate\ (FRR) = 0\%$$

Table 3.4 shows the FAR test on the fingerprint recognition subsystem when each user is asked to access the system as the other users and scan their fingerprint images for authentication:

Table 3.4(a): FAR Test for Fingerprint Simulation Scenario2

	Raw Test Fingerprint Images									
Users	*I1*	*I2*	*I3*	*I4*	*I5*	*P1*	*P2*	*P3*	*P4*	*P5*
Ivan	x	x	x	x	x	0	0	0	0	0
Paul	0	0	0	0	0	x	x	x	x	x
Lan	0	0	0	0	0	0	0	0	0	0
Esther	0	0	0	0	0	0	0	0	0	0
Steven	0	0	0	0	0	0	0	0	0	0
FR	N	N	N	N	N	N	N	N	N	N

Table 3.4(b): FARTest for Fingerprint Simulation Scenario2

	Raw Test Fingerprint Images									
Users	*L 1*	*L 2*	*L 3*	*L4*	*L5*	*E1*	*E2*	*E3*	*E4*	*E5*
Ivan	0	0	0	0	0	0	0	0	0	0
Paul	0	0	0	0	0	0	0	0	0	0
Lan	1	1	1	1	1	0	0	0	0	0
Esther	0	0	0	0	0	1	1	1	1	1
Steven	0	0	0	0	0	0	0	0	1	0
FR	N	N	N	N	N	N	N	N	N	N

Table 3.4(c): FARTest for Fingerprint Simulation Scenario2

	Raw Test Fingerprint Images				
Users	*S1*	*S2*	*S3*	*S4*	*S5*
Ivan	0	0	0	0	0
Paul	0	0	0	0	0
Lan	0	0	0	0	0
Esther	0	1	0	0	0
Steven	x	x	x	X	x
FR	N	**Y**	N	N	N

Based on the simulation results shown in Table 3.4, the FAR is calculated as shown in the following:

$$False\ Acceptance\ Rate\ (FAR) = \frac{Number\ of\ False\ Acceptance}{Total\ Number\ of\ Test} \times 100\%$$

$$False\ Aceptance\ Rate\ (FAR) = \frac{1}{5\ users \times 5\ tests} \times 100\%$$

$$False\ Acceptance\ Rate\ (FAR) = 4\%$$

E. Finger Print Simulation Scenario 3:100 hiddenNeurons

Figure 3.13 shows the line plot on training performance of the neural network, with respect to the MSE value and Figure 3.14 shows the MSE values and percentage errors for each type of the training data. Figure 3.15 shows the confusion matrix.

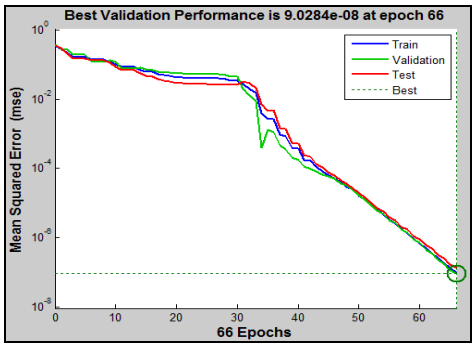

Figure 3.13: Training Performance for Fingerprint Simulation Scenario 3

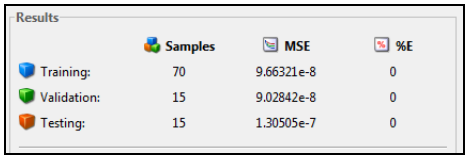

Figure 3.14: MSE Values and Percentage Errors for Fingerprint Simulation Scenario 3

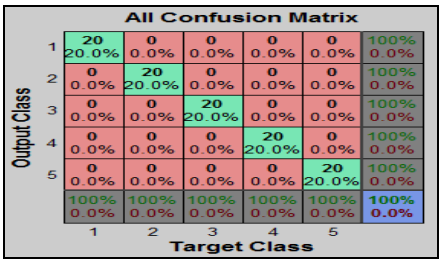

Figure 3.15: Confusion Matrix for Fingerprint Simulation Scenario 3

Table 3.5 shows the FRR test on the fingerprint recognition subsystem when tested with 25 raw fingerprint images from 5 users:

Table 3.5(a): FRRTest for Fingerprint Simulation Scenario3

	Raw Test Fingerprint Images									
Users	*I1*	*I2*	*I3*	*I4*	*I5*	*P1*	*P2*	*P3*	*P4*	*P5*
Ivan	1	1	1	1	1	0	0	0	0	0
Paul	0	0	0	0	0	1	1	1	1	1
Lan	0	0	0	0	0	0	0	0	0	0
Esther	0	0	0	0	0	0	0	0	0	0
Steven	0	0	0	0	0	0	0	0	0	0
FR	N	N	N	N	N	N	N	N	N	N

Table 3.5(b): FRRTest for Fingerprint Simulation Scenario3

	Raw Test Fingerprint Images									
Users	*L 1*	*L 2*	*L 3*	*L4*	*L5*	*E1*	*E2*	*E3*	*E4*	*E5*
Ivan	0	0	0	0	0	0	0	0	0	0
Paul	0	0	0	0	0	0	0	0	0	0
Lan	1	1	1	1	1	0	0	0	0	0
Esther	0	0	0	0	0	1	1	1	1	1
Steven	0	0	0	0	0	0	0	0	1	0
FR	N	N	N	N	N	N	N	N	N	N

Table 3.5(c): FRRTest for Fingerprint Simulation Scenario3

	Raw Test Fingerprint Images				
Users	*S1*	*S2*	*S3*	*S4*	*S5*
Ivan	0	0	0	0	0
Paul	0	0	0	0	0
Lan	0	0	0	0	0
Esther	0	0	0	0	0
Steven	1	1	1	1	1
FR	N	N	N	N	N

Based on the simulation results shown in Table 3.5, the FFR is calculated as shown in the following:

$$False\ Rejection\ Rate\ (FRR) = \frac{Number\ of\ False\ Rejection}{Total\ Number\ of\ Test} \times 100\%$$

$$False\ Rejection\ Rate\ (FRR) = \frac{0}{5\ users \times 5\ tests} \times 100\%$$

$$False\ Rejection\ Rate\ (FRR) = 0\%$$

Table 3.6 shows the FAR test on the fingerprint recognition subsystem when each user is asked to access the system as the other users and scan their fingerprint images for authentication:

Table 3.6(a): FARTest for Fingerprint Simulation Scenario2

	Raw Test Fingerprint Images									
Users	*I1*	*I2*	*I3*	*I4*	*I5*	*P1*	*P2*	*P3*	*P4*	*P5*
Ivan	x	x	x	x	x	0	0	0	0	0
Paul	0	0	0	0	0	x	x	x	x	x
Lan	0	0	0	0	0	0	0	0	0	0
Esther	0	0	0	0	0	0	0	0	0	0
Steven	0	0	0	0	0	0	0	0	0	0
FA	N	N	N	N	N	N	N	N	N	N

Table 3.6(b): FARTest for Fingerprint Simulation Scenario2

	Raw Test Fingerprint Images									
Users	*L 1*	*L 2*	*L 3*	*L4*	*L5*	*E1*	*E2*	*E3*	*E4*	*E5*
Ivan	0	0	0	0	0	0	0	0	0	0
Paul	0	0	0	0	0	0	0	0	0	0
Lan	x	x	x	x	x	0	0	0	0	0
Esther	0	0	0	0	0	x	x	x	x	x
Steven	0	0	0	0	0	0	0	0	1	0
FA	N	N	N	N	N	N	N	N	N	N

Table 3.6(c): FARTest for Fingerprint Simulation Scenario2

	Raw Test Fingerprint Images				
Users	*S1*	*S2*	*S3*	*S4*	*S5*
Ivan	0	0	0	0	0
Paul	0	0	0	0	0
Lan	0	0	0	0	0
Esther	0	0	0	0	0
Steven	x	x	x	x	x
FA	N	N	N	N	N

Based on the simulation results shown in Table 3.6, the FAR is calculated as shown in the following:

$$False\ Acceptance\ Rate\ (FAR) = \frac{Number\ of\ False\ Acceptance}{Total\ Number\ of\ Test} \times 100\%$$

$$False\ Acceptance\ Rate\ (FAR) = \frac{0}{5\ users \times 5\ tests} \times 100\%$$

$$False\ Acceptance\ Rate\ (FAR) = 0\%$$

The time taken to train the neural network has been taken into consideration. Therefore, an additional test is conducted to record the average time taken to train the neural network. Table 3.7 shows results of 10 simulation tests to, calculate the time taken to train the neural network.

Table 3.7: Neural Networks Training Time

No. of Test	Training Time for Fingerprint Neural Network (sec)
1	1.33
2	1.01
3	1.06
4	1.17
5	1.05
6	1.00
7	1.22
8	1.62
9	1.12
10	1.56
Total	12.14

The average time taken for neural networks training is calculated as follows:

$$Average\ Training\ Time = \frac{Total\ Training\ Time}{No.\ of\ Test}$$

$$Average\ Training\ Time = \frac{12.14}{10}$$

$$\therefore Average\ Training\ Time = 1.214\ sec$$

Table 3.8: Simulation Results for Finger Print Recognition System

Simulation Scenario	Number of Hidden Neuron	MSE	Epoch	Percentage Error	FRR	FAR
1	10	2.7971×10^{-7}	95	0%	8%	4%
2	50	1.3686×10^{-7}	56	0%	0%	4%
3	100	9.0284×10^{-9}	66	0%	0%	0%

With reference to the results tabulated for the fingerprint recognition test in Table 3.8, the fingerprint recognition system has shown desired outcomes that achieve the objective of obtaining FRR and FAR below 10%, for all three simulation scenarios. The fingerprint recognition system has shown better recognition performance compared with voice and face recognition systems. This is due to the

usage of minutiae extraction algorithm for fingerprint feature extraction is a reliable and accurate technique, to represent fingerprint biometric data for the neural network to recognize.

A slight recognition error may be due to the variation in finger placement on the fingerprint reader during the fingerprint acquisition process. Consequently, the minutiae points are extracted at a different region, resulting in incorrect recognition. To avoid this, all training and testing fingerprint samples are acquired consistently in terms of placement.

Another factor affecting the recognition accuracy is the quality of the original fingerprint image. Due to the usage of capacitive fingerprint reader, it is sensitive to dirt in the environment, causing speckles of white dots in the fingerprint image. This is a minor issue, as these white dots can be removed through pre-processing of the fingerprint image using 2-D Median filter and 2-D adaptive Wiener filter. Figure 3.16 illustrates a comparison between fingerprint images with and without filters, as well as comparing the resultant extracted minutiae points:

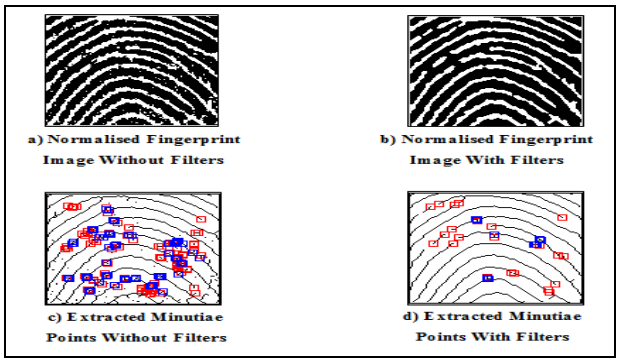

Figure 3.16: Comparison of Fingerprint Images With and Without Filters

As seen in Figure 3.16(c), the thinned image contained numerous dots creating fake termination and bifurcation points, which results in extraction of fake minutiae points. In contrast, Figure 3.16(d) shows a more refined thinned image, where the minutiae points are accurately extracted. Thus, the reduction in fake minutiae points using the mentioned filters will improve the recognition accuracy. However, when the fingerprint reader is repeatedly used, excessive dirt will accumulate on the reader, even applying the filters is not sufficient to repair the fingerprint image. The dirt can be removed by cleaning the fingerprint reader with a soft tissue.

IV. CONCLUSION

Thus, the proposed integrated finger print recognition system utilizes minutiae extraction to extract the ridge termination and bifurcation points of the fingerprint. Furthermore the performance of the integrated system is evaluated and from the simulated results it is found that the system has achieved an overall FRR of 8% and FAR of 8%. The calculated average time taken to train the neural networks for the overall system is 1.214 seconds only. In future, further more biometric measurement technique can be integrated together to robust the system so developed.

REFERENCES

[1] Al-Taee, M. A., et al. (2009) Biometric-Based Security System for Plaintext e-Mail Messages. *2009 Second International Conference on Developments in eSystems Engineering*. Abu Dhabi, 14th to 16th December 2009. Abu Dhabi: DESE. pp. 202 – 206.

[2] Francis , C. et al. (2008) Comparison of the Performances of a Fuzzy Logic and a Neural Networks based Controllers for Seismic Vibration Elimination. *International Conference on Computer Engineering & Systems*. Cairo, 25th to 27th November 2008. Cairo: ICCES, pp. 387 – 391

[3] Gregory , P. & Simon, M. A. (2008) *Biometric For Dummies*. Indianapolis: Wiley Publishing, Inc.

[4] Sanderson , C. & Paliwal, K. K. (2002) Information Fusion and Person Verification Using Speech and Face. *IDIAP*. Martigny, September 2002. Martigny: IDIAP. pp. 2 – 33.

[5] Seal, A. et al. (2011) Minutiae Based Thermal Face Recognition using Blood Perfusion Data. *2011 International Coference on Image Information Processing*. Himachal Pradesh, 3rd to 5th November 2011. Himachal Pradesh: ICIIP: pp. 1 – 4.

[6] Vasuhi , S. et al (2010) An efficient multi-modal biometric person authentication system using fuzzy logic. Proceedings of the 2nd International Conference on Advanced Computing, Dec. 14-16, IEEE Xplore Press, Chennai, pp: 74-81. DOI: 10.1109/ICOAC.2010.

[7] Yesu , K. et al. (2012) Innovative Feature Extraction Method for Artificial Neural Network Based Face Recognition. *3rd National Conference on Emerging Trends and Application in Computer Science*. Shilong, 30th to 31st March 2012. Shilong: NCETACS. pp. 137 – 142.

[8] Zhang , K. et al. (2010). Study on the Embedded Fingerprint Image Recognition System. *2010 International Conference of Information Science and Management Engineering*. Xi'an, 7th to 8th August 2010. Xi'an: ISME. pp. 169 – 172.

[9] Zhang, X. et al. (2011) Fingerprint Minutia Extraction Algorithm Based on DSP. *2011 International Conference on Electronic & Mechanical Engineering and Information Technology*. Harbin, 12th to 14th August 2011. Harbin: EMEIT. pp. 4805 – 4808.

[10] Zin, M. & Myint, M. (2011) Fingerprint Recognition System for Low Quality Images. *2011 Proceedings of SICE Annual Conference*. Tokyo, 13th to 18th September 2011. Tokyo: SICE. pp. 1133 – 1137.