10

Fingerprint Recognition

Amira Saleh, Ayman Bahaa and A. Wahdan
Computer and systems engineering department
Faculty of Engineering /Ain Shams University
Egypt

1. Introduction

Recognition of persons by means of biometric characteristics is an emerging phenomenon in modern society. It has received more and more attention during the last period due to the need for security in a wide range of applications. Among the many biometric features, the fingerprint is considered one of the most practical ones. Fingerprint recognition requires a minimal effort from the user, does not capture other information than strictly necessary for the recognition process, and provides relatively good performance. Another reason for the popularity of fingerprints is the relatively low price of fingerprint sensors, which enables easy integration into PC keyboards, smart cards and wireless hardware (Maltoni et al., 2009).
Fig. 1 presents a general framework for a general fingerprint identification system (FIS) (Ji & Yi, 2008). Fingerprint matching is the last step in Automatic Fingerprint Identification System (AFIS). Fingerprint matching techniques can be classified into three types:

- Correlation-based matching,
- Minutiae-based matching, and
- Non-Minutiae feature-based matching.

Minutiae-based matching is the most popular and widely used technique, being the basis of the fingerprint comparison.

2. Previous work and motivation

In (Bazen & Gerez, 2003), a novel minutiae matching method is presented that describes elastic distortions in fingerprints by means of a thin-plate spline model, which is estimated using a local and a global matching stage. After registration of the fingerprints according to the estimated model, the number of matching minutiae can be counted using very tight matching thresholds. For deformed fingerprints, the algorithm gives considerably higher matching scores compared to rigid matching algorithms, while only taking 100 ms on a 1 GHz P-III machine. Furthermore, it is shown that the observed deformations are different from those described by theoretical models proposed in the literature.
In (Liang & Asano, 2006), minutia polygons are used to match distorted fingerprints. A minutia polygon describes not only the minutia type and orientation but also the minutia shape. This allows the minutia polygon to be bigger than the conventional tolerance box without losing matching accuracy. In other words, a minutia polygon has a higher ability to

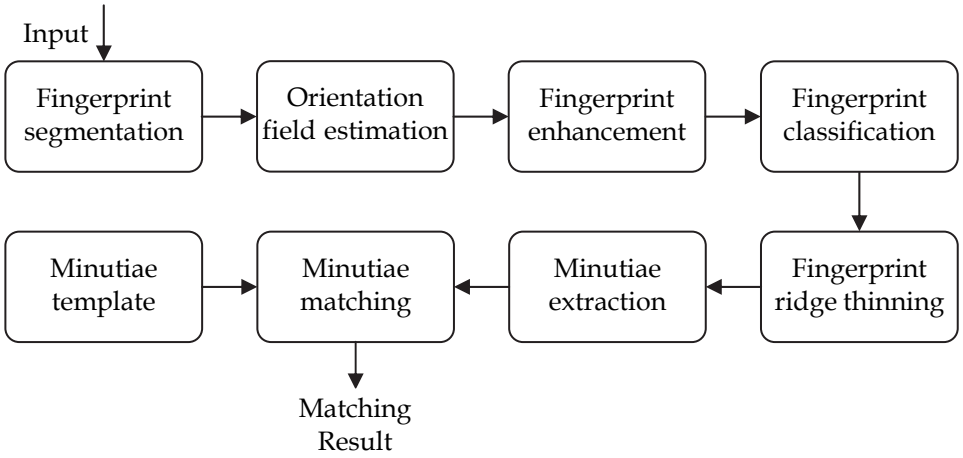

Fig. 1. General block diagram for a Fingerprint Identification System.

tolerate distortion. Furthermore, the proposed matching method employs an improved distortion model using a Multi-quadric basis function with parameters. Adjustable parameters make this model more suitable for fingerprint distortion. Experimental results show that the proposed method is two times faster and more accurate (especially, on fingerprints with heavy distortion) than the method in (Bazen & Gerez, 2003).

In (Jiang & Yau, 2000), a new fingerprint minutia matching technique is proposed, which matches the fingerprint minutiae by using both the local and global structures of minutiae. The local structure of a minutia describes a rotation and translation invariant feature of the minutia in its neighborhood. It is used to find the correspondence of two minutiae sets and increase the reliability of the global matching. The global structure of minutiae reliably determines the uniqueness of fingerprint. Therefore, the local and global structures of minutiae together provide a solid basis for reliable and robust minutiae matching. The proposed minutiae matching scheme is suitable for an on-line processing due to its high processing speed. Their experimental results show the performance of the proposed technique.

In (Jain et al., 2001), a hybrid matching algorithm that uses both minutiae (point) information and texture (region) information is presented for matching the fingerprints. Their results obtained show that a combination of the texture-based and minutiae-based matching scores leads to a substantial improvement in the overall matching performance. This work was motivated by the small contact area that sensors provide for the fingertip and, therefore, only a limited portion of the fingerprint is sensed. Thus multiple impressions of the same fingerprint may have only a small region of overlap. Minutiae-based matching algorithms, which consider ridge activity only in the vicinity of minutiae points, are not likely to perform well on these images due to the insufficient number of corresponding points in the input and template images.

In (Eckert et al., 2005), a new and efficient method for minutiae-based fingerprint matching is proposed, which is invariant to translation, rotation and distortion effects of fingerprint patterns. The algorithm is separated from a prior feature extraction and uses a compact description of minutiae features in fingerprints. The matching process consists of three major steps:

- Finding pairs of possibly corresponding minutiae in both fingerprint patterns,
- Combining these pairs to valid tuples of four minutiae each, containing two minutiae from each pattern.
- The third step is the matching itself.
  It is realized by a monotonous tree search that finds consistent combinations of tuples with a maximum number of different minutiae pairs. The approach has low and scalable memory requirements and is computationally inexpensive.

In (Yuliang et al., 2003), three ideas are introduced along the following three aspects:

- Introduction of ridge information into the minutiae matching process in a simple but effective way, which solves the problem of reference point pair selection with low computational cost;
- Use of a variable sized bounding box to make their algorithm more robust to non-linear deformation between fingerprint images;
- Use of a simpler alignment method in their algorithm.
  Their experiments using the Fingerprint Verification Competition 2000 (FVC2000) databases with the FVC2000 performance evaluation show that these ideas are effective.

In (Zhang et al., 2008), a novel minutiae indexing method is proposed to speed up fingerprint matching, which narrows down the searching space of minutiae to reduce the expense of computation. An orderly sequence of features is extracted to describe each minutia and the indexing score is defined to select minutiae candidates from the query fingerprint for each minutia in the input fingerprint. The proposed method can be applied in both minutiae structure-based verification and fingerprint identification. Experiments are performed on a large-distorted fingerprint database (FVC2004 DB1) to approve the validity of the proposed method.

In most existing minutiae-based matching methods, a reference minutia is chosen from the template fingerprint and the query fingerprint, respectively. When matching the two sets of minutiae, the template and the query, firstly, reference minutiae pair is aligned coordinately and directionally, and secondly, the matching score of the remaining minutiae is evaluated. This method guarantees satisfactory alignments of regions adjacent to the reference minutiae. However, the alignments of regions far away from the reference minutiae are usually not so satisfactory. In (Zhu et al., 2005), a minutia matching method based on global alignment of multiple pairs of reference minutiae is proposed. These reference minutiae are commonly distributed in various fingerprint regions. When matching, these pairs of reference minutiae are to be globally aligned, and those region pairs far away from the original reference minutiae will be aligned more satisfactorily. Their experiment shows that this method leads to improvement in system identification performance.

In (Jain et al., 1997a), the design and implementation of an on-line fingerprint verification system is described. This system operates in two stages: minutia extraction and minutia matching. An improved version of the minutia extraction algorithm proposed by (Ratha et al., 1995), which is much faster and more reliable, is implemented for extracting features from an input fingerprint image captured with an on-line inkless scanner. For minutia matching, an alignment-based elastic matching algorithm has been developed. This algorithm is capable of finding the correspondences between minutiae in the input image and the stored template without resorting to exhaustive search and has the ability of adaptively compensating for the nonlinear deformations and inexact pose transformations between fingerprints. The system has been tested on two sets of fingerprint images captured with inkless scanners. The verification accuracy is found to be acceptable. Typically, a

complete fingerprint verification procedure takes, on an average, about eight seconds on a SPARC 20 workstation. These experimental results show that their system meets the response time requirements of on-line verification with high accuracy.
In (Luo et al., 2000), a minutia matching algorithm which modified (Jain et al., 1997a) algorithm is proposed The algorithm can better distinguish two images from different fingers and is more robust to nonlinear deformation. Experiments done on a set of fingerprint images captured with an inkless scanner shows that the algorithm is fast and of high accuracy.
In (Jie et al., 2006), a new fingerprint minutiae matching algorithm is proposed, which is fast, accurate and suitable for the real time fingerprint identification system. In this algorithm, the core point is used to determine the reference point and a round bounding box is used for matching. Experiments done on a set of fingerprint images captured with a scanner showed that the algorithm is faster and more accurate than that in (Luo et al., 2000) algorithm.
There are two major shortcomings of the traditional approaches to fingerprint representation (Jain et al., 2000):

1. For a considerable fraction of population, the representations based on explicit detection of complete ridge structures in the fingerprint are difficult to extract automatically. The widely used minutiae-based representation does not utilize a significant component of the rich discriminatory information available in the fingerprints. Local ridge structures cannot be completely characterized by minutiae.
2. Further, minutiae-based matching has difficulty in quickly matching two fingerprint images containing different number of unregistered minutiae points.

The filter-based algorithm in (Jain et al., 2000) uses a bank of Gabor filters to capture both local and global details in a fingerprint as a compact fixed length FingerCode. The fingerprint matching is based on the Euclidean distance between the two corresponding FingerCodes and hence is extremely fast. Verification accuracy achieved is only marginally inferior to the best results of minutiae-based algorithms published in the open literature (Jain et al., 1997b). Proposed system performs better than a state-of-the-art minutiae-based system when the performance requirement of the application system does not demand a very low false acceptance rate. Finally, it is shown that the matching performance can be improved by combining the decisions of the matchers based on complementary (minutiae-based and filter-based) fingerprint information.
Motivated by this analysis, a new algorithm is proposed in this chapter. This novel algorithm is minutiae-based matching algorithm. The proposed matching algorithm is described in section 3, the advantages are drawn in section 4, and finally, implementation, performance evaluation of the algorithm and conclusion are explained in section 5.

3. Proposed matching algorithm

Any Fingerprint Identification System (FIS) has two phases, fingerprint enrolment and fingerprint matching (identification or verification).

3.1 Enrolment phase

Fig. 2 shows the steps of the enrolment phase of the proposed matching algorithm, which is divided into the following steps:

1. Get the core point location of the fingerprint to be enrolled after applying enhancement process.

2. Extract all minutiae from the fingerprint image.
3. From output data of step2, get the minutiae locations (x, y coordinates) together with their type: type1 for termination minutiae and type2 for bifurcation minutiae.

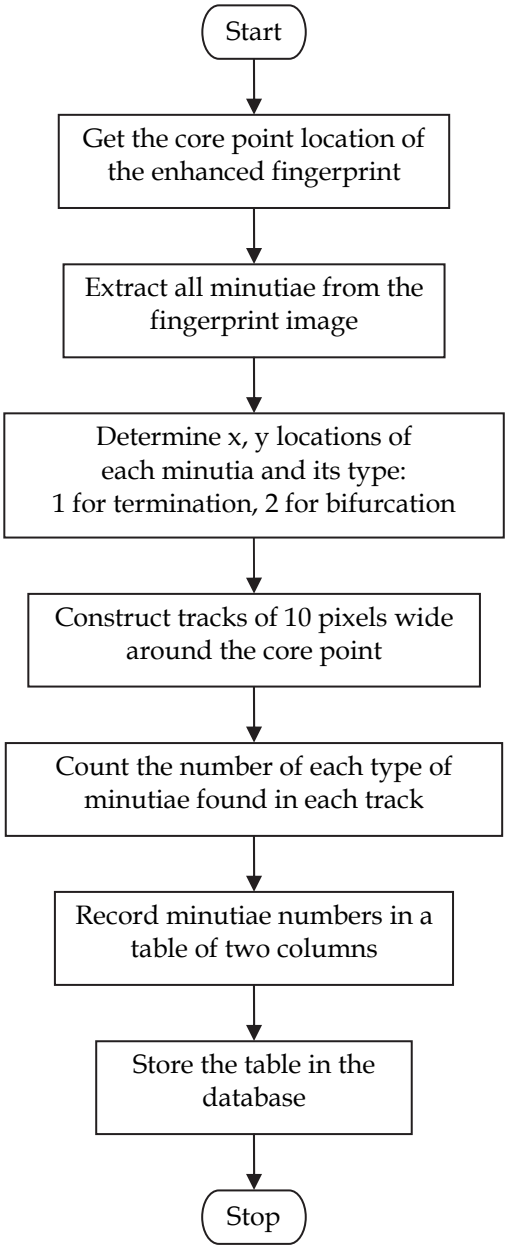

Fig. 2. Flowchart of the enrolment phase of the proposed matching algorithm.

4. Construct tracks of 10 pixels wide centred at the core point.
5. In each track, count the number of minutiae of type1 and the number of minutiae of type2.
6. Construct a table of two columns, column 1 for type1 minutiae and column 2 for type2 minutiae, having number of rows equal to number of found tracks.
7. In the first row, record the number of minutiae of type1 found in the first track in the first column, and the number of minutiae of type2 found in the first track in the second column.
8. Repeat step 7 for the remaining tracks of the fingerprint, and then store the table in the database.

This enrolment phase will be repeated for all prints of the same user's fingerprint. The number of prints depends on the application requirements at which the user registration takes place. For FVC2000 (Maio et al., 2002), there are 8 prints for each fingerprint. So, eight enrolments will be required for each user to be registered in the application. Finally, eight tables will be available in the database for each user.

3.2 Verification phase

For authenticating a user, verification phase should be applied on the user's fingerprint to be verified at the application. Fig. 3. shows the steps of the verification phase of the proposed matching algorithm, which is divided into the following steps:

1. Capture the fingerprint of the user to be verified.
2. Apply the steps of enrolment phase, described in section 3.1, on the captured fingerprint to obtain its minutiae table T.
3. Get all the minutiae tables corresponding to the different prints of the claimed fingerprint from the database.
4. Get the absolute differences, cell by cell, between minutiae table T and all minutiae tables of the claimed fingerprint taken from the database, now we have eight difference tables.
5. Get the summations of all cells in each of column1 (type1) and column2 (type2) for each difference table, now we have sixteen summations.
6. Get the geometric mean of the eight summations of type1 columns (gm1), and the geometric mean of the eight summations of type2 columns (gm2).
7. Check: if gm1<= threshold1 and gm2<=threshold2 then the user is genuine and accept him; else the user is imposter and reject him.

4. Advantages of the proposed matching algorithm

The proposed minutiae-based matching algorithm has the following advantages:

1. Since all cells in each minutiae table, representing the fingerprint in database, contain *just* the number of minutiae of type1 or type2 in each track around the core point of the fingerprint, neither position (x or y) nor orientation (θ) of the minutiae is considered; the algorithm is *rotation* and *translation invariant*.
2. The numbers of minutiae to be stored in the database *need less storage* than traditional minutiae-based matching algorithms which store position and orientation of each minutia. Experiments show that nearly 50% reduction in storage size is obtained.
3. Matching phase itself *takes less time* which, as will be shown in following sections, reaches 0.00134 sec.

5. Implementation of the proposed matching algorithm

Using MATLAB Version 7.9.0.529 (R2009b), both proposed enrolment and verification phases are implemented as described in next two subsections:

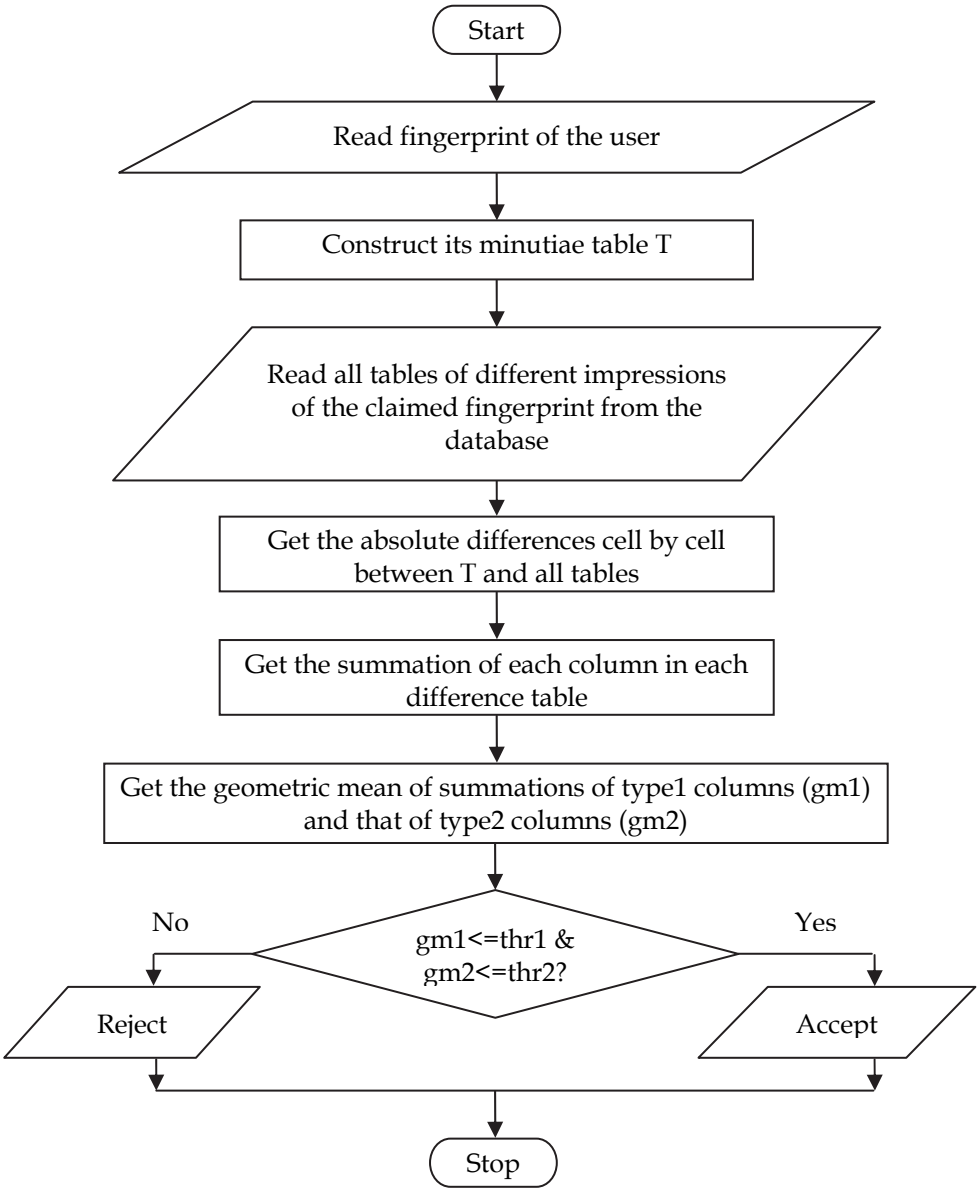

Fig. 3. Flowchart of the proposed verification phase.

5.1 Enrolment phase

5.1.1 Enhancement of the fingerprint image

The first step is to enhance the fingerprint image using Short Time Fourier Transform STFT analysis (O'Gorman, 1998). The performance of a fingerprint matching algorithm depends critically upon the quality of the input fingerprint image. While the quality of a fingerprint image may not be objectively measured, it roughly corresponds to the clarity of the ridge structure in the fingerprint image, and hence it is necessary to enhance the fingerprint image. Since the fingerprint image may be thought of as a system of oriented texture with non-stationary properties, traditional Fourier analysis is not adequate to analyze the image completely as the STFT analysis does (Yang & Park, 2008). Fingerprint enhancement MATLAB code is available at (http://www.hackchina.com/en/cont/18456).

The algorithm for image enhancement consists of two stages as summarized below:

Step 1. STFT analysis

1. For each overlapping block in an image, generate and reconstruct a ridge orientation image by computing gradients of pixels in a block, and a ridge frequency image through obtaining the FFT value of the block, and an energy image by summing the power of FFT value;
2. Smoothen the orientation image using vector average to yield a smoothed orientation image, and generate a coherence image using the smoothed orientation image;
3. Generate a region mask by thresholding the energy image;

Step 2. Apply Enhancement

For each overlapping block in the image, the next five sub-steps are applied:

1. Generate an angular filter Fa centered on the orientation in the smoothed orientation image with a bandwidth inversely proportional to coherence image;
2. Generate a radial filter Fr centered on frequency image;
3. Filter a block in the FFT domain, F=F×Fa×Fr;
4. Generate the enhanced block by inverse Fourier transform IFFT(F);
5. Reconstruct the enhanced image by composing enhanced blocks, and yield the final enhanced image with the region mask.

The result of the enhancement process is shown in Fig. 4, where Fig. 4.a is taken from FVC2000 DB1_B (108_5) and Fig. 4.b is the enhanced version of Fig. 4.a.

5.1.2 Get core point of the enhanced fingerprint

Core point MATLAB code is available at (http://www.hackchina.com/en/cont/18456) where the idea of determining the reference point is taken from (Yang & Park, 2008), which is described as follows:

The reference point is defined as "the point of the maximum curvature on the convex ridge (Liu et al., 2005)" which is usually located in the central area of fingerprint. The reliable detection of the position of a reference point can be accomplished by detecting the maximum curvature using complex filtering methods (Nilsson & Bigun, 2003).

They apply complex filters to ridge orientation field image generated from original fingerprint image. The reliable detection of reference point with the complex filtering methods is summarized below:

1. For each overlapping block in an image;
    a. Generate a ridge orientation image with the same method in STFT analysis;

(a) (b)

Fig. 4. a) Fingerprint image 108_5 from DB1_B in FVC2000, b) Enhanced version of fingerprint image 108_5.

b. Apply the corresponding complex filter $h = (x + iy)^m g(x, y)$ centered at the pixel orientation in the orientation image, where m and $g(x, y) = \exp\{-((x^2 + y^2)/2\sigma^2))\}$ indicate the order of the complex filter and a Gaussian window, respectively;
c. For $m = 1$, the filter response of each block can be obtained by a convolution, $h * O(x, y) = g(y) * ((xg(x))^t * O(x,y)) + ig(x)^t * ((yg(y) * O(x, y)))$
where $O(x, y)$ represents the pixel orientation image;

2. Reconstruct the filtered image by composing filtered blocks.

The maximum response of the complex filter in the filtered image can be considered as the reference point. Since there is only one output, the unique output point is taken as the reference point (core point).

5.1.3 Minutiae extraction

To extract minutiae from the enhanced fingerprint image, the minutiae extraction method (Maltoni et al., 2003) is used. Hence we have three information for each minutia: x and y coordinates of its location, type of minutia (type1 if it is a termination, type2 if it is a bifurcation).

The result of this minutiae extraction stage is shown in Fig. 5, where Fig. 5.a is the same as Fig. 4.b, Fig. 5.b shows the termination minutiae in circles and the bifurcation minutiae in diamonds together with the core point of the fingerprint with an asterisk.

(a) (b)

Fig. 5. a) Enhanced Fingerprint image 108_5 from DB1_B in FVC2000, b) core point (asterisk), terminations (circles) and bifurcations (diamonds).

5.1.4 Construct the minutiae table

From the output of the minutiae extraction step, the proposed minutiae table is constructed as following:

a. Get all minutiae locations together with their types.
b. Using Euclidean distances, get the distances between all the minutiae and the core point of the fingerprint: if the core location is at (x_c, y_c) and a minutia location is at (x, y), the Euclidean distance between them will be:

$$\sqrt{(x-x_c)^2+(y-y_c)^2}$$

c. Construct tracks of 10×n pixels wide (where n=1…max_distance/10) centered at the core point until all minutiae are exhausted, the track width is chosen to be 10 as the average distance (in pixels) between two consecutive ridges is 10 pixels, this is achieved in the fingerprint 108_5 in Fig. 5.a. where it is of 96 dpi resolution.
d. In each track, count the number of existed minutiae of type1 and the number of existed minutiae of type2.
e. Construct a table of two columns, column 1 for type1 minutiae and column 2 for type2 minutiae, having a number of rows that is equal to the number of found tracks.
f. In the first row, record the number of minutiae of type1 found in the first track in the first column, and record the number of minutiae of type2 found in the first track in the second column.

g. Repeat last step for the remaining tracks of the fingerprint until all tracks are processed, and then store the minutiae table in the database.

The resulted minutiae table from Fig. 5.b is as shown in Table 1. The first column is for illustration only but in MATLAB, it does not exist and it is used as the index for each row of the minutiae table consisting of just two columns.

To validate the table's data, the total number of the minutiae of fingerprint 108_5 of both types: termination and bifurcation, in Fig. 5.b is found to be the total summation of both columns of minutiae table which is equal to 51 minutiae.

fingerprint	108_5		108_7	
Track number	# of type1 minutiae	# of type2 minutiae	# of type1 minutiae	# of type2 minutiae
1	0	0	1	0
2	1	2	1	2
3	0	0	1	0
4	0	1	0	0
5	0	1	1	2
6	0	2	1	1
7	0	3	6	1
8	2	2	5	1
9	0	2	2	1
10	1	2	4	0
11	6	1	3	1
12	3	1	2	0
13	3	0	5	1
14	4	1	1	1
15	1	2	3	1
16	0	1	2	1
17	0	0	0	2
18	1	0	2	0
19	1	0	4	1
20	0	0	4	1
21	0	0	3	2
22	1	0	1	0
23	1	4	1	1
24	0	1	--	--

Table 1. The minutiae table of fingerprint 108_5 and 108_7 from DB1_B in FVC2000

5.2 Verification phase

5.2.1 Capture the fingerprint to be verified

The user, who is claiming he is (e.g. M), will put his finger on the scanner to be captured at the application he wants to access. Now, a fingerprint will be available to be verified to check if he is actually M so he will be accepted or he is not so he will be rejected.

5.2.2 Construct the minutiae table of that fingerprint

The same steps of the enrolment explained in section 5.1 will be applied on the fingerprint obtained from the previous step. Now, a minutiae table T corresponding to the fingerprint under test will be built.

5.2.3 Get all corresponding minutiae tables from the database

In FVC2000 (Maio et al., 2002), each fingerprint has eight prints, so to verify a certain input fingerprint, all the corresponding minutiae tables of different prints of that claimed fingerprint stored in the database must be fetched.

Taking as an example the fingerprint 108_7 (see Fig. 6.a) taken from DB1_B in FVC2000 to be verified, and applying the same steps of enrolment, the results are shown in Fig. 6 where Fig. 6.b shows the enhanced version of Fig. 6.a, and Fig. 6.c shows its thinned version together with the terminations in circles, the bifurcations in diamonds and finally the core point in asterisk. As can be shown, because the image is of poor quality, the number of minutiae is so different.

Now the minutiae table is ready to be constructed as shown in Table 1. Summing all the minutiae of both types results in 73 minutiae which is different from the number before for fingerprint 108_5 which was 51 minutiae.

Following the same steps, fingerprints 108_1, 108_2, 108_3, 108_4, 108_6, and 108_8 are taken from DB1_B in FVC2000, and their corresponding minutiae tables are constructed in six tables.

5.2.4 Calculate the absolute differences between minutiae table of the input fingerprint and all minutiae tables of the claimed fingerprint

Now, the absolute differences between minutiae table corresponding to fingerprint 108_7 and all minutiae tables corresponding to fingerprints 108_1, 108_2, 108_3, 108_4, 108_5, 108_6, and 108_8 are calculated. Because the sizes (number of rows) of minutiae tables are not equal, the minimum size must be determined to be able to perform the absolute subtraction on the same size for different tables. The minimum number of tracks (rows) found in DB1_B is 14.

So, only the first 14 rows of each minutiae table will be considered during the absolute differences calculation. Table 2 shows the absolute differences between the minutiae table of fingerprint 108_7 and the minutiae tables of fingerprints 108_1 and 108_2.

5.2.5 Get the summation of each column in each difference table

At the bottom of Table 2 in the row titled "sum", the summation of each column in each difference table is drawn, applying the same steps for calculating the absolute differences between the minutiae table of fingerprint 108_7 and the minutiae tables of fingerprints 108_3, 108_4, 108_5, 108_6, and 108_8, will result in seven summations for type1 columns, and other seven summations for type2 columns.

(a)

(b) (c)

Fig. 6. a) Fingerprint image 108_7 from DB1_B in FVC2000, b) Enhanced version of fingerprint image 108_7, c) core point (asterisk), terminations (circles) and bifurcations (diamonds).

5.2.6 Get the geometric mean of the resulted summations for both of type1 and type2

Get the geometric mean of the summations of type1 columns in all difference tables. The geometric mean, in mathematics, is a type of mean or average, which indicates the central tendency or typical value of a set of numbers (http://en.wikipedia.org/wiki/Geometric_mean). It is similar to the arithmetic mean, which is what most people think of with the word "average", except that the numbers are multiplied and then the n^{th} root (where n is the count of numbers in the set) of the resulting product is taken.

$$gm1 = \sqrt[7]{25 \times 24 \times 24 \times 26 \times 27 \times 17 \times 69} = 27.488$$

Get the geometric mean of the summations of type2 columns in all difference tables.

$$gm2 = \sqrt[7]{9 \times 9 \times 10 \times 15 \times 11 \times 7 \times 6} = 9.2082$$

Check the values of gm1 and gm2:

If gm1 <= threshold1 and gm2 <= threshold2 then
the user is genuine and accept him
else
the user is imposter and reject him

	abs(108_7-108_1)		abs(108_7-108_2)	
Track number	# of type1 minutiae	# of type2 minutiae	# of type1 minutiae	# of type2 minutiae
1	0	0	1	0
2	0	2	1	1
3	0	0	0	0
4	0	1	2	0
5	2	1	0	1
6	2	1	1	1
7	2	1	3	1
8	5	0	4	1
9	2	0	1	1
10	4	0	2	0
11	2	1	3	1
12	1	1	1	0
13	4	1	4	1
14	1	0	1	1
Sum	25	9	24	9

Table 2. Absolute differences between minutiae table of fingerprint 108_7 and both minutiae tables of fingerprints 108_1, 108_2.

5.3 Performance evaluation

To evaluate any matching algorithm performance, some important quantities have to be measured such as (Maio et al., 2002):

- False NonMatch Rate (FNMR) often referred to as False Rejection Rate (FRR)
- False Match Rate (FMR) often referred to as False Acceptance Rate (FAR)
- Equal Error Rate (EER)
- ZeroFNMR
- ZeroFMR
- Average enroll time
- Average match time

Because the presence of the fingerprint cores and deltas is not guaranteed in FVC2000 since no attention was paid on checking the correct finger position on the sensor (Maio et al., 2002), and the core point detection is the first step in the proposed matching algorithm, another group of fingerprints has been captured experimentally; this group contains the right forefinger of 20 different persons, each is captured three times, having 60 different fingerprint images. They are numbered as follows: 101_1, 101_2, 101_3, 102_1, …., 120_1, 120_2, 120_3. All these fingerprints have a core point. This group will be tested first and then the four databases DB1, DB2, DB3, and DB4 (from FVC2000) will be tested afterwards. All steps used to evaluate the performance of the proposed algorithm are implemented.

5.3.1 Calculate False NonMatch Rate (FNMR) or False Rejection Rate (FRR)

Each fingerprint template (minutiae table) T_{ij}, i=1…20, j=1…3, is matched against the fingerprint images (minutiae tables) of F_i, and the corresponding Genuine Matching Scores (GMS) are stored. The number of matches (denoted as NGRA – Number of Genuine Recognition Attempts (Maio et al., 2002)) is $20 \times 3 = 60$.

Now FRR(t) curve will be easily computed from GMS distribution for different threshold values. Given a threshold t, FRR(t) denotes the percentage of GMS $\geq t$. Here, because the input fingerprint is verified whether it gives less difference values between corresponding minutiae tables, lower scores are associated with more closely matching images. This is the opposite of most fingerprint matching algorithms in fingerprint verification, where higher scores are associated with more closely matching images. So, the FRR(t) (or FNMR(t)) curve will start from the left not from the right as usual. Also, it is worth to be noted that the curve of FRR(t) will be a 2D surface (FRR(t_1, t_2)) because there are two thresholds as mentioned in previous section.

For example, consider the fingerprint 101_1, or any slightly different version of it, is to be matched with its other prints 101_2, 101_3, this is considered as a genuine recognition attempt because they are all from the same fingerprint. Fig. 7 shows the fingerprint 101_1 together with its enhanced thinned version where core point is shown in a solid circle, terminations are shown with circles, and bifurcations are shown with diamonds.

As an example, some noise is applied on the minutiae table of fingerprint 101_1, to act as a new user's fingerprint. Noise is a sequence of Pseudorandom integers from a uniform discrete distribution used to randomly select tracks from the minutiae table that will be changed by adding '1' to values under termination (or bifurcation) column and subtracting '1' to values under bifurcation (or termination) column in each selected track. The sequence of numbers, produced using Matlab function called "randi", is determined by the internal state of the uniform pseudorandom number generator. The number of random selected tracks is a constant ratio (30%) from the overall number of tracks in each database.

Now, minutiae tables of fingerprints 101_1, 101_2, and 101_3 will be fetched from the database. The minimum number of rows (tracks) in all the minutiae tables in the database under study is found to be 13, so only the first 13 rows of any minutiae table are considered during calculations of absolute differences.

Fig. 7. Fingerprint 101_1 and its enhanced thinned version.

The second step is to calculate the geometric mean of the sum of each of type1 and type2 absolute differences:

$$\text{gm1}=\sqrt[3]{2\times 17\times 17}=8.33\,,$$

$$\text{gm2}=\sqrt[3]{3\times 8\times 7}=5.52$$

Then, since a user is accepted if the two geometric means satisfy that:

$$\text{gm1}<=\text{threshold1}(t_1)\wedge \text{gm2}<=\text{threshold2}(t_2)$$

Using Demorgan's law, the (FNMR) or false rejection rate will be computed as follows:

$$\text{FNMR}(t_1,t_2)=\frac{\text{NGMS1s}\ >\ t_1\vee \text{NGMS2s}\ >\ t_2}{\text{NGRA}}$$

Where NGMS1s is the number of genuine matching scores computed from gm1 values, NGMS2s is the number of genuine matching scores computed from gm2 values, and NGRA is the number of genuine recognition attempts which is 60. The threshold values, t_1 and t_2, vary from 1 to 100.

The same steps are performed for the remaining fingerprints, all 60 instances. The previous example is considered as a genuine recognition attempt as the comparison is held between a

noisy version of the first of the three prints and the three true versions of them fetched from the database.

5.3.2 Calculate False Match Rate (FMR) or False Acceptance Rate (FAR)

Each fingerprint template (minutiae table) T_{ij}, i = 1…20, j = 1…3 in the database is matched against the other fingerprint images (minutiae tables) F_k, $k \neq i$ from different fingers and the corresponding *Imposter Matching Scores* ims are stored. Number of Impostor Recognition Attempts is $(20 \times 3) \times (20 - 1) = 60 \times 19 = 1140$.

Now FAR(t_1, t_2) surface will be easily computed from IMS distribution for different threshold values. Given thresholds t_1 and t_2, FAR(t_1, t_2) denotes the percentage of IMS1s <= t_1 and IMS2s<=t_2. Here, because the input fingerprint is rejected if it gives high difference values between corresponding minutiae tables; higher scores are associated with mismatching images. This is the opposite of most fingerprint matching algorithms in fingerprint verification, where lower scores are associated with mismatching images. So, the FAR(t_1, t_2) (or FMR(t_1, t_2)) surface will start from the right not from the left as usual.

For example, consider the noisy version of fingerprint 101_1 is to be matched with another fingerprint like 103, this is considered as an imposter recognition attempt because they are from different fingers. Now, all minutiae tables of fingerprints 103_1, 103_2, and 103_3 have to be fetched from the database. As before, because the minimum number of rows is 13, so only the first 13 rows of any minutiae table are considered during calculations of the absolute differences tables.

Geometric mean gm1 and gm2 are calculated as follows:

$$\text{gm1} = \sqrt[3]{26 \times 23 \times 16} = 21.23 \text{ ,}$$

$$\text{gm2} = \sqrt[3]{14 \times 13 \times 13} = 13.33$$

The same steps are performed for the remaining fingerprints; all 60 instances (20 fingerprints, each having 3 impressions) will be matched against the other 19 fingerprints, so a total of 1140 IMSs.

FMR(t_1, t_2) will be calculated as follows:

$$\text{FMR}(t_1, t_2) = \frac{\text{NIMS1s} \le t_1 \wedge \text{NIMS2s} \le t_2}{\text{NIRA}}$$

Where NIMS1s is the number of imposter matching scores computed from gm1 values, NIMS2s is the number of imposter matching scores computed from gm2 values, and NIRA is the number of imposter recognition attempts which is 1140. The threshold values, t_1 and t_2, vary from 1 to 70.

Both surfaces FRR(t_1, t_2) and FAR(t_1, t_2) are drawn in Fig. 8 with blue and red colors respectively. The intersection between the two surfaces is drawn with a solid line used in the next section.

5.3.3 Equal Error Rate EER

The Equal Error Rate is computed as the point where FMR(t) = FNMR(t). From Fig. 8, to determine the equal error rate, the intersection line between the two surfaces is drawn and then the minimum value of error rates along this line is the EER from where the values of thresholds t_1 and t_2 can be determined.

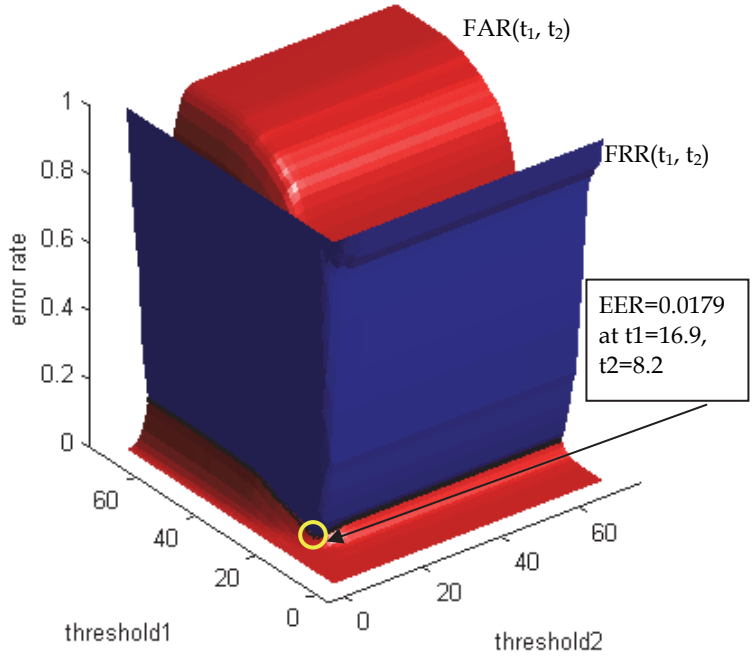

Fig. 8. FRR and FAR surfaces where the intersection between the two surfaces is drawn with a solid line.

In Fig. 8, it is shown that EER = 0.0179, where it corresponds to the threshold values of t_1 = 16.92 and t_2 = 8.21, values of thresholds are not always integers because it is not necessary for the two surfaces to intersect at integer values of thresholds.
Now to determine the integer values of thresholds that corresponds to error rates FRR and FAR, the four possible combinations of thresholds around the two thresholds given before are tested and the two values combination that gives the minimum difference between FRR and FAR (because EER is defined as the point where FRR and FAR are equal) are considered as the thresholds t_1 and t_2 that will be used for that database for any later fingerprint recognition operation.
So, the four possible combinations that threshold values t_1 and t_2 can take are: (16, 8), (16, 9), (17, 8), and (17, 9). It is found by experiment that the combination (17, 8) gives the minimum difference between FAR and FRR. So, when these thresholds are used in the proposed matching algorithm, the result is that FRR = 0.0167 and FAR = 0.0184.
True Acceptance Rate is

$$TAR = 1\text{-}FAR = 1\text{-}0.0184 = 0.9816$$

And the True Rejection Rate is

$$TRR = 1\text{-}FRR = 1\text{-}0.0167 = 0.9833$$

So, the recognition accuracy is ≈ 98% when thresholds values are t_1 = 17 and t_2 = 8.

5.3.4 ZeroFMR and ZeroFNMR

ZeroFMR is defined as the lowest FNMR at which no False Matches occur and ZeroFNMR as the lowest FMR at which no False Non-Matches occur(Maio et al., 2002):

$$\text{ZeroFMR}(t) = \min_{t}\{\text{FNMR}(t) \mid \text{FMR}(t) = 0\},$$
$$\text{ZeroFNMR}(t) = \min_{t}\{\text{FMR}(t) \mid \text{FNMR}(t) = 0\}.$$

Because now the FRR(FNMR) and FAR(FMR) are drawn as 2D surfaces, all locations of FAR points having zero values are determined and the minimum value of the corresponding FRR values at these locations is the ZeroFAR. Also, to calculate the ZeroFAR value, all locations of FRR points having zero values are determined and the minimum value of the corresponding FAR values at these locations is the ZeroFRR.
From Fig. 8, following values are drawn:

$$\text{ZeroFMR} = 0.3167 \text{ at } t_1 = 14 \text{ and } t_2 = 5,$$

$$\text{ZeroFRR} = 0.0316 \text{ at } t_1 = 16 \text{ and } t_2 = 10.$$

5.3.5 Drawing ROC curve

A ROC (Receiving Operating Curve) is given where FNMR is plotted as a function of FMR; the curve is drawn in log-log scales for better comprehension(Maio et al., 2002). To draw the curve in the positive portions of x- and y-axis, FMR and FNMR values are multiplied by 100 before applying the logarithm on them. Fig. 9 shows the ROC curve of the proposed matching algorithm. To get one curve, only one column of the FAR matrix is drawn against one column of the FRR matrix, after multiplying with 100 and applying the logarithm on both. As can be shown, the recognition performance is good by comparison with the curve of a good recognition performance system seen in (O'Gorman, 1998). It is noted that the curve in Fig. 9 is going to the top right portion of the plotting area whereas the good recognition performance curve in (O'Gorman, 1998) is going to the bottom left portion of the

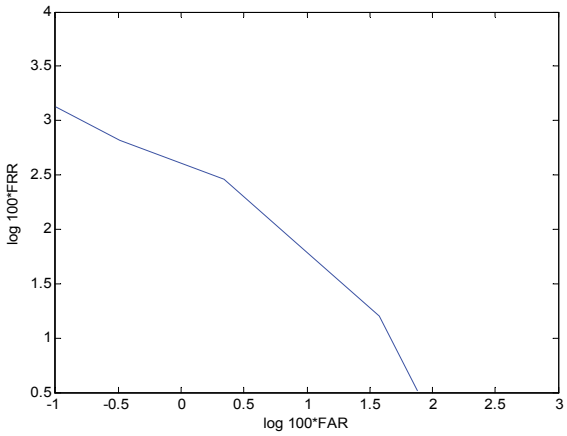

Fig. 9. ROC curve

plotting area, this is because in the proposed matching algorithm, lower scores are associated with matching fingerprints and higher scores are associated with mismatching fingerprints. This is the opposite of most fingerprint matching algorithms in fingerprint verification.

5.3.6 Applying the proposed matching algorithm on FVC2000

Applying the proposed matching algorithm and all above steps in previous sections on the database FVC2000, it is not expected to get good results compared with the results obtained in the previous section. This is due to the reasons mentioned in the beginning of section 5.3. Table 3 and Table 4 show the results of the proposed matching algorithm on FVC2000.

As shown, the recognition accuracy ranges from (1-0.2315 for DB2_B) 77% to (1 – 0.0882 for DB3_B) 91%.

Database	EER	t_1	t_2
DB1_A	0.2109	18	6.99
DB1_B	0.1988	31.68	10
DB2_A	0.1649	18.48	8
DB2_B	0.2315	24.096	14
DB3_A	0.1454	28	12.55
DB3_B	0.0882	28	12.85
DB4_A	0.1815	10	4.88998
DB4_B	0.1206	14.35	9

Table 3. Results of EER and its corresponding thresholds after applying the proposed matching algorithm on FVC2000

Database	FAR	FRR	t_1	t_2
DB1_A	0.2113	0.2105	18	7
DB1_B	0.2049	0.1944	32	10
DB2_A	0.1835	0.15	18	9
DB2_B	0.2403	0.2375	24	15
DB3_A	0.1325	0.1525	29	12
DB3_B	0.0944	0.075	28	13
DB4_A	0.1844	0.1788	10	5
DB4_B	0.1153	0.125	14	10

Table 4. Results of FAR and FRR and their corresponding thresholds after applying the proposed matching algorithm on FVC2000

Fig. 10 and Fig. 11 show the FRR and FAR surfaces at the left side and the ROC curves at the right side for databases DB1_A, DB2_A, DB2_B, DB3_A, and DB4_A respectively.

5.3.7 Calculating average enroll time

The average enroll time is calculated as the average CPU time taken by a single enrolment operation (Maio et al., 2002). The steps of enrolment are discussed in section 5.1. table 5 shows a detailed timing for each step in the enrolment phase. These results were implemented using MATLAB version 7.9.0529 (R2009b) as the programming platform. Programs were tested on a 2.00GHz personal computer with 1.99 GB of RAM.
Total enroll time is found to be 6.043 sec

Step	Average time taken (sec)
Enhancement of the fingerprint	3.7
Core point detection	0.54
Thinning and minutiae extraction	1.8
Minutiae table construction	0.003
Total enroll time	6.043

Table 5. Enroll timing details

5.3.8 Calculating average match time

The average match time is calculated as the average CPU time taken by a single match operation between a template and a fingerprint image (Maio et al., 2002). The steps of matching are discussed in section 5.2. Table 6 shows a detailed timing for each step in the matching phase after the construction of the minutiae table corresponding to the input fingerprint, which has been already estimated to be 6.043 sec from section 5.3.7.
Total match time is found to be 0.00134 sec

Step	Average time taken (sec)
Get all minutiae tables of the claimed fingerprint stored in the database	0.0011
Calculate absolute differences between the input fgp minutiae table and all minutiae tables obtained from the previous step and get the two geometric means	0.0002
Compare the resulting means with the two thresholds and decide if the user is accepted or rejected	0.00004
Total match time	0.00134

Table 6. Match timing details

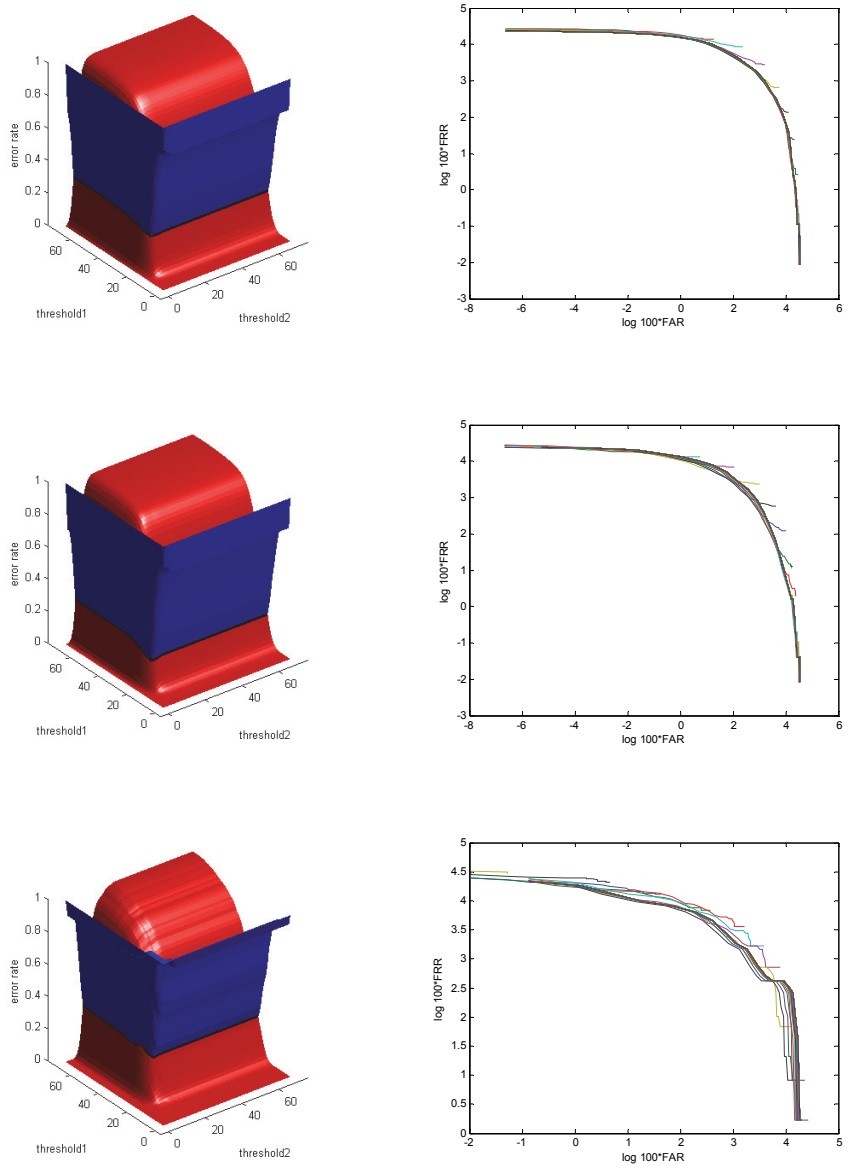

Fig. 10. FAR, FRR and ROC curves for DB1_A, DB2_A, and DB2_B respectively.

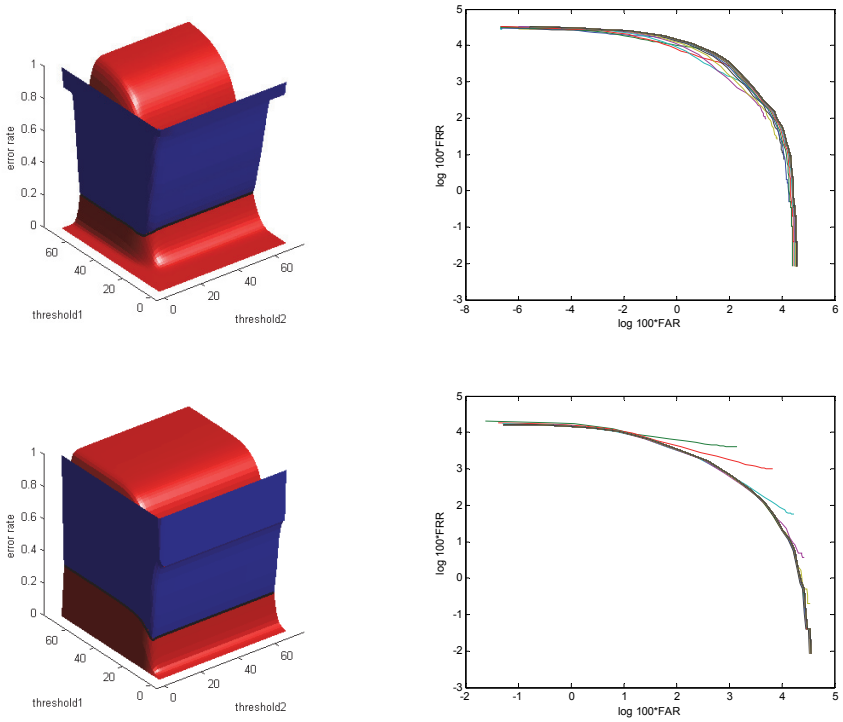

Fig. 11. FAR, FRR and ROC curves for DB3_A, DB4_A respectively.

5.4 Conclusion

As shown, the time for matching is extremely small using our algorithm as all the process is taking geometric mean of absolute differences. There is no need for any pre-alignment which is a very complicated and time consuming process. As a result, our algorithm is translation and rotation invariant.

Also, the space needed to store any minutiae table is in average 21 (as the average number of tracks in all database) × 2 × 4 = 168 bits = 168/8 bytes = 21 bytes which is small in comparison with the size of 85 bytes as in (Jain & Uludag, 2002) where the traditional method is storing locations and orientation for each minutia as a tuple <x, y, θ>.

6. References

Bazen, A. M., & Gerez, S. H. (2003). Fingerprint Matching by Thin-Plate Spline Modelling of Elastic Deformations. *Pattern Recognition*, Vol. 36, pp. 1859-1867

Eckert, G.; M¨uller, S., & Wiebesiek, T. (2005) Efficient Minutiae-Based Fingerprint Matching. *IAPR Conference on Machine VIsion Applications*, pp. 554-557, Tsukuba Science City, Japan, May 16-18

Jain, A.; Hong, L., & Bolle, R. (1997a). On-Line Fingerprint Verification. *IEEE Trans on Pattem Analysis and Machine Intelligence*, Vol. 19, No. 4, pp. 302-313

Jain, A.; Ross, A., & Prabhakar, S. (2001). Fingerprint Matching Using Minutiae and Texture Features. *Proc. Int. Conf. on Image Processing (ICIP)*, pp. 282-285, Thessaloniki, Greece, October 7-10

Jain, A. K.; Hong, L.; Pankanti, S., & Bolle, R. (1997b). An identity authentication system using fingerprints. *Proc. IEEE*, Vol. 85, pp. 1365-1388

Jain, A. K.; Prabhakar, S.; Hong, L., & Pankanti, S. (2000). Filterbank-Based Fingerprint Matching. *IEEE Transactions on Image Processing*, Vol. 9, No. 5, pp. 846-859

Jain, A. K., & Uludag, U. (2002). Hiding Fingerprint Minutiae in Images. *Proc. Workshop on Automatic Identification Advanced Technologies*, pp. 97-102

Ji, L., & Yi, Z. (2008). Fingerprint orientation field estimation using ridge projection. *Pattern Recognition*, Vol. 41, pp. 1491-1503

Jiang, X., & Yau, W. Y. (2000). Fingerprint Minutiae Matching Based on the Local and Global Structures. *in Proc. Int. Conf. on Pattern Recognition (15th)*, Vol. 2, pp. 1042-1045

Jie, Y.; fang, Y.; Renjie, Z., & Qifa, S. (2006). Fingerprint minutiae matching algorithm for real time system. *Pattern Recognition*, Vol. 39, pp. 143-146

Liang, X., & Asano, T. (2006). Fingerprint Matching Using Minutia Polygons. *Proc. Int. Conf. on Pattern Recognition (18th)*, Vol. 1, pp. 1046-1049

Liu, M.; Jiang, X. D., & Kot, A. (2005). Fingerprint reference-point detection. *EURASIP Journal on Applied Signal Processing*, Vol. 4, pp. 498-509

Luo, X.; Tian, J., & Wu, Y. (2000). A Minutia Matching Algorithm in Fingerprint Verification. *15th ICPR Int. Conf. on Pattern Recognition*, pp. 833-836, Barcelona, Spain, September 3-7

Maio, D.; Maltoni, D.; Cappelli, R.; Wayman, J. L., & Jain, A. K. (2002). FVC2000: Fingerprint Verification Competition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, No. 3, pp. 402-412

Maltoni, D.; Maio, D.; Jain, A. K., & Prabhakar, S. (2003). *Handbook of Fingerprint Recognition*, Springer-Verlag New York

Maltoni, D.; Maio, D.; Jain, A. K., & Prabhakar, S. (2009). *Handbook of Fingerprint Recognition*, Springer-Verlag, London

Nilsson, K., & Bigun, J. (2003). Localization of corresponding points in fingerprints by complex filtering. *Pattern Recognition Letters*, Vol. 24, pp. 2135-2144

O'Gorman, L. (1998). An Overview of fingerprint verification technologies. *Elsevier Information Security Technical Report*, Vol. 3, No. 1, pp. 21-32

Ratha, N. K.; Chen, S. Y., & Jain, A. K. (1995). Adaptive Flow Orientation-Based Feature Extraction in Fingerprint Images. *Pattern Recognition*, Vol. 28, No. 11, pp. 1657-1672

Yang, J. C., & Park, D. S. (2008). Fingerprint Verification Based on Invariant Moment Features and Nonlinear BPNN. *International Journal of Control, Automation, and Systems*, Vol. 6, No. 6, pp. 800-808

Yuliang, H.; Tian, J.; Luo, X., & Zhang, T. (2003). Image enhancement and minutiae matching in fingerprint verification. *Pattern Recognition*, Vol. 24, pp. 1349-1360

Zhang, Y.; Tian, J.; Cao, K.; Li, P., & Yang, X. (2008). Improving Efficiency of Fingerprint Matching by Minutiae Indexing. *19th International Conference on Pattern Recognition*, Tampa, FL, December 8-11

Zhu, E.; Yin, J., & Zhang, G. (2005). Fingerprint matching based on global alignment of multiple reference minutiae. *Pattern Recognition*, Vol. 38, pp. 1685-1694