Fingerprint Recognition System

Praveen Shukla[1], Rahul Abhishek[2], Chankit jain[3]
M.Tech (Control & Automation), School of Electrical Engineering, VIT University, Vellore

***Abstract* - Fingerprints are one of the most mature bio-metric technologies and are considered legitimate proofs of evidence in courts of law all over the world. In recent times, more and more civilian and commercial applications are either using or actively considering using fingerprint-based identification because of the availability of inexpensive and compact solid state scanners as well as its superior and proven matching performance over other bio-metric technologies.**

***Index Terms* - MATLAB, Fingerprint**

I. INTRODUCTION

Biometric recognition refers to the use of distinctive physiological (eg. fingerprint, palm print, iris, face) and behavioral (eg. gait, signature) characteristics, called biometric identifiers for recognizing individuals.

Fingerprint recognition is one of the oldest and most reliable biometric used for personal identification. Fingerprint recognition has been used for over 100 years now and has come a long way from tedious manual fingerprint matching. The ancient procedure of matching fingerprints manually was extremely cumbersome and time-consuming and required skilled personnel.

Finger skin is made up of friction ridges and sweat pores all along these ridges. Friction ridges are created during fetal life and only the general shape is genetically defined. The distinguishing nature of physical characteristics of a person is due to both the inherent individual genetic diversity within the human population as well as the random processes affecting the development of the embryo. Friction ridges remain the same throughout one's adult life. They can reconstruct themselves even in case of an injury as long as the injury is not too serious.

Some important terms related to fingerprint identification systems are explained Below:

1. Fingerprint Acquisition: How to acquire fingerprint images and how to represent them in a proper machine-readable format.
2. Fingerprint Verification: To determine whether two fingerprints are from the same finger.
3. Fingerprint Identification: To search for a query fingerprint in a database.
4. Fingerprint Classification: To assign a given fingerprint to one of the prespecified categories according to its geometric characteristics.

In case of both fingerprint identification and fingerprint verification systems, our Tasks will be broken up into 2 stages:

1. *Off-line phase*: Several fingerprint images of the fingerprint of a person to be verified are first captured and processed by a feature extraction module; the extracted features are stored as templates in a database for later use.
2. *On-line phase*: The individual to be verified gives his/her identity (in case of a verification system) and places his/her finger on the inkless fingerprint scanner, minutia points are extracted from the captured fingerprint image.

These minutiae are then fed to a matching module, which matches them against his/her own templates in the database (in case of a verification system) or against all the users in the database (in case of an identification system).

Hence, in case of a fingerprint identification system, we would need to match the incoming fingerprint with stored templates of every other user in the database. In order to reduce this computation and search overhead, it is essential to have some kind of fingerprint classification system which will help us to severely restrict the size of the database for which we need to extract minutiae features and match against the incoming fingerprint sample. Hence, fingerprints can be classified into one of six categories based on their geometric structure at a macro level.

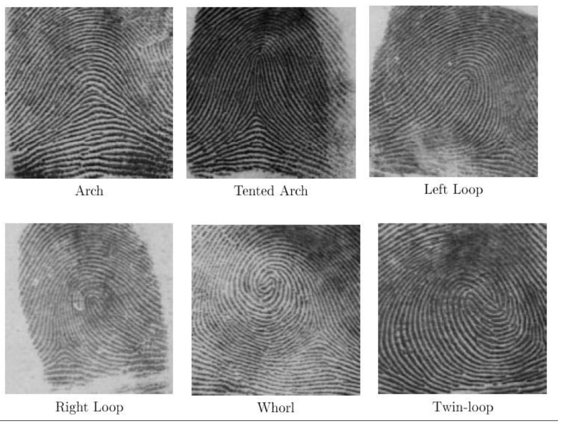


Figure.1 Sample finger print

Hence, by classifying fingerprints on the above basis we can make the database to be searched much smaller. Also, if images from multiple fingers are used, we will have many more categories and the size of the database to be searched will decrease further.

1.1 Fingerprint matching techniques

The large number of approaches to fingerprint matching can be coarsely classified into three families.

- **Correlation-based matching**: Two fingerprint images are superimposed and the correlation between corresponding pixels is computed for different alignments (e.g. various displacements and rotations).
- **Minutiae-based matching**: This is the most popular and widely used technique, being the basis of the fingerprint comparison made by fingerprint examiners. Minutiae are extracted from the two fingerprints and stored as sets of points in the two- dimensional plane. Minutiae-based matching essentially consists of finding the alignment between the template and the input minutiae sets that results in the maximum number of minutiae pairings
- **Pattern-based (image-based) matching**: Pattern based algorithms compare the basic fingerprint patterns (arch, whorl, and loop) between a previously stored template and a candidate fingerprint. This requires that the images be aligned in the same orientation. To do this, the algorithm finds a central point in the fingerprint image and centers on that. In a pattern-based algorithm, the template contains the type, size, and orientation of patterns within the aligned fingerprint image. The candidate fingerprint image is graphically compared with the template to determine the degree to which they match.

In Our project we have implemented a minutiae based matching technique. This approach has been intensively studied, also is the backbone of the current available fingerprint recognition products.

II. Theoretical Overview and Implementation Details

2.1 Fingerprint Database Used

In order to test the validity of our implementation, we have used the FVC2002 fingerprint image database. FVC2002 was the Second International Competition for Fingerprint Verification Algorithms and we acquired the database that was provided to the participants of this competition. The images used were online images of a reasonably good quality. Most parts of most of the fingerprint images were recoverable and did not produce too many spurious minutiae after enhancement and thinning.
The details of the fingerprint database used (*db1_b.zip*) by us are as following Table 1

Table 1.Database used

Sensor Type	Image Size	Size of Set	Resolution
Optical Sensor: “Touch View II” by Identix	388x374 (142 k pixels)	10 users x 8 fingerprints per user	500 dpi

This fingerprint database was downloaded from *bias.csr.unibo.it/fvc2002/download.asp*

2.2 Ridge Orientation

Ridge orientation is the process of obtaining the angle of the ridges throughout the image. Ridge orientations are calculated on a block-basis for a WxW block. W is generally equal to 16.

The first step in ridge orientation is the calculation of gradients at each pixel of the image. The gradients Gx(i,j) and Gy(i,j) are calculated using the Sobel Operator. The Sobel operator is given as follows:

$$Gx(i,j)=((A2+K\,A3+A4)-(A0+K\,A7+A6))/(K+2)$$
$$Gy(i,j)=((A0+K\,A1+A2)-(A6+K\,A5+A4))/(K+2)$$

Where K=2 for the Sobel Operator. The meanings of the pixels Ai are given as follows:

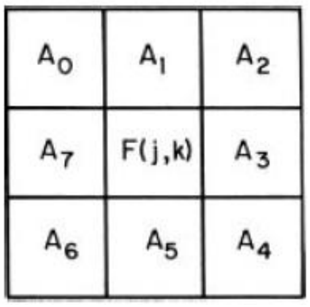


Fig. 2 Ridge orientation Ai

The next step in ridge orientation involves using the gradient values of all these pixels to calculate the ridge angle for a W*W block as follows:

$$\theta_o = \frac{1}{2}\tan^{-1}\left(\frac{\sum_{i=1}^{W}\sum_{j=1}^{W} 2G_x(i,j)G_y(i,j)}{\sum_{i=1}^{W}\sum_{j=1}^{W}\left(G_x^2(i,j) - G_y^2(i,j)\right)}\right)$$

Due to the presence of noise, corrupted ridge and valley structures, minutiae, etc. in the input image, the estimated local ridge orientation, *q*(*i*, *j*), may not always be correct. Since local ridge orientation varies slowly in a local neighborhood where no singular points appear, a low-pass filter can be used to modify the incorrect local ridge orientation. In order to perform the low-pass filtering, the orientation image needs to be converted into a *continuous vector field*, which is defined as follows:

$$\Phi_x(i,j) = \cos(2\theta(i,j)),$$

$$\Phi_y(i,j) = \sin(2\theta(i,j)),$$

This continuous vector field is passed through a low-pass Gaussian type filter in order to get the improved the orientation image. Finally, the local orientation is calculated from the filtered vector field by using the following formula:

$$O(i,j) = \frac{1}{2}\tan^{-1}\left(\frac{\Phi'_y(i,j)}{\Phi'_x(i,j)}\right)$$

2.3Image Enhancement

Implications of poor quality image:

- A significant number of spurious minutiae may be created,
- A large percent of genuine minutiae may be ignored, and
- Large errors in their localization (position and orientation) may be introduced.

Enhancement algorithm improves the clarity of ridge and valley structures in the fingerprint images. Following steps constitute the enhancement algorithm:

1. **Normalization**: An input image is normalized, so that the ridges and valleys are easily distinguished.
2. **Local Orientation Estimation**: From the normalized fingerprint image orientation of ridges are calculated in each block of desired size.
3. **Frequency Estimation**: Using the normalized image and the Orientation image the frequency in each of the blocks of the image is calculated.
4. **Filtering**: A bank of Gabor filter tuned to local ridge orientation and frequency is used to filter the image to clearly separate out ridges and valleys and thus reduce the probability of spurious minutiae.

2.4 Normalization

For normalization we have done simple histogram equalization, which enhances the contrast of images by transforming the values in the fingerprint image.

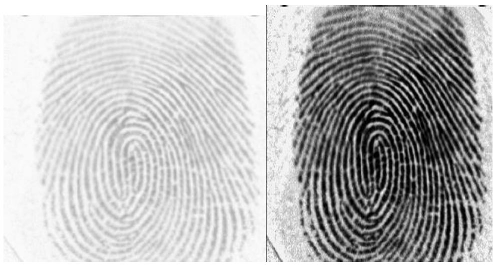


Fig 3 normalized image

2.5 Image Binarization

Image Binarization is a process which transforms the 8-bit Gray image to a 1-bit image with 0-value for ridges and 1-value for furrows. After the operation, ridges in the fingerprint are highlighted with black color while furrows are white.

A locally adaptive binarization method is performed to binarize the fingerprint image. In this method image is divided into blocks of 16 x 16 pixels. A pixel value is then set to 1 if its value is larger than the mean intensity value of the current block to which the pixel belongs (Figure 3.4).

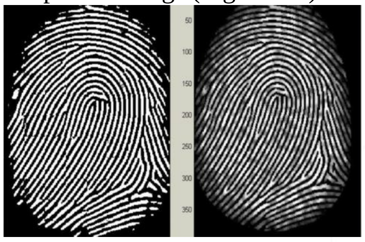


Figure.4 Binarized Image after FFT, (b) Image before Binarization.

Flowchart for Binarization

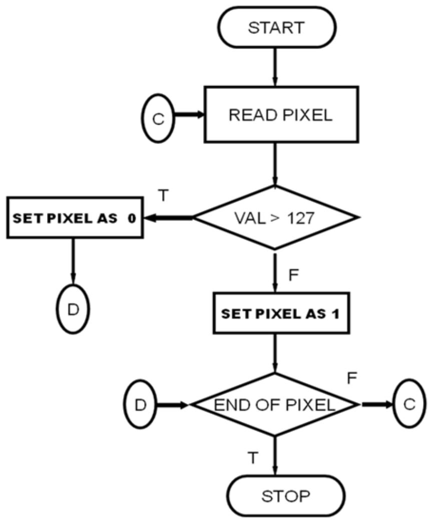


Figure.6 Flow Chart

2.6 Filtering

The sinusoidal-shaped waves of ridges and valleys vary slowly in a local constant orientation. Therefore, a band-pass filter that is tuned to the corresponding frequency and orientation can efficiently remove the undesired noise and preserve the true ridge and valley structures. Gabor filters have both frequency-selective and orientation selective properties and have optimal joint resolution in both spatial and frequency domains. Therefore, it is appropriate to use Gabor filters as band-pass filters to remove the noise and preserve true ridge/valley structures.

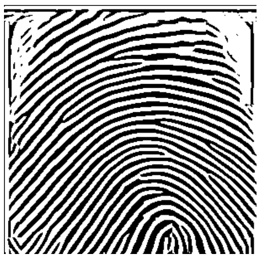


Figure.6 Image after the Gabor filter and Binarization

The structure of the Gabor filters is as follows:

$$G(x, y; \theta, f) = \exp\left\{-\frac{1}{2}\left[\frac{x_\theta^2}{\sigma_x^2} + \frac{y_\theta^2}{\sigma_y^2}\right]\right\}\cos(2\pi f x_\theta),$$

$$x_\theta = x\cos\theta + y\sin\theta,$$

$$y_\theta = -x\sin\theta + y\cos\theta,$$

In our implementation, the frequency f required by the Gabor filter is assumed to be constant and the angular inputs are provided from the orientation map. This is where the orientation map is useful. A frequency value of 1/7 was found to give satisfactory results. In the spatial domain, the Gabor filter looks like the following:

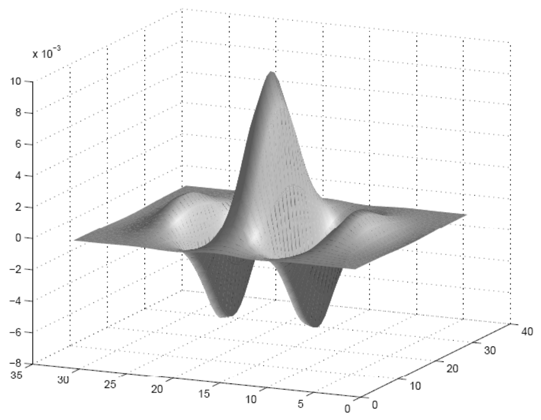


Figure .7 Gabor filter

In order to get the output image as shown in the enhanced image above, we need to carry out binarization of the image. This is done by using the inbuilt MATLAB function *im2bw* by supplying it a value of 0.3 for the threshold. This value was found from trial and error.

2.7 Thinning

The second class of sequential thinning algorithms is parallel. In parallel thinning algorithms the decision for individual pixel deletion is based on the results of the previous iteration. Like sequential algorithms, parallel thinning usually considers a 3*3 neighborhood around the current pixel. A set of rules for deletion is applied based on pixels in the neighborhood. Fully parallel algorithms have trouble maintaining connectedness, so they are often broken into sub-iterations where only subsets of the pixels are considered for deletion. Non-iterative thinning methods are not based on examining individual pixels. Some popular non-pixel based methods include medial axis transforms, distance transforms, and determination of centerlines by line following.

In line following methods, midpoints of black spaces in the image are determined and then joined to form a skeleton. This is fast to compute but tends to produce noisy skeletons. It has been conjectured that human beings naturally perform thinning in a manner similar to this.

Another method of centerline determination is by following contours of objects. By simultaneously following contours on either side of the object a continual centerline can be computed. The skeleton of the image is formed from these connected centerlines.

Medial axis transforms often use gray-level images where pixel intensity represents distance to the boundary of the object. The pixel intensities are calculated using distance transforms. In Figure below the maximum pixel intensity would increase toward the dark lines at the centers of the circles. Note that there are other methods of computing medial axis transforms.

0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	2	2	2	2	1	0
0	1	2	3	3	2	1	0
0	1	2	2	2	2	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

Figure.8 thinning matrix

2.8 Distance Transformation

In order to explain the thinning algorithm used, we will explain it step by step for a better understanding. The pseudo code of the algorithm uses the following variables. I = the original binary image in which black pixels are “0” and white pixels are “1”. The object in the image is made of connected white pixels. J and K are temporary images for use in each iteration of the algorithm. J is the (n-1)th iteration output and K is the current, or nth, iteration output. P(i) is the current pixel under consideration.

Table 2. the 8-Neighborhood of Pixel P1

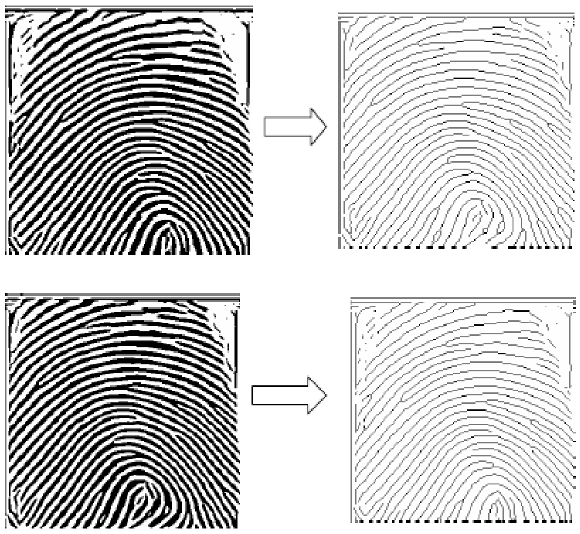
P9 (i-1,j-1)	P2 (i-1,j)	P3 (i-1,j+1)
P8 (i,j-1)	P1 (i,j)	P4 (i,j+1)
P7 (i+1,j-1)	P6 (i+1,j)	P5 (i+1,j+1)

Two other variables used in the algorithm are A and B. A of pixel P1 is defined as: SUM(P2…P9) . B of pixel P1 is the count of the number of 0 to 1 transitions in a clockwise circle from P9 back to itself. For example suppose we have the pixels shown in following Figure. B would equal 2 since there are two 0 to 1 transitions in a clockwise order.

0	0	1
0	P1(i,j)	1
1	0	1

Table 3. Calculation of B

The algorithm runs in 2 sub-iterations. During each sub-iteration different rules are applied for deciding whether or not a pixel will be deleted.



1) Minutiae Extraction

Our implementation of fingerprint identification and verification is based the topological structural matching of minutiae points. We only consider two kinds of minutiae; ridge endings and bifurcations as shown in the following figure:

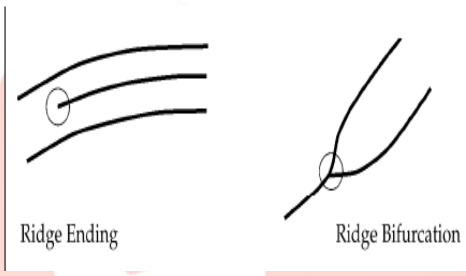
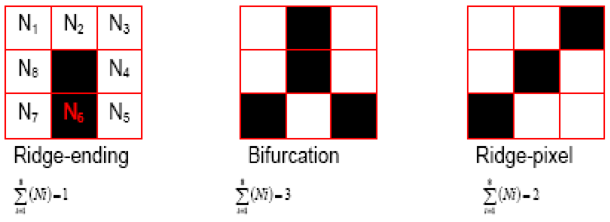


Figure. 9 Extraction

Minutiae extraction from a perfectly thinned ridge-map of a fingerprint image is a trivial task. All we need to do is to count the number of ridge pixels, every ridge pixel on the thinned image is surrounded by and depending on the following rule, and we can assign the minutiae points to those pixels:



However, due to noise, limitation on image acquisition, skin deformations etc the fingerprint image obtained is never ideal. As a result there are a lot of spurious minutiae that crop up if we simply follow the above approach to minutiae detection. To solve the problem, various heuristics have been proposed and we have implemented the following rules to remove most of the spurious minutiae, resulting from noise in the thinned image:

- If several minutiae form a cluster in a small region, then remove all of them except for the one nearest to the cluster center
- If two minutiae are located close enough, facing each other, but no ridges lie between them, then remove both of them

In addition to the noise in the fingerprint image, the thinned image may not be ideal. If such is the case, minutiae extraction may not yield the correct results.

2) Minutiae Matching

Minutiae matching is the step which comes after minutiae extraction and it is here that we match the minutiae obtained from two sample fingerprint images and test whether they are from the same fingerprint or not.

However, a crucial step that needs to be carried out before we can use brute force and match minutiae on two images is alignment of the images. Alignment is necessary so that we correctly match the images. We also need to take care of difference in positioning of minutiae due to plastic deformations in the finger. The algorithms prevalent for minutiae-matching either include the use of details of ridges on which minutiae are present, or use the Hough transform. Both these methods and most other methods are difficult to implement and several complicated functions need to be implemented.

Hence, we decided to implement a minutiae matching algorithm which was inspired by the techniques involving computation of local and global minutiae features.

Our algorithm grouped all the minutiae into triplets of minutiae. For each of these triplets of minutiae we stored the distance of one of the minutiae from both other minutiae and the angle formed in between these two distances.

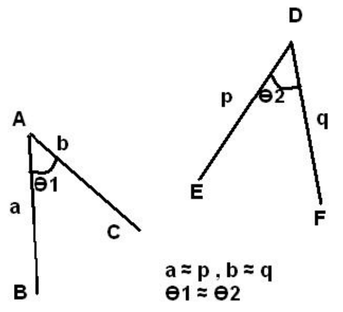


Figure.10 angle formed in between two distances

Hence, when we have n minutiae points in our image, we have n(n-1)(n-1)/6 minutiae triplets. Hence, we need to compute and store these triplets for both the incoming image and the stored image.

Next, the two tables containing the minutiae triplets are matched using brute force method and the number of matching triplets is computed.

While this algorithm is easy to implement and does not require the use of image rotation, it is computationally very intensive for form all the minutiae triplets and to then use a brute-force method to try and search for matching triplets. The process takes too much time and the current implementation is not very well suited for real time applications to match two minutiae tables. Hence, the algorithm cannot be used for real-time applications in its present form.

Another disadvantage with the algorithm is that it is difficult to predict with reasonable certainty the order of time the algorithm will take to match two fingerprints. This is because the number of minutiae triplets is O(n3) where n is the number of bifurcation minutiae in the image. Hence, if this number increases to a very large number due to spurious minutiae, in that case the matching step may take very long. Also, in the present implementation we are using any information provided by the ridge endings. This was done in order to reduce the computation time. Hence, by using a more efficient matching algorithm we can make use of the information provided by ridge endings. However, it must be noted that in case of low quality images or images from a dry finger etc. there is a higher probability of getting spurious ridge endings rather than spurious bifurcation minutiae. Hence, we chose to take only bifurcation minutiae for our matching process.

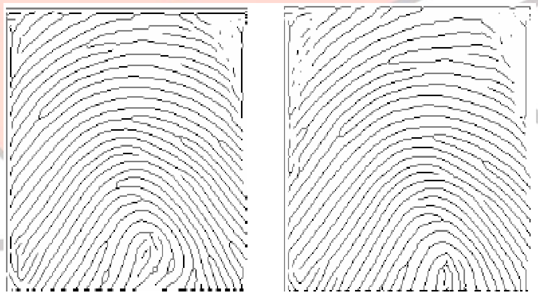


Fig. 11 these images were correctly detected as a match by the matching algorithm

An elastic string (,,) match algorithm is used to find number of matched minutia pairs among I1' & I2'.

According to the elastic string match algorithm minutia mi in I1' and a minutia mj in I2' are considered "matching," if the spatial distance (sd) between them is smaller than a given tolerance r0 and the direction difference (dd) between them is smaller than an angular tolerance ϴ0.

$$sd = \leq r_0$$
$$dd = \leq \Theta_0$$

Let mm(.) be an indicator function that returns 1 in the case where the minutiae m_i and m_j match according to above equations.

$$mm(m_i,m_j)=\begin{cases}1, & sd(m_i,m_j) \leq r_0 \text{ and } dd(m_i,m_j) \leq \theta_0 \\ 0, & otherwise\end{cases}$$

Now the total number of matched minutiae pair given by,
num (matched minutiae) = $\sum \mathrm{mm}(m_i,m_j)$

and final match score is given by,Match Score = $\dfrac{num\ (matched\ minutiae)}{\max\ (num\ of\ minutiae\ in\ I_1,I_2\ )}$

III. System Level Design

A fingerprint recognition system constitutes of fingerprint acquiring device, minutia extractor and minutia matcher [Figure 12].

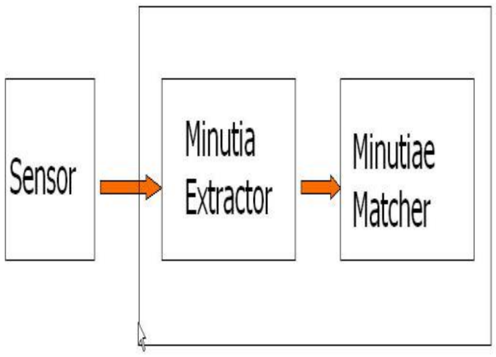


Figure.12 Simplified Fingerprint Recognition System

For fingerprint acquisition, optical or semi-conduct sensors are widely used. They have high efficiency and acceptable accuracy except for some cases that the user's finger is too dirty or dry. However, the testing database for my project is from the available fingerprints provided by FVC2002 (Fingerprint Verification Competition 2002). So no acquisition stage is implemented.

The minutia extractor and minutia matcher modules are explained in detail in the next part for algorithm design and other subsequent sections.

IV. Algorithm Level Design

To implement a minutia extractor, a three-stage approach is widely used by researchers. They are preprocessing, minutia extraction and post processing stage [Figure 13].

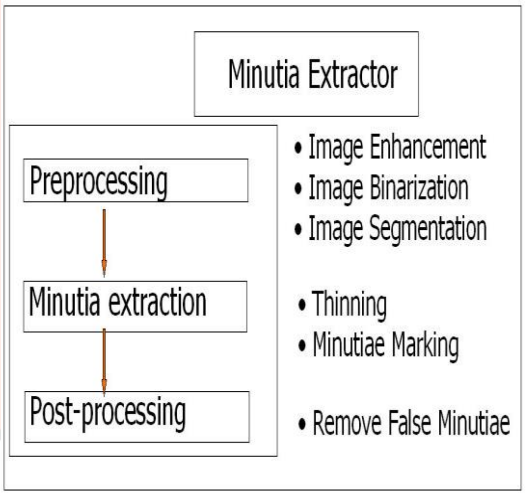
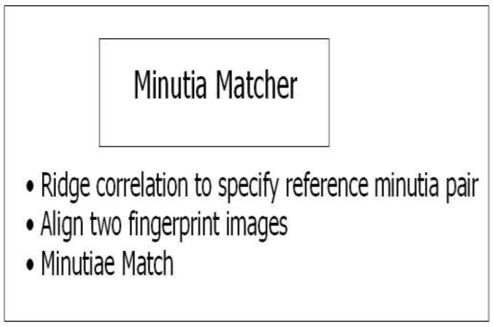


Figure 13 Minutia Extractor

For the fingerprint image preprocessing stage, I use Histogram Equalization and Fourier Transform to do image enhancement . And then the fingerprint image is binarized using the locally adaptive threshold method . The image segmentation task is fulfilled by a three-step approach: block direction estimation, segmentation by direction intensity and Region of Interest extraction by Morphological operations. Most methods used in the preprocessing stage are developed by other researchers but they form a brand new combination in my project through trial and error. Also the morphological operations for extraction ROI are introduced to fingerprint image segmentation by me.

For minutia extraction stage, three thinning algorithms are tested and the Morphological thinning operation is finally bid out with high efficiency and pretty good thinning quality. The minutia marking is a simple task as most literatures reported but one special case is found during my implementation and an additional check mechanism is enforced to avoid such kind of oversight.
For the post processing stage, a more rigorous algorithm is developed to remove false minutia based on. Also a novel representation for bifurcations is proposed to unify terminations and bifurcations.



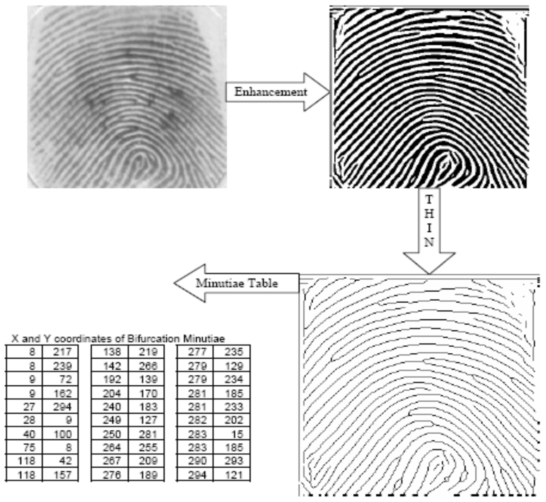
Minutia Matcher

- Ridge correlation to specify reference minutia pair
- Align two fingerprint images
- Minutiae Match

Figure 14 Minutia Matcher

The minutia matcher chooses any two minutia as a reference minutia pair and then match their associated ridges first. If the ridges match well, two fingerprint images are aligned and matching is conducted for all remaining minutia [Figure 14].

V. RESULTS



X and Y coordinates of Bifurcation Minutiae

8	217	138	219	277	235
8	239	142	266	279	129
9	72	192	139	279	234
9	162	204	170	281	185
27	294	240	183	281	233
28	9	249	127	282	202
40	100	250	281	283	15
75	8	264	255	283	185
118	42	267	209	290	293
118	157	276	189	294	121

Figure.15 Result of each step

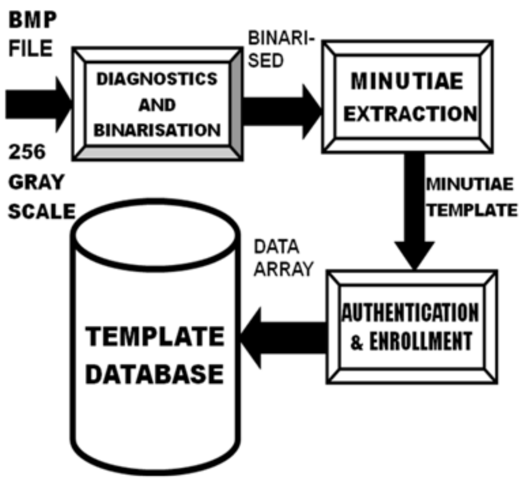


Figure.16: Overall block diagram

VI. CONCLUSION

Paper has combined many methods to build a minutia extractor and a minutia matcher. The combination of multiple methods comes from a wide investigation into research paper. Also some novel changes like segmentation using Morphological operations, minutia marking with special considering the triple branch counting, minutia unification by decomposing a branch into three terminations, and matching in the unified x-y coordinate system after a two-step transformation are used in my project, which are not reported in other literatures I referred to.

Also a program coding with MATLAB going through all the stages of the fingerprint recognition is built. It is helpful to understand the procedures of fingerprint recognition. And demonstrate the key issues of fingerprint recognition.

The above implementation was an effort to understand how Fingerprint Recognition is used as a form of biometric to recognize identities of human beings. It includes all the stages from minutiae extraction from fingerprints to minutiae matching which generates a match score. Various standard techniques are used in the intermediate stages of processing.

The relatively low percentage of verification rate as compared to other forms of biometrics indicates that the algorithm used is not very robust and is vulnerable to effects like scaling and elastic deformations. Various new techniques and algorithm have been found out which give better results.

Also a major challenge in Fingerprint recognition lies in the pre processing of the bad quality of fingerprint images which also add to the low verification rate.

REFERENCES

[1] The Handbook of Fingerprint Recognition. Davide Maltoni, Dario Maio, Anil K Jain, Salil Prabhakar; Springer; 2003

[2] A. Jain and L. Hong, On-line Fingerprint Verification, Proc. 13th ICPR, Vienna, pp. 596-600, 1996.

[3] T. Jea and V. Govindaraju, "A Minutia-Based Partial Fingerprint Recognition System", Pattern Recognition 2005

[4] A. K. Jain, S. Prabhakar, and S. Pankanti, "Twin Test: On Discriminability of Fingerprints, " Proc. 3rd International Conference on Audio- and Video-Based Person Authentication, pp. 211-216, Sweden, June 6-8

[5] Raymond Thai, 'Fingerprint Image Enhancement and Minutiae-Extraction," Thesis submitted to School of Computer Science and Software Engineering, University of Western Australia

[6] Kovacs-Vajna, Z.M., "A fingerprint verification system based on triangular matching and dynamic time warping", IEEE Trans. Patt. Anal. Machine Intell., vol. 22, no. 11, pp. 1266 -1276, Nov.2000.

[7] Ratha, N.K.; Bolle, R.M.; Pandit, V.D.; Vaish, V., "Robust fingerprint authentication using local structural similarity", *Applications of Computer Vision*, 2000, Fifth IEEE Workshop on. , 2000 Page(s): 29 –34

[8] AK Jain, A. Ross, and S. Prabhakar, Fingerprint Matching Using Minutiae and Texture Features , Proc. of International Conference on Image Processing, 2001