Student Attendance System Based On Fingerprint Recognition and One-to-Many Matching

---

A thesis submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology
in
Computer Science and Engineering
by

Rishabh Mishra
(Roll no. 107cs016)
and
Prashant Trivedi
(Roll no. 107cs015)

Under the guidance of :
Prof. B. Majhi

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Orissa, India

Dedicated to

Our Parents

and

Indian Scientific Community

National Institute of Technology Rourkela

Certificate

This is to certify that the project entitled, '**Student Attendance System Based On Fingerprint Recognition and One-to-Many Matching**' submitted by **Rishabh Mishra** and **Prashant Trivedi** is an authentic work carried out by them under my supervision and guidance for the partial fulfillment of the requirements for the award of **Bachelor of Technology Degree** in **Computer Science and Engineering** at **National Institute of Technology, Rourkela**.

To the best of my knowledge, the matter embodied in the project has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date - 9/5/2011
Rourkela

(Prof. B. Majhi)
Dept. of Computer Science and Engineering

Abstract

Our project aims at designing an student attendance system which could effectively manage attendance of students at institutes like NIT Rourkela. Attendance is marked after student identification. For student identification, a fingerprint recognition based identification system is used. Fingerprints are considered to be the best and fastest method for biometric identification. They are secure to use, unique for every person and does not change in one's lifetime. Fingerprint recognition is a mature field today, but still identifying individual from a set of enrolled fingerprints is a time taking process. It was our responsibility to improve the fingerprint identification system for implementation on large databases e.g. of an institute or a country etc. In this project, many new algorithms have been used e.g. gender estimation, key based one to many matching, removing boundary minutiae. Using these new algorithms, we have developed an identification system which is faster in implementation than any other available today in the market. Although we are using this fingerprint identification system for student identification purpose in our project, the matching results are so good that it could perform very well on large databases like that of a country like India (MNIC Project).

This system was implemented in Matlab10, Intel Core2Duo processor and comparison of our one to many identification was done with existing identification technique i.e. one to one identification on same platform. Our matching technique runs in O(n+N) time as compared to the existing O(Nn^2). The fingerprint identification system was tested on FVC2004 and Verifinger databases.

Acknowledgments

We express our profound gratitude and indebtedness to **Prof. B. Majhi**, Department of Computer Science and Engineering, NIT, Rourkela for introducing the present topic and for their inspiring intellectual guidance, constructive criticism and valuable suggestion throughout the project work.

We are also thankful to **Prof. Pankaj Kumar Sa , Ms. Hunny Mehrotra** and other staffs in Department of Computer Science and Engineering for motivating us in improving the algorithms.

Finally we would like to thank our parents for their support and permitting us stay for more days to complete this project.

Date - 9/5/2011
Rourkela

Rishabh Mishra

Prashant Trivedi

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Problem Statement

Designing a student attendance management system based on fingerprint recognition and faster one to many identification that manages records for attendance in institutes like NIT Rourkela.

1.2 Motivation and Challenges

Every organization whether it be an educational institution or business organization, it has to maintain a proper record of attendance of students or employees for effective functioning of organization. Designing a better attendance management system for students so that records be maintained with ease and accuracy was an important key behind motivating this project. This would improve accuracy of attendance records because it will remove all the hassles of roll calling and will save valuable time of the students as well as teachers. Image processing and fingerprint recognition are very advanced today in terms of technology. It was our responsibility to improve fingerprint identification system. We decreased matching time by partitioning the database to one-tenth and improved matching using key based one to many matching.

1.3 Using Biometrics

Biometric Identification Systems are widely used for unique identification of humans mainly for verification and identification. Biometrics is used as a form of identity access management and access control. So use of biometrics in student attendance management system is a secure approach. There are many types of biometric systems like fingerprint recognition, face recognition, voice recognition, iris recognition, palm recognition etc. In this project, we used fingerprint recognition system.

1.4 What is fingerprint?

A fingerprint is the pattern of ridges and valleys on the surface of a fingertip. The endpoints and crossing points of ridges are called minutiae. It is a widely accepted assumption that the minutiae pattern of each finger is unique and does not change during one's life. Ridge endings are the points where the ridge curve terminates, and bifurcations are where a ridge splits from a single path to two paths at a Y-junction. Figure 1 illustrates an example of a ridge ending and a bifurcation. In this example, the black pixels correspond to the ridges, and the white pixels correspond to the valleys.

Figure 1.1: Example of a ridge ending and a bifurcation

When human fingerprint experts determine if two fingerprints are from the same finger, the matching degree between two minutiae pattern is one of the most important factors. Thanks to the similarity to the way of human fingerprint experts and compactness of templates, the minutiae-based matching method is the most widely studied matching method.

1.5 Why use fingerprints?

Fingerprints are considered to be the best and fastest method for biometric identification. They are secure to use, unique for every person and does not change in one's lifetime. Besides these, implementation of fingerprint recognition system is cheap, easy and accurate up to satisfiability.
Fingerprint recognition has been widely used in both forensic and civilian applications. Compared with other biometrics features , fingerprint-based biometrics is the most proven technique and has the largest market shares . Not only it is faster than other techniques but also the energy consumption by such systems is too less.

1.6 Using fingerprint recognition system for attendance management

Managing attendance records of students of an institute is a tedious task. It consumes time and paper both. To make all the attendance related work automatic and on-line, we have designed an attendance management system which could be implemented in NIT Rourkela. It uses a fingerprint identification system developed in this project. This fingerprint identification system uses existing as well as new techniques in fingerprint recognition and matching. A new one to many matching algorithm for large databases has been introduced in this identification system.

1.7 Organization of the thesis

This thesis has been organized into ten chapters. **Chapter 1** introduces with our project. **Chapter 2** explains the proposed design of attendance management system. **Chapter 3** explains the fingerprint identification system used in this project.
Chapter 4 explains enhancement techniques, **Chapter 5** explains feature extraction methods, **Chapter 6** explains our database partitioning approach .
Chapter 7 explains matching technique. **Chapter 8** explains experimental work done and performance analysis. **Chapter 9** includes conclusions and **Chapter 10** introduces proposed future work.

Chapter 2

Attendance Management Framework

Manual attendance taking and report generation has its limitations. It is well enough for 30-60 students but when it comes to taking attendance of students large in number, it is difficult. For taking attendance for a lecture, a conference, etc. roll calling and manual attendance system is a failure. Time waste over responses of students, waste of paper etc. are the disadvantages of manual attendance system. Moreover, the attendance report is also not generated on time. Attendance report which is circulated over NITR webmail is two months old. To overcome these non-optimal situations, it is necessary that we should use an automatic on-line attendance management system. So we present an implementable attendance management framework. Student attendance system framework is divided into three parts : Hardware/Software Design, Attendance Management Approach and On-line Report Generation. Each of these is explained below.

2.1 Hardware - Software Level Design

Required hardware used should be easy to maintain, implement and easily available. Proposed hardware consists following parts:
(1)Fingerprint Scanner,
(2)LCD/Display Module (optional),
(3)Computer

Table 2.1: Estimated Budget

Device Name	Cost of One Unit	Number of Units Required	Total Unit Budget
Scanner	500	100	50000
PC	21000	100	2100000
Total			21,50,000

(4)LAN connection

Fingerprint scanner will be used to input fingerprint of teachers/students into the computer software. LCD display will be displaying rolls of those whose attendance is marked. **Computer Software** will be interfacing fingerprint scanner and LCD and will be connected to the network. It will input fingerprint, will process it and extract features for matching. After matching, it will update database attendance records of the students.

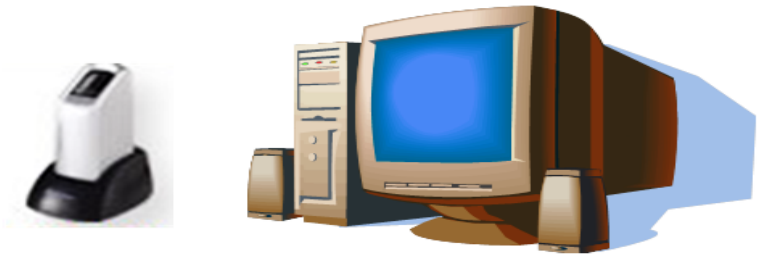

Figure 2.1: Hardware present in classrooms

Estimated Budget Estimated cost of the hardware for implementation of this system is shown in the table 2.1. Total number of classrooms in NIT Rourkela is around 100. So number of units required will be 100.

2.2 Attendance Management Approach

This part explains how students and teachers will use this attendance management system. Following points will make sure that attendance is marked correctly, without any problem:

(1)All the hardware will be inside classroom. So outside interference will be absent.

(2)To remove unauthorized access and unwanted attempt to corrupt the hardware by students, all the hardware except fingerprint scanner could be put inside a small

cabin. As an alternate solution, we can install CCTV cameras to prevent unprivileged activities.

(3)When teacher enters the classroom, the attendance marking will start. Computer software will start the process after inputting fingerprint of teacher. It will find the Subject ID, and Current Semester using the ID of the teacher or could be set manually on the software. If teacher doesn't enter classroom, attendance marking will not start.

(4)After some time, say 20 minutes of this process, no attendance will be given because of late entrance. This time period can be increased or decreased as per requirements.

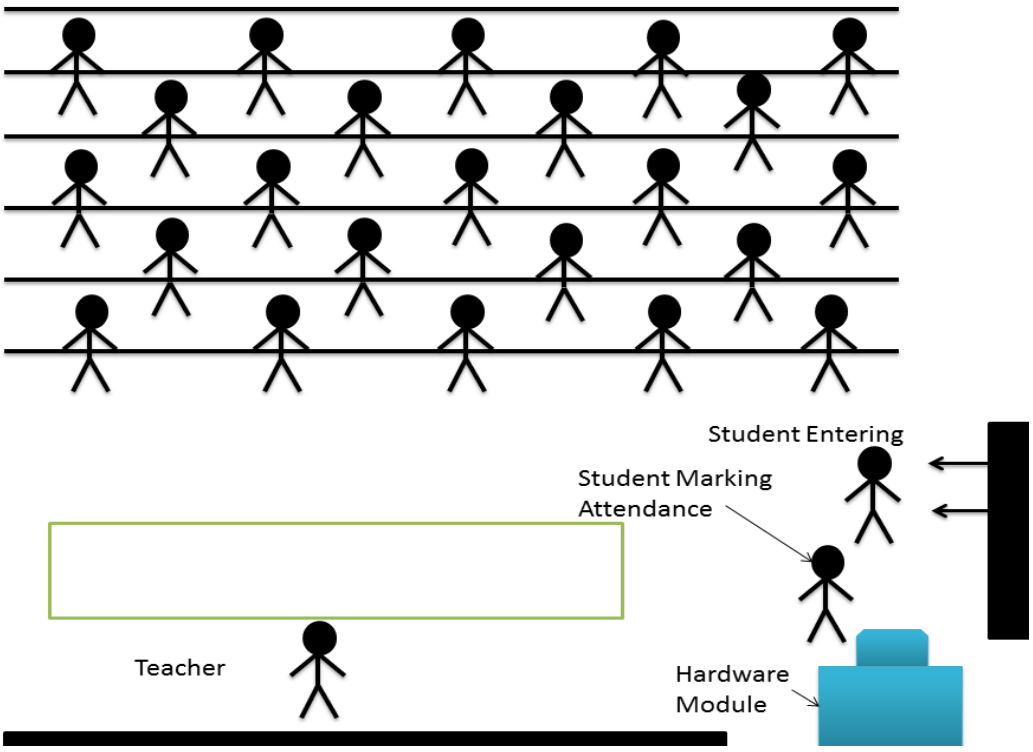

Figure 2.2: Classroom Scenario

2.3 On-Line Attendance Report Generation

Database for attendance would be a table having following fields as a combination for primary field: (1)Day,(2)Roll,(3)Subject and following non-primary fields: (1)Attendance,(2)Semester. Using this table, all the attendance can be managed for a student. For on-line report generation, a simple website can be hosted on NIT Rourkela servers,

which will access this table for showing attendance of students. The sql queries will be used for report generation. Following query will give total numbers of classes held in subject CS423:

SELECT COUNT(DISTINCT Day) FROM AttendanceTable WHERE SUBJECT = CS423 AND Attendance = 1

For attendance of roll 107CS016, against this subject, following query will be used:

SELECT COUNT(Day) FROM AttendanceTable WHERE Roll = 107CS016 AND SUBJECT = CS423 AND Attendance = 1

Now the attendance percent can easily be calculated :

$$Attendance = \frac{ClassesAttended}{ClassesHeld} * 100 \tag{2.1}$$

2.4 Network and Database Management

This attendance system will be spread over a wide network from classrooms via intranet to internet. Network diagram is shown in fig. 2.3. Using this network, attendance reports will be made available over internet and e-mail. A monthly report will be sent to each student via email and website will show the updated attendance. Entity relationship diagram for database of students and attendance records is shown in fig. 2.4. In ER diagram, primary fields are Roll, Date, SubjectID and TeacherID. Four tables are Student, Attendance, Subject and Teacher. Using this database, attendance could easily be maintained for students. Dataflow is shown in data flow diagrams (DFD) shown in figures 2.5, 2.6 and 2.7.

2.5 Using wireless network instead of LAN and bringing portability

We are using LAN for communication among servers and hardwares in the classrooms. We can instead use wireless LAN with portable devices. Portable device will have an embedded fingerprint scanner, wireless connection, a microprocessor loaded with a software, memory and a display terminal, see figure 2.5. Size of device could be small like a mobile phone depending upon how well the device is manufactured.

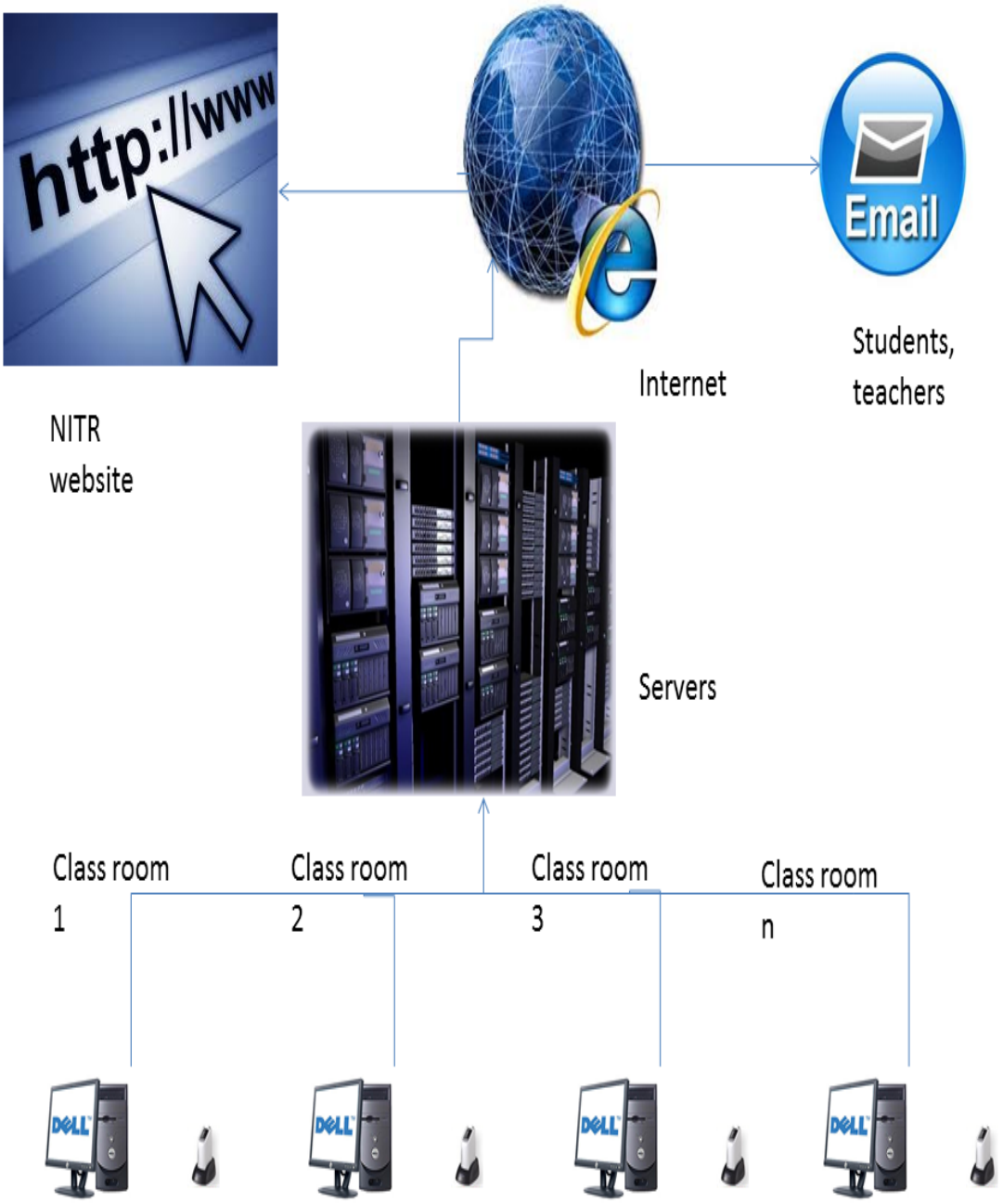

Figure 2.3: Network Diagram

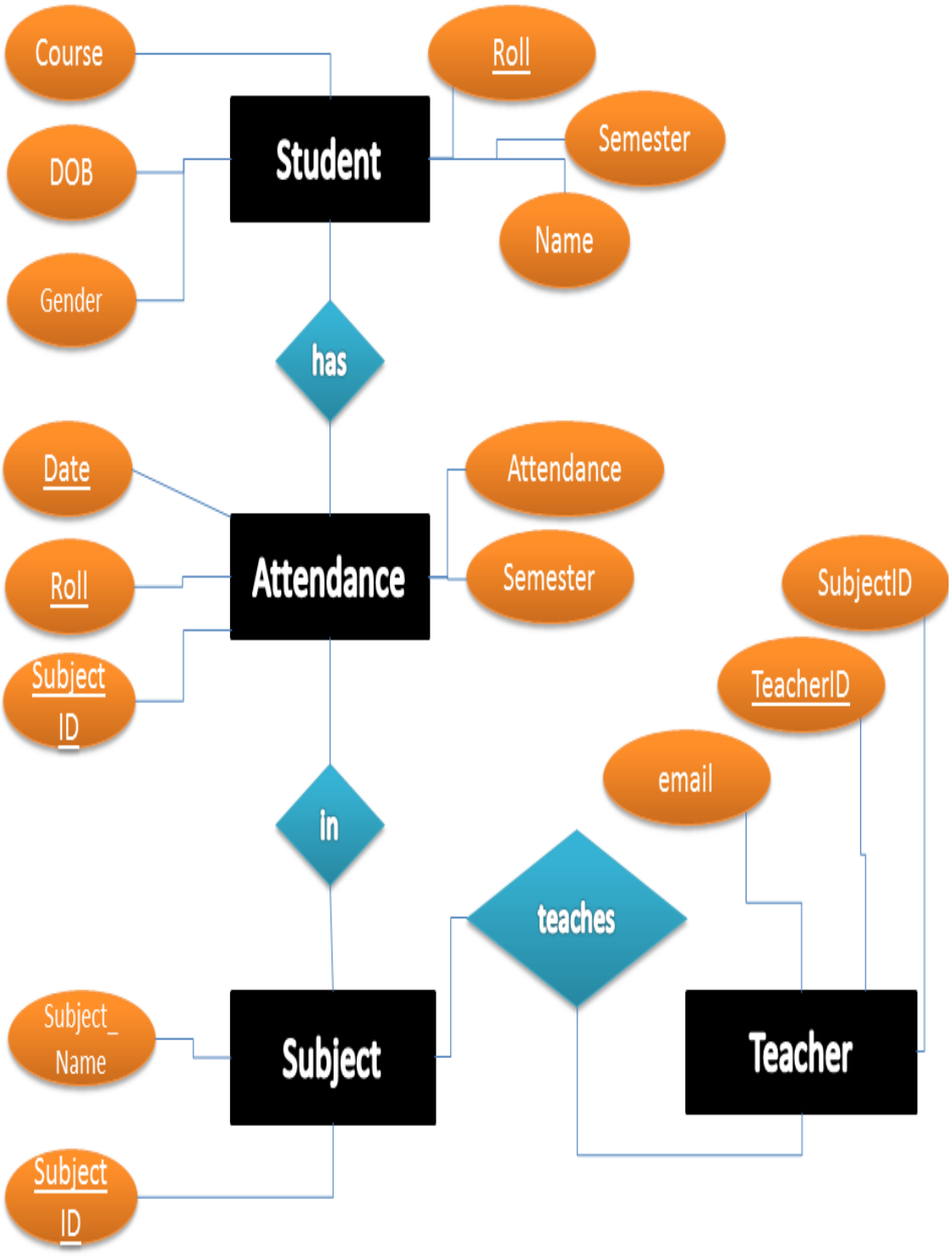

Figure 2.4: ER Diagram

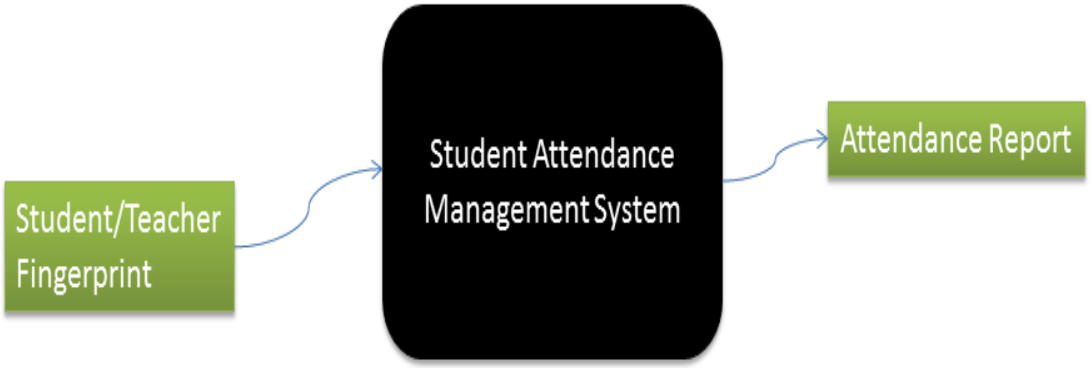

Figure 2.5: Level 0 DFD

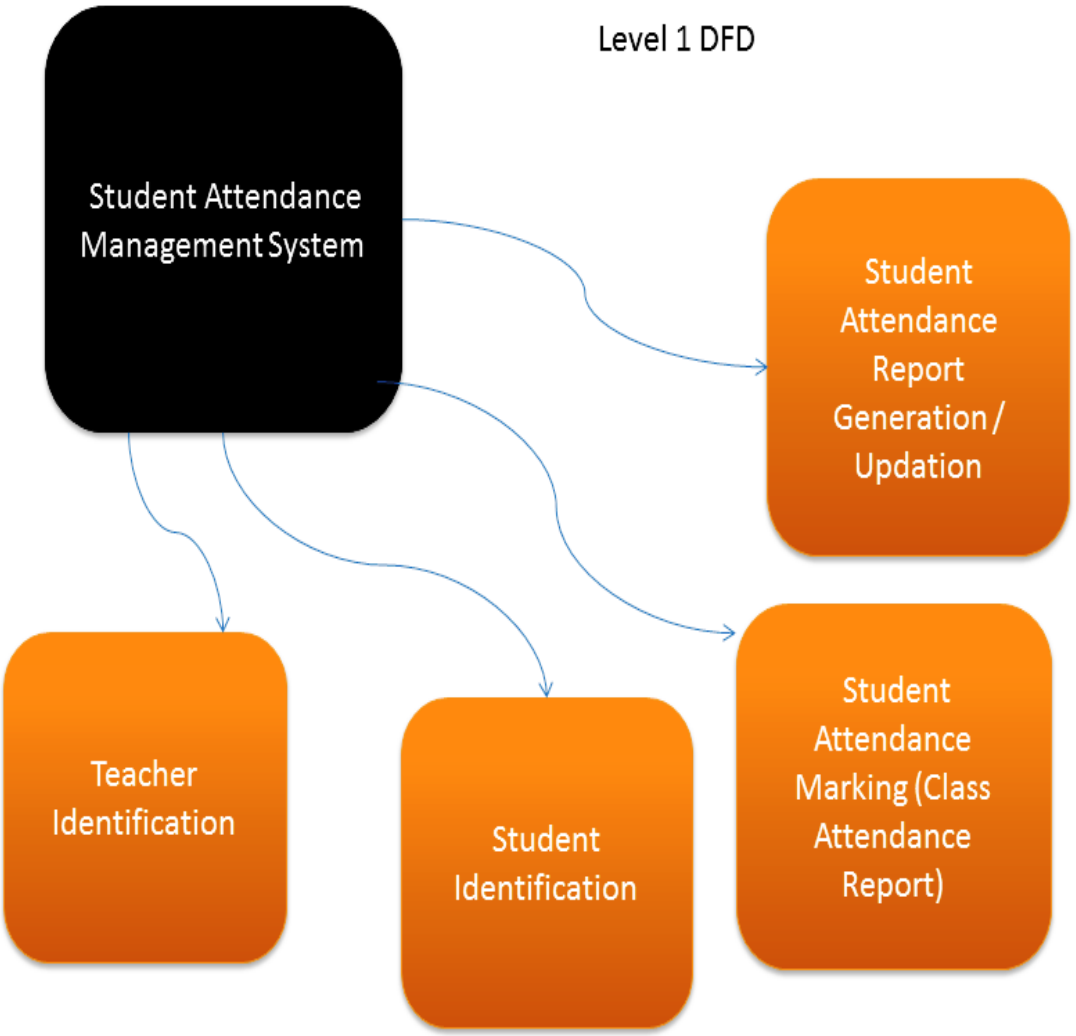

Figure 2.6: Level 1 DFD

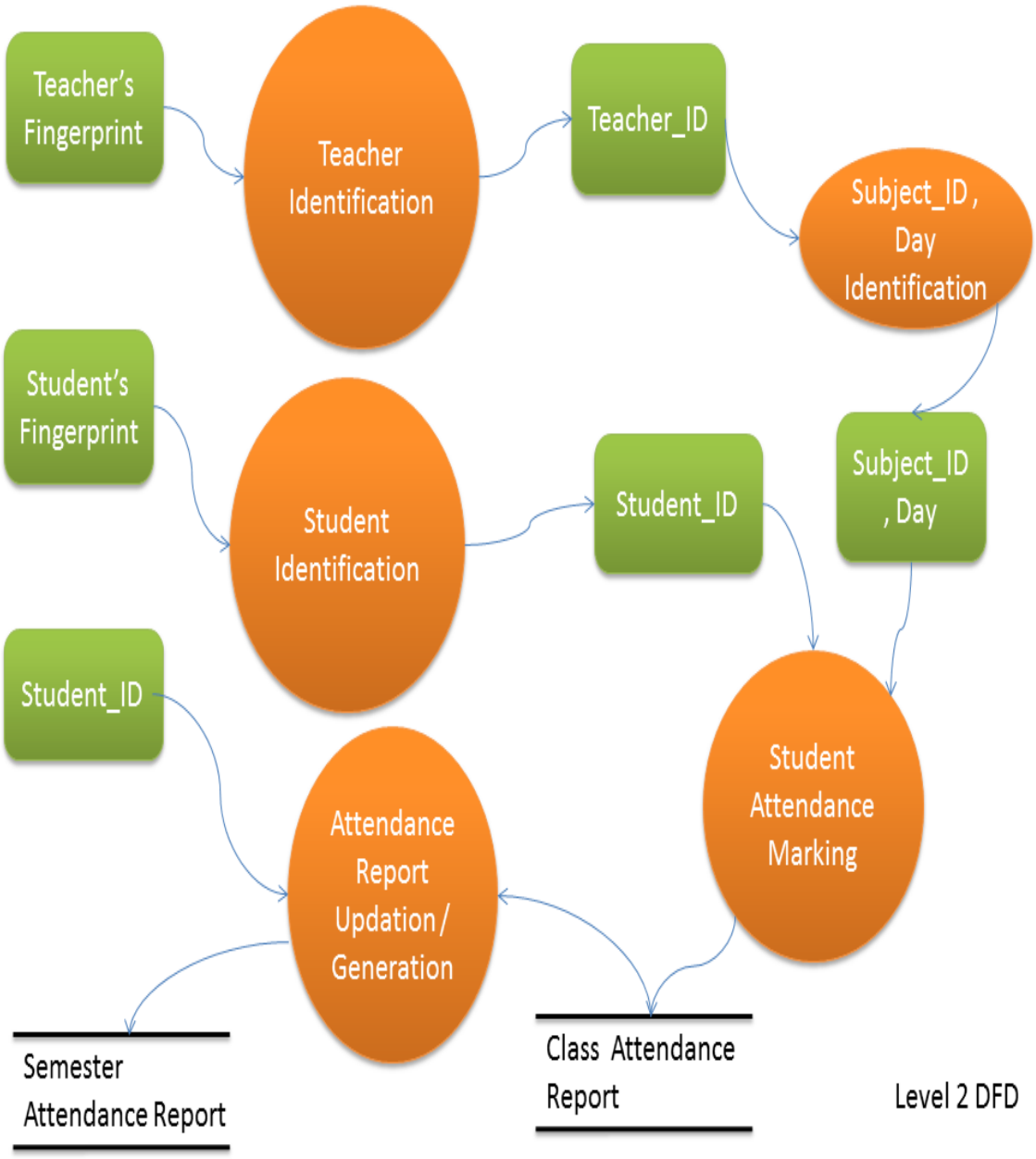

Figure 2.7: Level 2 DFD

This device should have a wireless connection. Using this wireless connection,

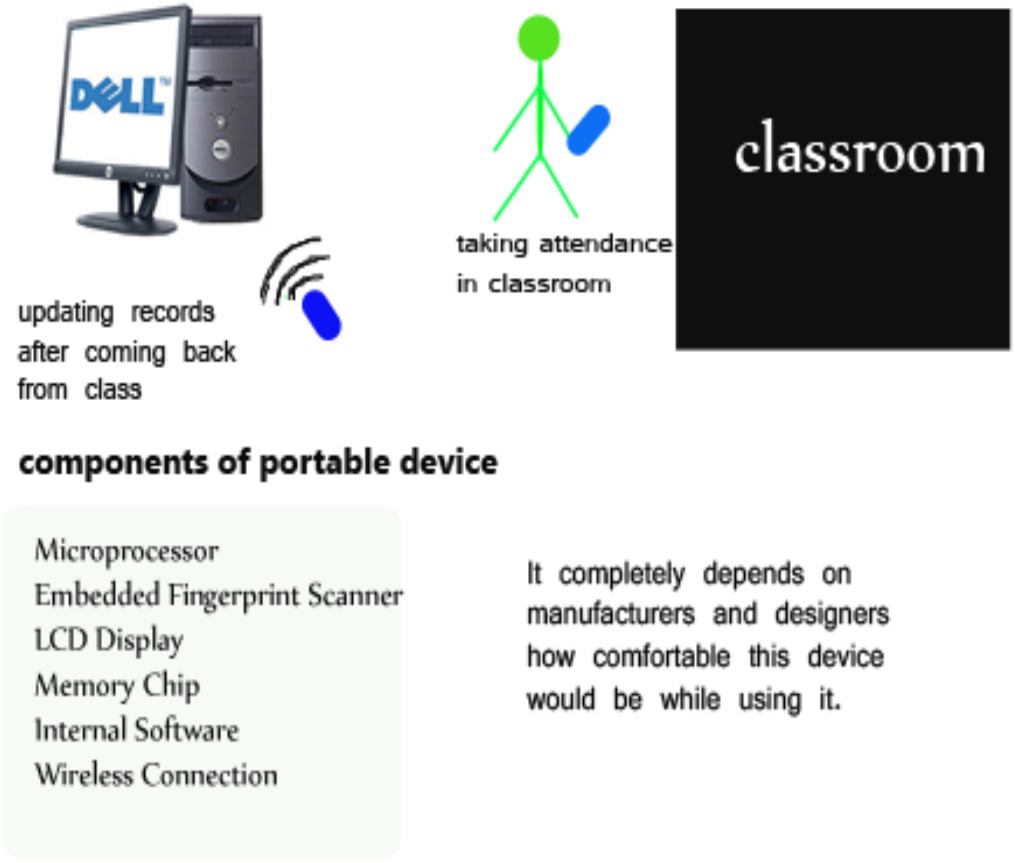

Figure 2.8: Portable Device

attendance taken would be updated automatically when device is in network of the nodes which are storing the attendance records. Database of enrolled fingerprints will be in this portable device. Size of enrolled database was 12.1 MB when 150 fingerprints were enrolled in this project. So for 10000 students, atleast 807 MB or more space would be required for storing enrolled database. For this purpose, a removable memory chip could be used. We cannot use wireless LAN here because fetching data using wireless LAN will not be possible because of less range of wireless devices. So enrolled data would be on chip itself. Attendance results will be updated when portable device will be in the range of nodes which are storing attendance reports. We may update all the records online via the mobile network provided by different companies. Today 3G network provides sufficient throughput which can be used for updating attendance records automatically without going near nodes. In such case,

the need of database inside memory chip will not be mandatory. It will be fetched by using 3G mobile network from central database repository. The design of such a portable device is the task of embedded system engineers.

2.5.1 Using Portable Device

In this section, we suggest the working of portable device and the method of using it for marking attendance. The device may either be having touchscreen input/display or buttons with lcd display. A software specially designed for the device will be running on it. Teachers will verify his/her fingerprint on the device before giving it to students for marking attendance. After verifying the teacher's identity, software will ask for course and and other required information about the class which he or she is going to teach. Software will ask teacher the time after which device will not mark any attendance. This time can vary depending on the teacher's mood but our suggested value is 25 minutes. This is done to prevent late entrance of students. This step will hardly take few seconds. Then students will be given device for their fingerprint identification and attendance marking. In the continuation, teacher will start his/her lecture. Students will hand over the device to other students whose attendance is not marked. After 25 minutes or the time decided by teacher, device will not input any attendance. After the class is over, teacher will take device and will end the lecture. The main function of software running on the device will be fingerprint identification of students followed by report generation and sending reports to servers using 3G network. Other functions will be downloading and updating the database available on the device from central database repository.

2.6 Comparison with other student attendance systems

There are various other kind of student attendance management systems available like RFID based student attendance system and GSM-GPRS based student attendance system. These systems have their own pros and cons. Our system is better because first it saves time that could be used for teaching. Second is portability. Portability

has its own advantage because the device could be taken to any class wherever it is scheduled. While GSM-GPRS based systems use position of class for attendance marking which is not dynamic and if schedule or location of the class changes, wrong attendance might be marked. Problem with RFID based systems is that students have to carry RFID cards and also the RFID detectors are needed to be installed. Nonetheless, students may give proxies easily using friend's RFID card. These problems are not in our system. We used fingerprints as recognition criteria so proxies cannot be given. If portable devices are used, attendance marking will be done at any place and any time. So our student attendance system is far better to be implemented at NITR.

Chapter 3

Fingerprint Identification System

An **identification system** is one which helps in identifying an individual among many people when detailed information is not available. It may involve matching available features of candidate like fingerprints with those already enrolled in database.

3.1 How Fingerprint Recognition works?

Fingerprint images that are found or scanned are not of optimum quality. So we remove noises and enhance their quality. We extract features like minutiae and others for matching. If the sets of minutiae are matched with those in the database, we call it an identified fingerprint. After matching, we perform post-matching steps which may include showing details of identified candidate, marking attendance etc. A brief flowchart is shown in next section.

3.2 Fingerprint Identification System Flowchart

A brief methodology of our Fingerprint Identification System is shown here in following flowchart. Each of these are explained in the later chapters.

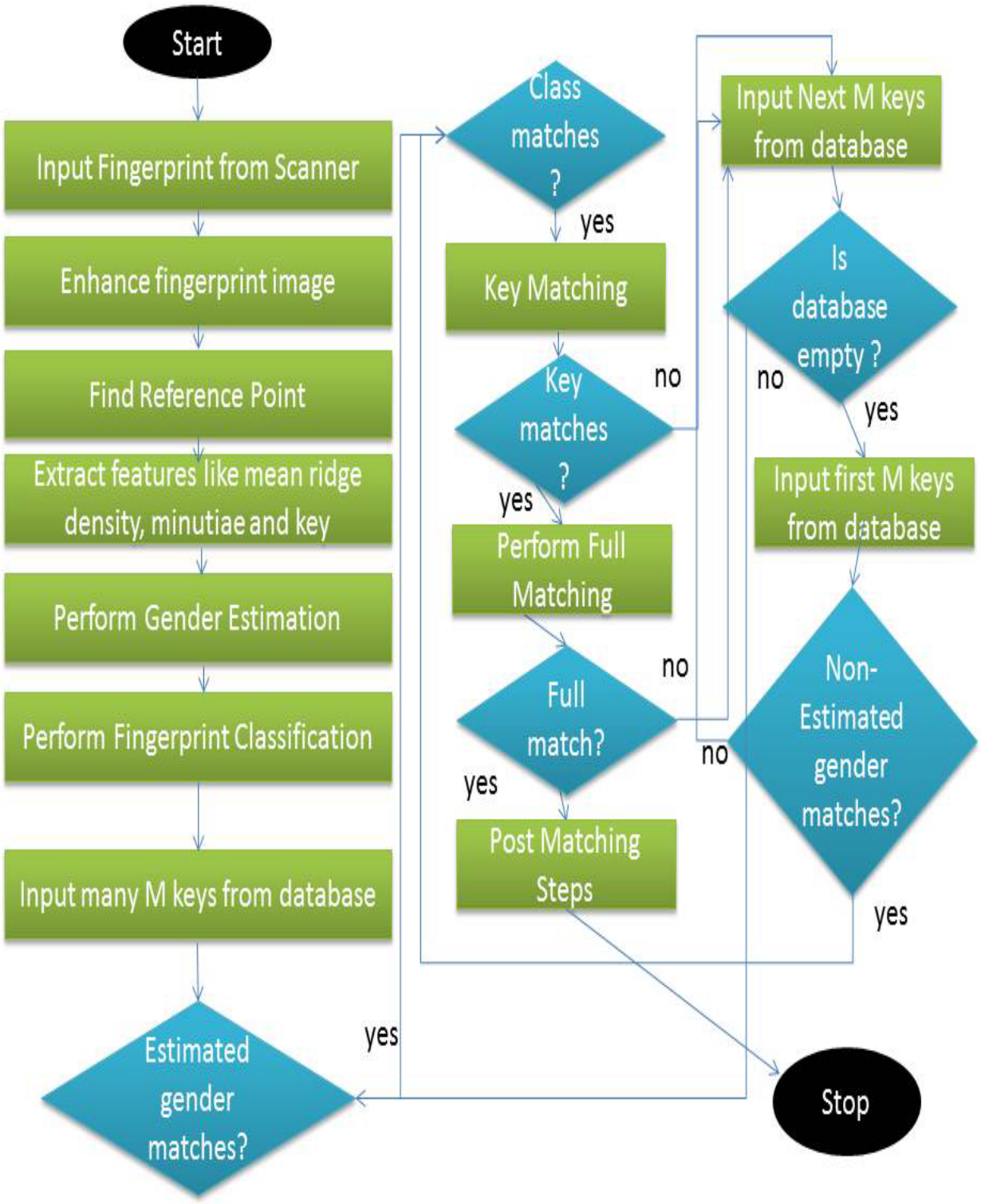

Figure 3.1: Fingerprint Identification System Flowchart

Chapter 4

Fingerprint Enhancement

The image acquired from scanner is sometimes not of perfect quality .It gets corrupted due to irregularities and non-uniformity in the impression taken and due to variations in the skin and the presence of the scars, humidity, dirt etc. To overcome these problems , to reduce noise and enhance the definition of ridges against valleys, various techniques are applied as following.

4.1 Segmentation

Image segmentation [1] separates the foreground regions and the background regions in the image.The foreground regions refers to the clear fingerprint area which contains the ridges and valleys. This is the area of interest. The background regions refers to the regions which is outside the borders of the main fingerprint area, which does not contain any important or valid fingerprint information. The extraction of noisy and false minutiae can be done by applying minutiae extraction algorithm to the background regions of the image. Thus, segmentation is a process by which we can discard these background regions, which results in more reliable extraction of minutiae points.

We are going to use a method based on variance thresholding . The background regions exhibit a very low grey - scale variance value , whereas the foreground regions have a very high variance . Firstly , the image is divided into blocks and the grey-scale variance is calculated for each block in the image . If the variance is less than the global threshold , then the block is assigned to be part of background region or else

it is part of foreground . The grey - level variance for a block of size S x S can be calculated as :

$$Var(k) = \frac{1}{S^2} \sum_{i=0}^{S-1} \sum_{j=0}^{S-1} (G(i,j) - M(k))^2 \tag{4.1}$$

where Var(k) is the grey - level variance for the block k , G(i,j) is the grey - level value at pixel (i,j) , and M(k) denotes the mean grey - level value for the corresponding block k .

4.2 Normalization

Image normalization is the next step in fingerprint enhancement process. Normalization [1] is a process of standardizing the intensity values in an image so that these intensity values lies within a certain desired range. It can be done by adjusting the range of grey-level values in the image. Let G(i, j) denotes the grey-level value at pixel (i, j), and N(i, j) represent the normalized grey-level value at pixel (i, j). Then the normalized image can defined as:

$$N(i,j) = \begin{cases} M_0 + \sqrt{\frac{V_0(G(i,j)-M)^2}{V}} & , if I(i,j) > M \\ M_0 - \sqrt{\frac{V_0(G(i,j)-M)^2}{V}} & , otherwise \end{cases}$$

where M_0 and V_0 are the estimated mean and variance of I(i, j), respectively .

4.3 Orientation estimation

The orientation field of a fingerprint image defines the local orientation of the ridges contained in the fingerprint . The orientation estimation is a fundamental step in the enhancement process as the subsequent Gabor filtering stage relies on the local orientation in order to effectively enhance the fingerprint image. The least mean square estimation method used by Raymond Thai [1] is used to compute the orientation image. However, instead of estimating the orientation block-wise, we have chosen to extend their method into a pixel-wise scheme, which produces a finer and more accurate estimation of the orientation field. The steps for calculating the orientation at pixel (i, j) are as follows:

1. Firstly , a block of size W x W is centered at pixel (i, j) in the normalized fingerprint image.

2. For each pixel in the block, compute the gradients dx (i, j) and dy (i, j), which are the gradient magnitudes in the x and y directions, respectively. The horizontal Sobel operator[6] is used to compute dx(i, j) : [1 0 -1; 2 0 -2;1 0 -1]

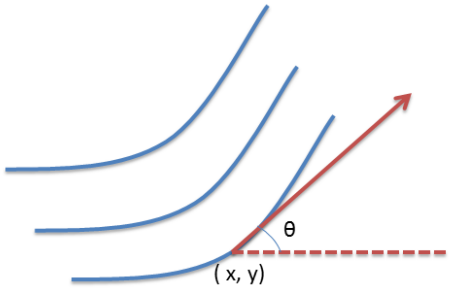

Figure 4.1: Orientation Estimation

3. The local orientation at pixel (i; j) can then be estimated using the following equations:

$$V_x(i,j) = \sum_{u=i-\frac{W}{2}}^{i+\frac{W}{2}} \sum_{v=j-\frac{W}{2}}^{j+\frac{W}{2}} 2\partial_x(u,v)\partial_y(u,v) \tag{4.2}$$

$$V_y(i,j) = \sum_{u=i-\frac{W}{2}}^{i+\frac{W}{2}} \sum_{v=j-\frac{W}{2}}^{j+\frac{W}{2}} \partial_x^2(u,v) - \partial_y^2(u,v), \tag{4.3}$$

$$\theta(i,j) = \frac{1}{2}\tan^{-1}\frac{V_y(i,j)}{V_x(i,j)}, \tag{4.4}$$

where $\theta(i,j)$ is the least square estimate of the local orientation at the block centered at pixel (i, j).

4. Smooth the orientation field in a local neighborhood using a Gaussian filter. The orientation image is firstly converted into a continuous vector field, which is defined as:

$$\phi_x(i,j) = \cos 2\theta(i,j), \tag{4.5}$$

$$\phi_y(i,j) = \sin 2\theta(i,j), \tag{4.6}$$

where ϕ_x and ϕ_y are the x and y components of the vector field, respectively. After

the vector field has been computed, Gaussian smoothing is then performed as follows:

$$\phi'_x(i,j) = \sum_{u=-\frac{w_\phi}{2}}^{\frac{w_\phi}{2}} \sum_{v=-\frac{w_\phi}{2}}^{\frac{w_\phi}{2}} G(u,v)\phi_x(i-uw, j-vw), \tag{4.7}$$

$$\phi'_y(i,j) = \sum_{u=-\frac{w_\phi}{2}}^{\frac{w_\phi}{2}} \sum_{v=-\frac{w_\phi}{2}}^{\frac{w_\phi}{2}} G(u,v)\phi_y(i-uw, j-vw), \tag{4.8}$$

where G is a Gaussian low-pass filter of size w_ϕ x w_ϕ.

5. The final smoothed orientation field O at pixel (i, j) is defined as:

$$O(i,j) = \frac{1}{2}\tan^{-1}\frac{\phi'_y(i,j)}{\phi'_x(i,j)} \tag{4.9}$$

4.4 Ridge Frequency Estimation

Another important parameter,in addition to the orientation image, that can be used in the construction of the Gabor filter is the local ridge frequency. The local frequency of the ridges in a fingerprint is represented by the frequency image. The first step is to divide the image into blocks of size W x W. In the next step we project the grey-level values of each pixels located inside each block along a direction perpendicular to the local ridge orientation. This projection results in an almost sinusoidal-shape wave with the local minimum points denoting the ridges in the fingerprint.It involves smoothing the projected waveform using a Gaussian lowpass filter of size W x W which helps in reducing the effect of noise in the projection. The ridge spacing S(i, j) is then calculated by counting the median number of pixels between the consecutive minima points in the projected waveform. The ridge frequency F(i, j) for a block centered at pixel (i, j) is defined as:

$$F(i,j) = \frac{1}{S(i,j)} \tag{4.10}$$

4.5 Gabor filter

Gabor filters [1] are used because they have orientation-selective and frequency-selective properties. Gabor filters are called the mother of all other filters as other filter can be derived using this filter. Therefore, applying a properly tuned Gabor filter can preserve the ridge structures while reducing noise. An even-symmetric Gabor filter in the spatial domain is defined as :

$$G(x, y, \theta, f) = exp\{-\frac{1}{2}[\frac{x_\theta^2}{\sigma_x^2} + \frac{y_\theta^2}{\sigma_y^2}]\} \cos 2\pi f x_\theta, \tag{4.11}$$

$$x_\theta = x \cos\theta + y \sin\theta, \tag{4.12}$$

$$y_\theta = -x \sin\theta + y \cos\theta, \tag{4.13}$$

where θ is the orientation of the Gabor filter, f is the frequency of the cosine wave, ϕ_x and ϕ_y are the standard deviations of the Gaussian envelope along the x and y axes, respectively, and x_θ and y_θ define the x and y axes of the filter coordinate frame respectively.

The Gabor Filter is applied to the fingerprint image by spatially convolving the image with the filter. The convolution of a pixel (i,j) in the image requires the corresponding orientation value O(i,j) and the ridge frequency value F(i,j) of that pixel .

$$E(i, j) = \sum_{u=-\frac{w_x}{2}}^{\frac{w_x}{2}} \sum_{v=-\frac{w_y}{2}}^{\frac{w_y}{2}} G(u, v, O(i, j), F(i, j)) N(i - u, j - v), \tag{4.14}$$

where O is the orientation image, F is the ridge frequency image, N is the normalized fingerprint image, and wx and wy are the width and height of the Gabor filter mask, respectively.

4.6 Binarisation

Most minutiae extraction algorithms operate on basically binary images where there are only two levels of interest: the black pixels represent ridges, and the white pixels represent valleys. Binarisation [1] converts a greylevel image into a binary image. This helps in improving the contrast between the ridges and valleys in a fingerprint image, and consequently facilitates the extraction of minutiae. One very useful property of the Gabor filter is that it contains a DC component of zero, which indicates that the resulting filtered image has a zero mean pixel value. Hence, binarisation of the image can be done by using a global threshold of zero. Binarisation involves examining the grey-level value of every pixel in the enhanced image, and, if the grey-level value is greater than the predefined global threshold, then the pixel value is set to value one; else, it is set to zero. The outcome of binarisation is a binary image which contains two levels of information, the background valleys and the foreground ridges.

4.7 Thinning

Thinning is a morphological operation which is used to remove selected foreground pixels from the binary images. A standard thinning algorithm from [1] is used, which performs this operation using two subiterations. The algorithm can be accessed by a software MATLAB via the 'thin' operation of the bwmorph function. Each subiteration starts by examining the neighborhood of every pixel in the binary image, and on the basis of a particular set of pixel-deletion criteria, it decides whether the pixel can be removed or not. These subiterations goes on until no more pixels can be removed.

Figure 4.2: (a)Original Image, (b)Enhanced Image, (c)Binarised Image, (d)Thinned Image

Chapter 5

Feature Extraction

After improving quality of the fingerprint image we extract features from binarised and thinned images. We extract reference point, minutiae and key(used for one to many matching).

5.1 Finding the Reference Point

Reference point is very important feature in advanced matching algorithms because it provides the location of origin for marking minutiae. We find the reference point using the algorithm as in [2]. Then we find the relative position of minutiae and estimate the orientation field of the reference point or the singular point.

The technique is to extract core and delta points using Poincare Index. The value of Poincare index is $180^o, -180^o$ and 0^o for a core, a delta and an ordinary point respectively. Complex filters are used to produce blur at different resolutions.Singular point (SP) or reference point is the point of maximum filter response of these filters applied on image. Complex filters , $exp(im\theta)$, of order m (= 1 and -1) are used to produce filter response. Four level resolutions are used here:level 0, level 1, level 2, level 3. Level 3 is lowest resolution and level 0 is highest resolution.Only filters of first order are used : $h = (x + iy)^m g(x, y)$ where g(x,y) is a gaussian defined as $g(x, y) = exp-((x^2 + y^2)/2\sigma^2)$ and $m = 1, -1$.

Filters are applied to the complex valued orientation tensor field image $z(x, y) = (f_x + if_y)^2$ and not directly to the image. Here fx is the derivative of the original image in the x-direction and fy is the derivative in the y-direction.To find the position of a possible

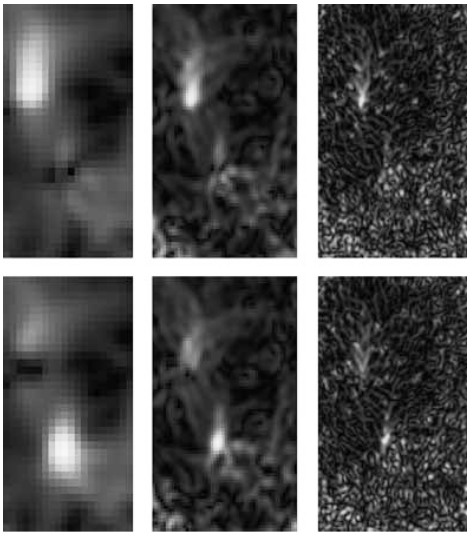

Figure 5.1: Row 1: filter response c_{1k} , k = 3, 2, and 1. Row 2: filter response c_{2k} , k = 3, 2, and 1.

SP in a fingerprint the maximum filter response is extracted in image c_{13} and in c_{23} (i.e. filter response at $m = 1$ and level 3 (c_{13}) and at $m = -1$ and level 3 (c_{23})). The search is done in a window computed in the previous higher level (low resolution).The filter response at lower level (high resolution) is used for finding response at higher level (low resolution).At a certain resolution (level k), if $c_{nk}(x_j, y_j)$ is higher than a threshold an SP is found and its position (x_j, y_j) and the complex filter response $c_{nk}(x_j, y_j)$ are noted.

5.2 Minutiae Extraction and Post-Processing

5.2.1 Minutiae Extraction

The most commonly employed method of minutiae extraction is the Crossing Number (CN) concept [1] . This method involves the use of the skeleton image where the ridge flow pattern is eight-connected. The minutiae are extracted by scanning the local neighborhood of each ridge pixel in the image using a 3 x 3 window. The CN value is then computed, which is defined as half the sum of the differences between pairs of adjacent pixels in the eight-neighborhood. Using the properties of the CN as shown in figure 5, the ridge pixel can then be classified as a ridge ending, bifurcation or non-minutiae point. For example, a ridge pixel with a CN of one corresponds to a ridge ending, and a CN of three corresponds to a bifurcation.

Table 5.1: Properties of Crossing Number

CN	Property
0	Isolated Point
1	Ridge Ending Point
2	Continuing Ridge Point
3	Bifurcation Point
4	Crossing Point

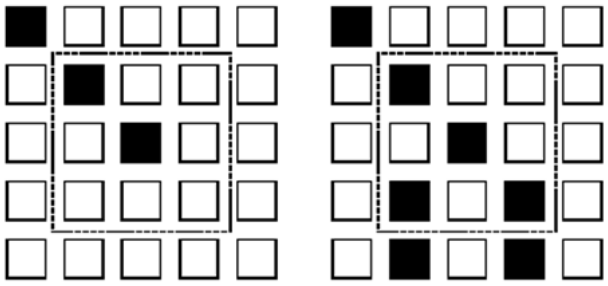

Figure 5.2: Examples of (a)ridge-ending (CN=1) and (b)bifurcation pixel (CN=3)

5.2.2 Post-Processing

False minutiae may be introduced into the image due to factors such as noisy images, and image artefacts created by the thinning process. Hence, after the minutiae are extracted, it is necessary to employ a post-processing [1] stage in order to validate the minutiae. Figure 5.3 illustrates some examples of false minutiae structures, which include the spur, hole, triangle and spike structures . It can be seen that the spur structure generates false ridge endings, where as both the hole and triangle structures generate false bifurcations. The spike structure creates a false bifurcation and a false ridge ending point.

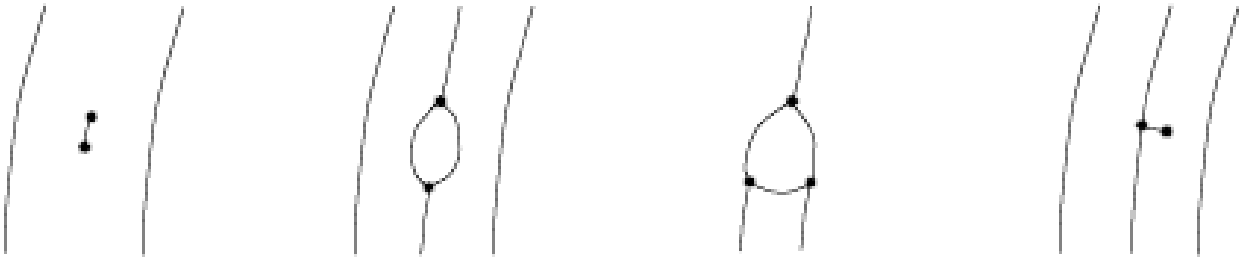

Figure 5.3: Examples of typical false minutiae structures : (a)Spur, (b)Hole, (c)Triangle, (d)Spike

5.2.3 Removing Boundary Minutiae

For removing boundary minutiae, we used pixel-density approach. Any point on the boundary will have less white pixel density in a window centered at it, as compared to inner minutiae. We calculated the *limit*, which indicated that pixel density less than that means it is a boundary minutiae. We calculated it according to following formula:

$$limit = (\frac{w}{W_{freq}} * (ridgedensity)) * \frac{w}{2} \tag{5.1}$$

where w is the window size, W_{freq} is the window size used to compute ridge density.

Figure 5.4: Skeleton of window centered at boundary minutiae

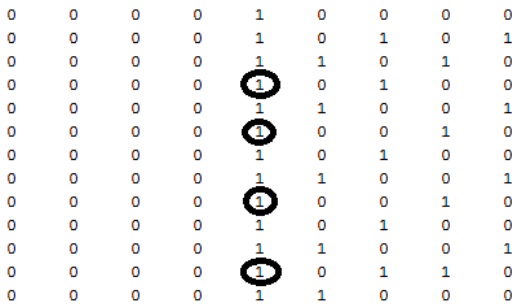

Figure 5.5: Matrix Representation of boundary minutiae

Now, in thinned image, we sum all the pixels in the window of size w centered at the boundary minutiae. If sum is less than limit, the minutiae is considered as boundary minutiae and is discarded.

5.3 Extraction of the key

5.3.1 What is key?

Key is used as a hashing tool in this project. Key is small set of few minutiae closest to reference point. We match minutiae sets, if the keys of sample and query fingerprints matches. Keys are stored along with minutiae sets in the database. Advantage of using key is that, we do not perform full matching every time for non-matching minutiae sets, as it would be time consuming. For large databases, if we go on matching full minutiae set for every enrolled fingerprint, it would waste time unnecessarily. Two types of keys are proposed - simple and complex. Simple key has been used in this project.

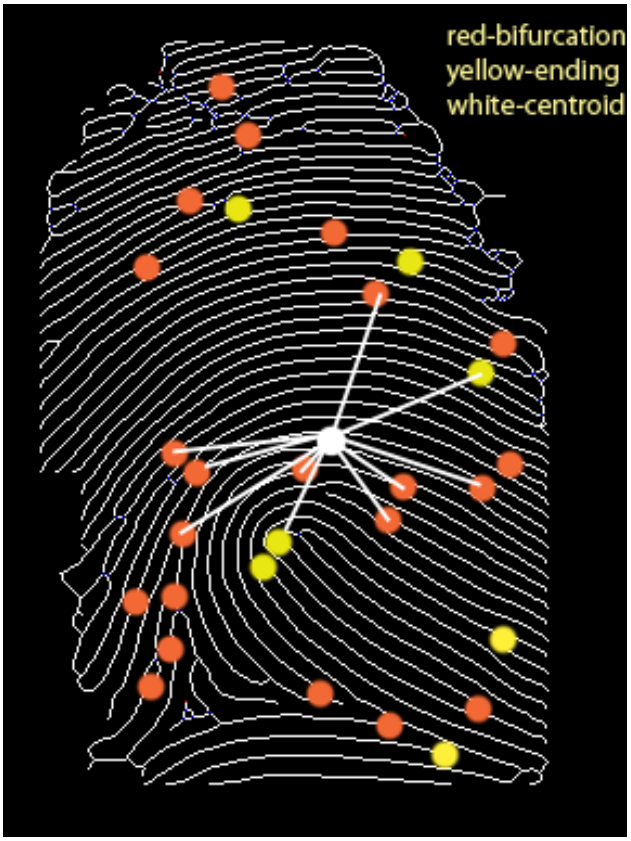

Figure 5.6: Key Representation

Simple Key

This type of key has been used in this project. Minutiae which constitute this key are ten minutiae closest to the reference point or centroid of all minutiae, in sorted

order. Five fields are stored for each key value i.e. $(x, y, \theta, t, r).(x, y)$ is the location of minutiae, θ is the value of orientation of ridge related to minutia with respect to orientation of reference point, t is type of minutiae, and r is distance of minutiae from origin. Due to inaccuracy and imperfection of reference point detection algorithm, we used centroid of all minutiae for construction of key.

Complex Key

The complex key stores more information and is structurally more complex. It stores vector of minutiae in which next minutiae is closest to previous minutiae, starting with reference point or centroid of all minutiae. It stores $< x, y, \theta, t, r, d, \alpha >$. Here x,y,t,r,$\theta$ are same, d is distance from previous minutiae entry and α is difference in ridge orientation from previous minutiae.

Data: minutiaelist = Minutiae Set, refx = x-cordinate of centroid, refy = y-cordinate of centroid

Result: Key

d(10)=null;

for $j = 1$ *to* 10 **do**

  for $i = 1$ *to* $rows(minutiaelist)$ **do**

    $d(i) \leftarrow ((minutiaelist(i, 1) - refx)^2 + (minutiaelist(i, 2) - refy)^2)^{0.5}$;

    if $d(i) \leq d(i - 1)$ **then**

      $next \leftarrow minutiaelist(i, :)$;

    end

  end

end

Algorithm 1: Key Extraction Algorithm

Chapter 6

Partitioning of Database

Before we partition the database, we perform gender estimation and classification.

6.1 Gender Estimation

In [3], study on 100 males and 100 females revealed that significant sex differences occur in the fingerprint ridge density. Henceforth, gender of the candidate can be estimated on the basis of given fingerprint data. Henceforth, gender of the candidate can be estimated on the basis of given fingerprint data. Based on this estimation, searching for a record in the database can be made faster. **Method for finding mean ridge density and estimated gender:** The highest and lowest values for male and female ridge densities will be searched. If ridge density of query fingerprint is less than the lowest ridge density value of females, the query fingerprint is obviously of a male. Similarly, if it is higher than highest ridge density value of males, the query fingerprint is of a female. So the searching will be carried out in male or female domains. If the value is between these values, we search on the basis of whether the mean of these values is less than the density of query image or higher.

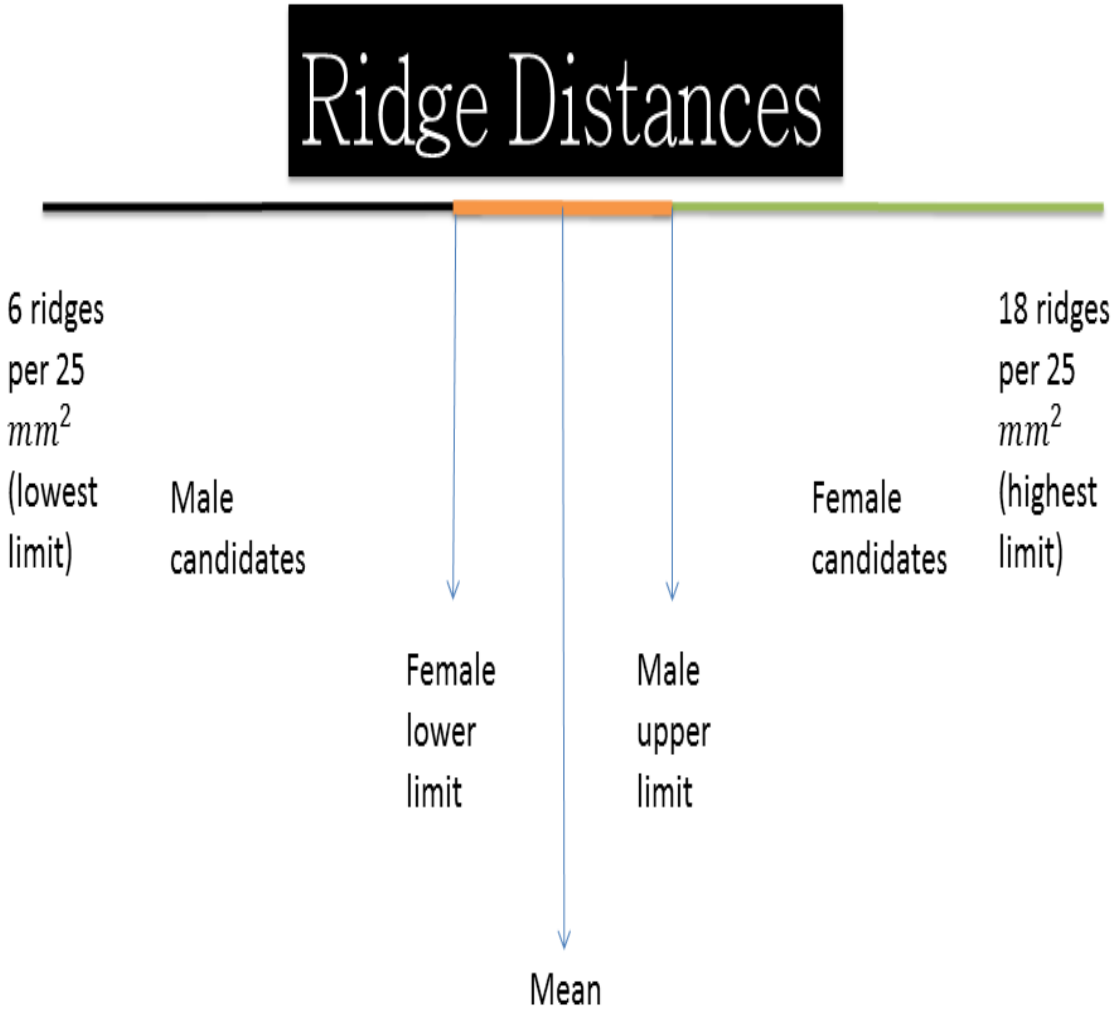

Figure 6.1: Gender Estimation

Data: Size of Database = N; Ridge Density of query fingerprint = s

Result: Estimated Gender i.e. male or female

maleupperlimit=0;

femalelowerlimit=20;

mean=0;

for *image* ← 1 **to** N **do**

  if *gender of image is male* **then**

    if *Ridge Density of image* > *maleupperlimit* **then**
      maleupperlimit ← Ridge Density of image;

      mean ← mean + Ridge Density of image;

    end

  end

  if *gender of image is female* **then**

    if *Ridge Density of image* < *femalelowerlimit* **then**
      femalelowerlimit ← Ridge Density of image;

      mean ← mean + Ridge Density of image;

    end

  end

end

mean ← $\frac{mean}{N}$;

if $s <$ *maleupperlimit* **then**
  estimatedgender ← female

end

else if $s <$ *femalelowerlimit* **then**
  estimatedgender ← male

end

else if $s <$ *mean* **then**
  estimatedgender ← male

end

else if $s >=$ *mean* **then**
  estimatedgender ← female

end

Algorithm 2: Gender Estimation Algorithm

6.2 Classification of Fingerprint

We divide fingerprint into five classes - arch or tented arch, left loop, right loop, whorl and unclassified. The algorithm for classification [4] is used in this project. They used a ridge classification algorithm that involves three categories of ridge structures:non-recurring ridges, type I recurring ridges and type II recurring ridges. N1 and N2 represent number of type I recurring ridges and type II recurring ridges respectively. Nc and Nd are number of core and delta in the fingerprint. To find core and delta, separate 135^o blocks from orientation image. 135^o blocks are shown in following figures.

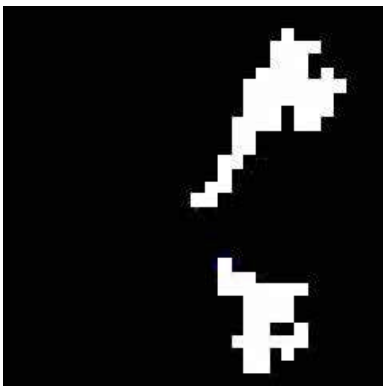

Figure 6.2: 135^o blocks of a fingerprint

Based on number of such blocks and their relative positions, the core and delta are found using Poincare index method. After these, classification is done as following:

1. If (N2 > 0) and (Nc = 2) and (Nd = 2), then a whorl is identified.
2. If (N1 = 0) and (N2 = 0) and (Nc = 0) and (Nd = 0), then an arch is identified.
3. If (N1 > 0) and (N2 = 0) and (Nc = 1) and (Nd = 1), then classify the input using the core and delta assessment algorithm[4].
4. If (N2 > T2) and (Nc > 0), then a whorl is identified.
5. If (N1 > T1) and (N2 = 0) and (Nc = 1) then classify the input using the core and delta assessment algorithm[4].
6. If (Nc = 2), then a whorl is identified.
7. If (Nc = 1) and (Nd = 1), then classify the input using the core and delta assessment algorithm[4].
8. If (N1 > 0) and (Nc = 1), then classify the input using the core and delta assessment algorithm.

9. If (Nc = 0) and (Nd = 0), then an arch is identified.

10. If none of the above conditions is satisfied, then reject the fingerprint.

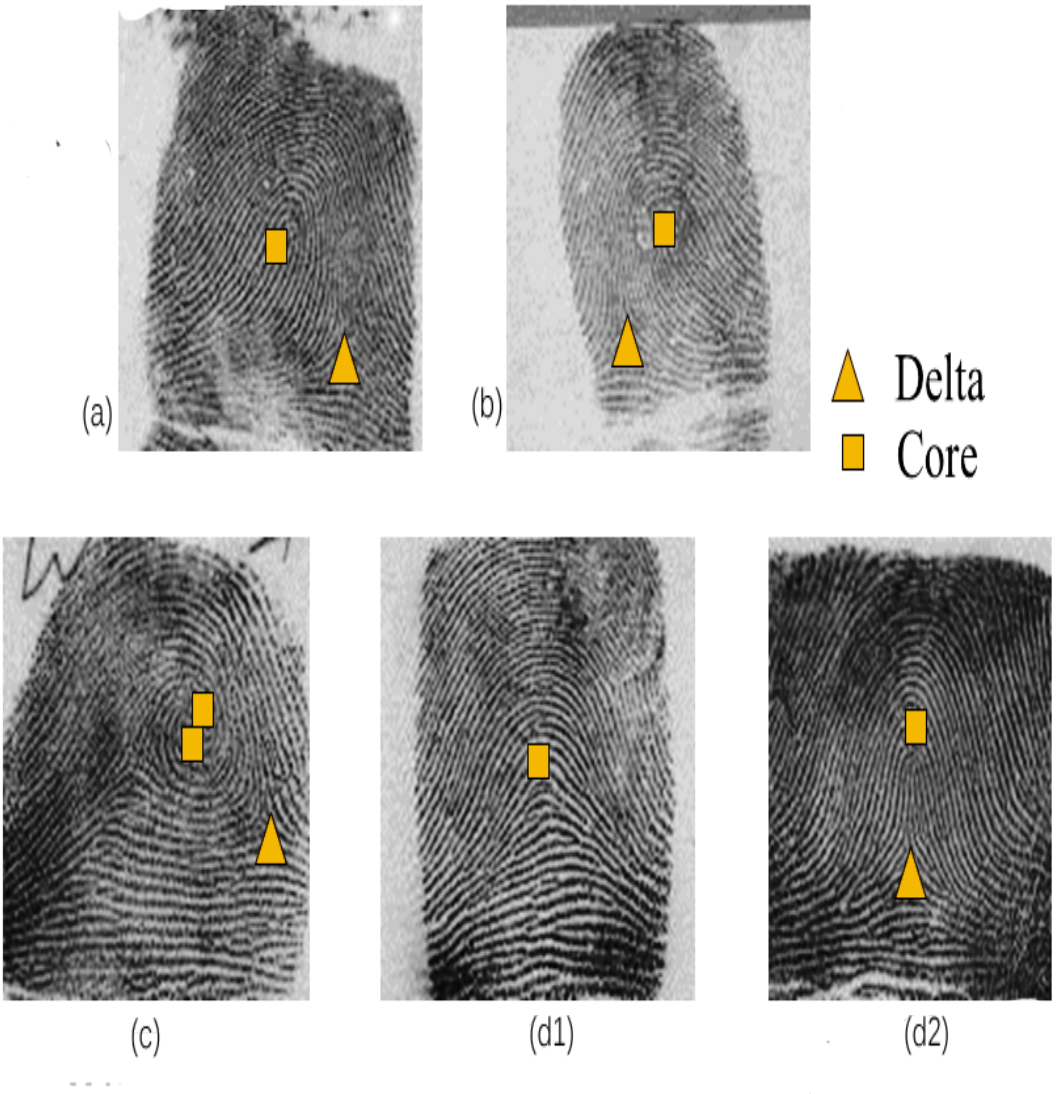

Figure 6.3: Fingerprint Classes (a)Left Loop, (b)Right Loop, (c)Whorl, (d1)Arch, (d2)Tented Arch

6.3 Partitioning

After we estimate gender and find the class of fingerprint, we know which fingerprints to be searched in the database. We roughly divide database into one-tenth using the above parameters. This would roughly reduce identification time to one-tenth.

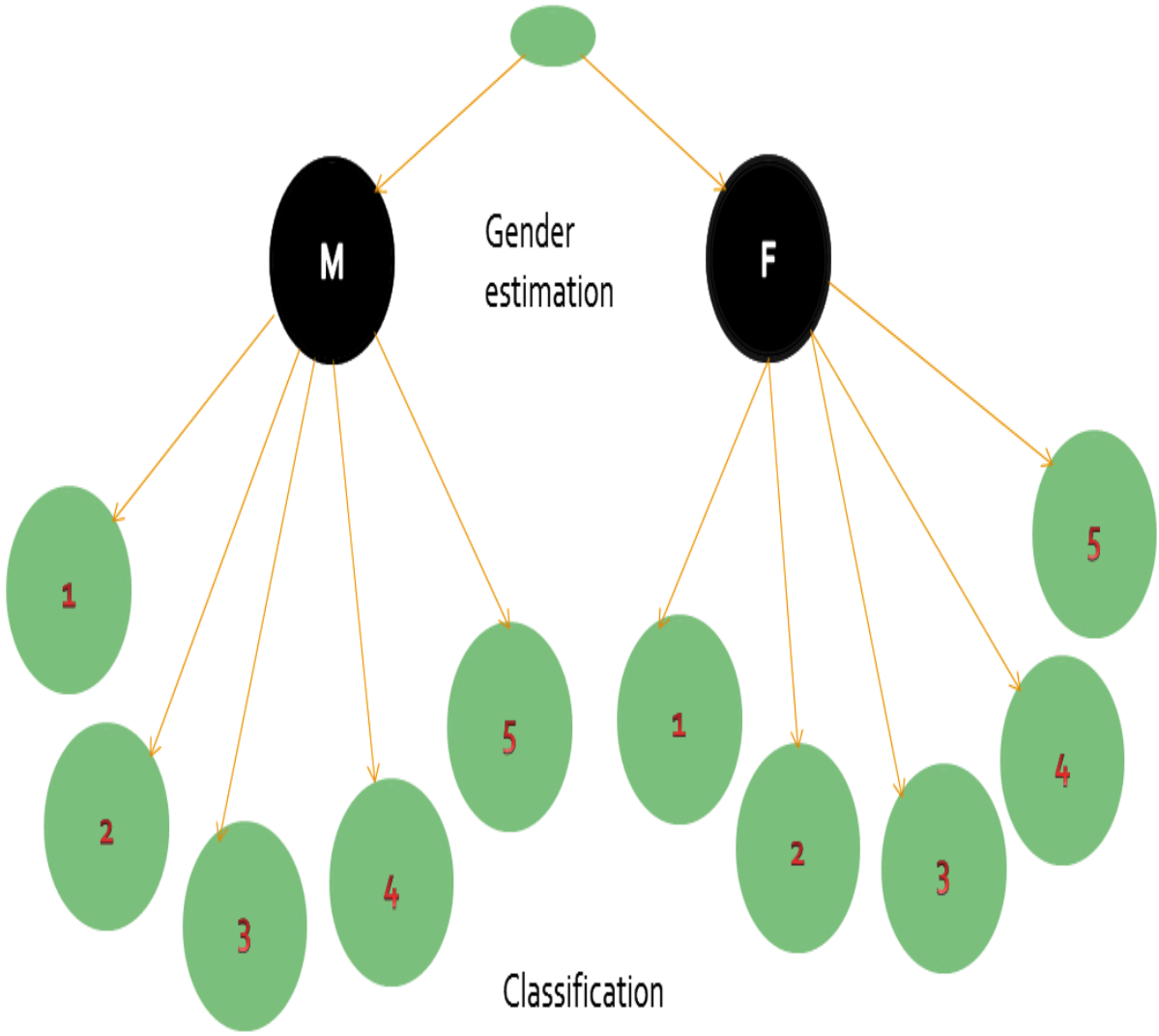

Figure 6.4: Partitioning Database

Chapter 7

Matching

Matching means finding most appropriate similar fingerprint to query fingerprint. Fingerprints are matched by matching set of minutiae extracted. Minutiae sets never match completely, so we compute match score of matching. If match score satisfies accuracy needs, we call it successful matching. We used a new key based one to many matching intended for large databases.

7.1 Alignment

Before we go for matching, minutiae set need to be aligned(registered) with each other. For alignment problems, we used hough transform based registration technique similar to one used by Ratha et al[5]. Minutiae alignment is done in two steps - minutiae registration and pairing. Minutiae registration involves aligning minutiae using parameters $< \delta x, \delta y, \alpha >$ which range within specified limits. $(\delta x, \delta y)$ are translational parameters and α is rotational parameter. Using these parameters, minutiae sets are rotated and translated within parameters limits. Then we find pairing scores of each transformation and transformation giving maximum score is registered as alignment transformation. Using this transformation $< \delta x, \delta y, \alpha >$, we align query minutiae set with the database minutiae set. Algorithm is same as in [5] but we have excluded factor δs i.e. the scaling parameter because it does not affect much the alignment process. α lies from -20 degrees to 20 degrees in steps of 1 or 2 generalized as $< \alpha_1, \alpha_2, \alpha_3 ... \alpha_k >$ where k is number of rotations applied. For every query minutiae i we check if $\alpha_k + \theta_i = \theta_j$ where θ_i and θ_j are orientation

parameters of i^{th} minutia of query minutiae set and j^{th} minutia of database minutiae set. If condition is satisfied, A(i,j,k) is flagged as 1 else 0. For all these flagged values, $(\delta x, \delta y)$ is calculated using following formula:

$$(\delta_x, \delta_y) = q_j - \begin{pmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{pmatrix} * p_i, \tag{7.1}$$

where q_j and p_i are the coordinates of j^{th} minutiae of database minutiae set and i^{th} minutiae of query minutiae set respectively. Using these $< \delta x, \delta y, \alpha_k >$ values, whole query minutiae set is aligned. This aligned minutiae set is used to compute pairing score. Two minutiae are said to be paired only when they lie in same bounding box and have same orientation. Pairing score is (number of paired minutiae)/(total number of minutiae). The i,j,k values which have highest pairing score are finally used to align minutiae set. Co-ordinates of aligned minutiae are found using the formula:

$$q_j = \begin{pmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{pmatrix} * p_i + (\delta_x, \delta_y), \tag{7.2}$$

After alignment, minutiae are stored in sorted order of their distance from their centroid or core.

7.2 Existing Matching Techniques

Most popular matching technique of today is the simple minded n^2 matching where n is number of minutiae. In this matching each minutiae of query fingerprint is matched with n minutiae of sample fingerprint giving total number of n^2 comparisons. This matching is very orthodox and gives headache when identification is done on large databases.

7.3 One to Many matching

Few algorithms are proposed by many researchers around the world which are better than normal n^2 matching. But all of them are one to one verification or one to one identification matching types. We developed a one to many matching technique which uses key as the hashing tool. Initially, we do not match minutiae sets instead we per-

form key matching with many keys of database. Those database fingerprints whose keys match with key of query fingerprint, are allowed for full minutiae matching. Key matching and full matching are performed using k*n matching algorithm discussed in later section. Following section gives method for one to many matching.

Data: Query Fingerprint;
Result: Matching Results;
Acquire Fingerprint, Perform Enhancement, Find Fingerprint Class, Extract Minutiae, Remove Spurious and Boundary Minutiae, Extract Key,Estimate Gender;
M ← Input M from user;
j=1;
N ← Size of Database;
while $j <= N$ **do**
  if $N - j >= 9$ **then**
    limit=M;
  else
    limit=N-j;
  end
  for $i = j$ *to* $j + limit - 1$ **do**
    Perform Matching(Gender, Class, i);
  end
  for $i = 1$ *to* M **do**
    Gender ← Opposite of Estimated Gender;
    Perform Matching(Gender, Class, i);
  end
end

Algorithm 3: Key Based One to Many Matching Algorithm

7.3.1 Method of One to Many Matching

The matching algorithm will be involving matching the key of the query fingerprint with the many(M) keys of the database.Those which matches ,their full matching will be processed, else the query key will be matched with next M keys and so on.

```
Data: Gender, Class, i;
Result: Matching Results;
egender ← Fetch gender of enrolled fingerprint(i);
if Gender = egender then
    eclass ← Fetch class of enrolled fingerprint(i);
    if Class = eclass then
        ekey ← Fetch key of enrolled fingerprint(i);
        keymatchstatus ← Perform key matching;
        if keymatchstatus = success then
            eminutiae ← Fetch Minutiae list of enrolled fingerprint(i);
            fullmatchstatus ← Perform full matching;
            if fullmatchstatus = success then
                Do Post-Matching Steps;
            end
        end
    end
end
```

Algorithm 4: Matching Algorithm

7.4 Performing key match and full matching

Both key matching and full matching are performed using our k*n matching technique. Here k is a constant(recommended value is 15) chosen by us. In this method, we match i^{th} minutiae of query set with k unmatched minutiae of sample set. Both the query sets and sample sets must be in sorted order of distance from reference point or centroid. i^{th} minutia of query minutiae list is matched with top k unmatched minutiae of database minutiae set.

This type of matching reduces matching time of n^2 to k*n. If minutiae are 80 in number and we chose k to be 15, the total number of comparisons will reduce from 80*80=6400 to 80*15=1200. And this means our matching will be k/n times faster than n^2 matching.

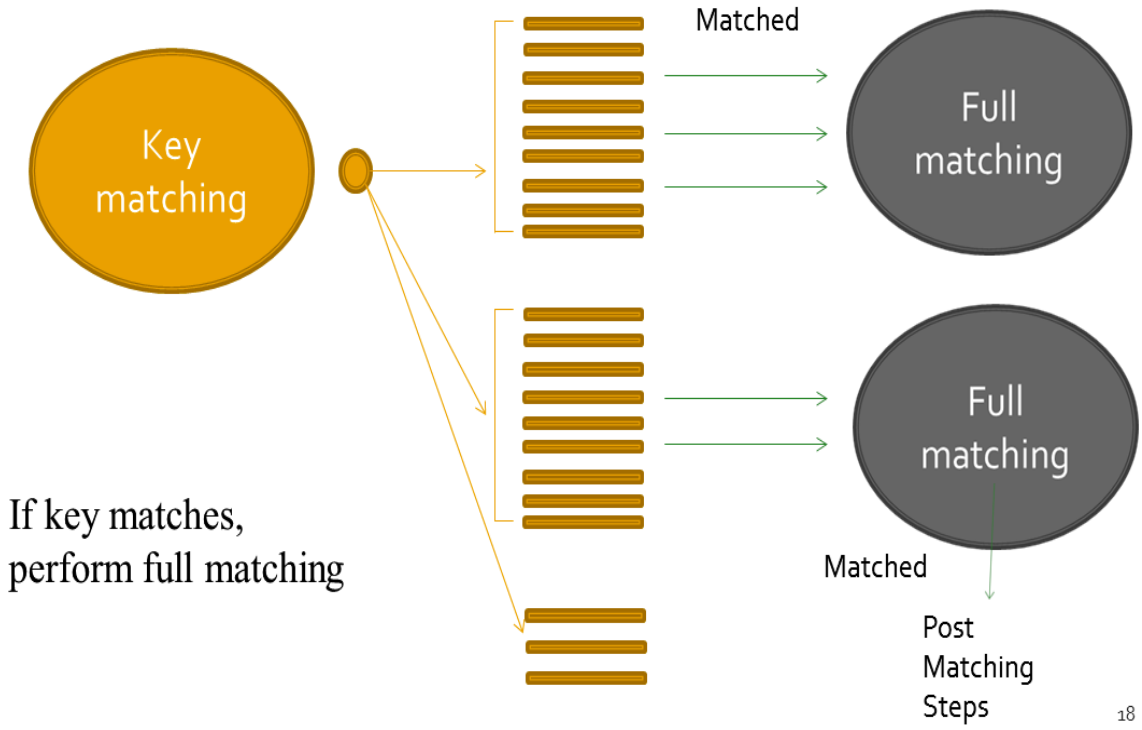

Figure 7.1: One to Many Matching

7.5 Time Complexity of this matching technique

Let s = size of the key, n = number of minutiae, N = number of fingerprints matched till successful identification, k = constant (see previous section). There would be N-1 unsuccessful key matches, one successful key match, one successful full match. Time for N-1 unsuccessful key matches is (N-1)*s*k (in worst case), for successful full match is s*k and for full match is n*k. Total time is (N-1)*s*k+n*k+s*k = k(s*N+n). Here s=10 and we have reduced database to be searched to 1/10th ,so $N \leftarrow N/10$. Total time now is k(10*N/10+n)=k(N+n). Hence, order of our matching algorithm is O(k(N+n))=O(N+n), k is constant. While if we would have used simple n^2 matching technique, it would have been O(Nn^2). For large databases, our matching technique is best to use. Averaging for every fingerprint, we have O(1+n/N) in this identification process which comes to O(1) when N >> n. So we can say that our identification system has constant average matching time when database size is millions.

Chapter 8

Experimental Analysis

8.1 Implementation Environment

We tested our algorithm on several databases like FVC2004, FVC2000 and Verifinger databases. We used a computer with 2GB RAM and 1.83 GHz Intel Core2Duo processor and softwares like Matlab10 and MSAccess10.

8.2 Fingerprint Enhancement

8.2.1 Segmentation and Normalization

Segmentation was performed and it generated a mask matrix which has values as 1 for ridges and 0 for background . Normalization was done with mean = 0 and variance = 1 (fig 8.1).

Figure 8.1: Normalized Image

8.2.2 Orientation Estimation

In orientation estimation, we used block size = 3*3. Orientations are shown in figure 8.2.

Figure 8.2: Orientation Image

8.2.3 Ridge Frequency Estimation

Ridge density and mean ridge density were calculated. Darker blocks indicated low ridge density and vice-versa. Ridge frequencies are shown in figure 8.3.

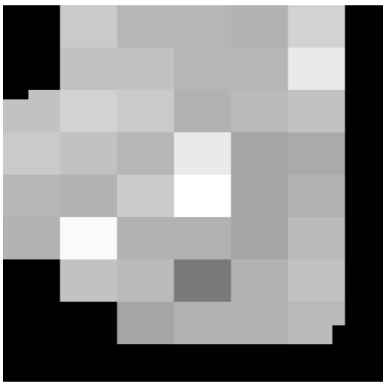

Figure 8.3: Ridge Frequency Image

8.2.4 Gabor Filters

Gabor filters were employed to enhance quality of image. Orientation estimation and ridge frequency images are requirements for implementing gabor filters. σ_x and σ_y are taken 0.5 in Raymond Thai, but we used $\sigma_x = 0.7$ and $\sigma_y = 0.7$. Based on these values , we got results which were satisfiable and are shown in figure 8.4.

Figure 8.4: Left-Original Image, Right-Enhanced Image

8.2.5 Binarisation and Thinning

After the fingerprint image is enhanced, it is then converted to binary form, and submitted to the thinning algorithm which reduces the ridge thickness to one pixel wide. Results of binarisation are shown in figure 8.5 and of thinning are shown in figure 8.6.

Figure 8.5: Binarised Image

Figure 8.6: Thinned Image

8.3 Feature Extraction

8.3.1 Minutiae Extraction and Post Processing

Minutiae Extraction

Using the crossing number method, we extracted minutiae. For this we used skeleton image or the thinned image. Due to low quality of fingerprint, a lot of false and boundary minutiae were found. So we moved forward for post-processing step. Results are shown in figure 8.7 and 8.8.

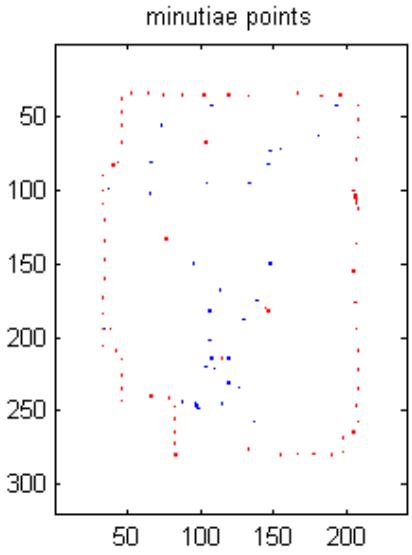

Figure 8.7: All Extracted Minutiae

Figure 8.8: Composite Image with spurious and boundary minutiae

After Removing Spurious and Boundary Minutiae

False minutiae were removed using method described in earlier section. For removing boundary minutiae, we employed our algorithm which worked fine and minutiae extraction results are shown in table 8.2. Results are shown in figure 8.9 and 8.10.

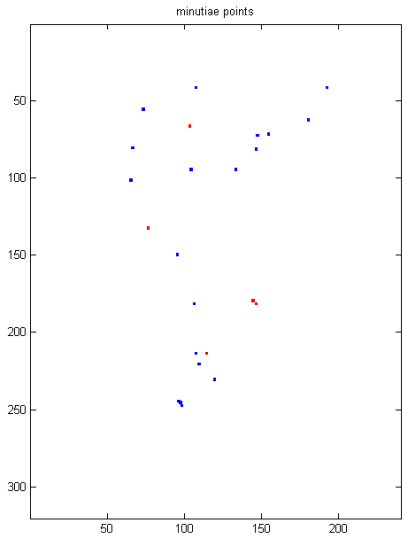

Figure 8.9: Minutiae Image after post-processing

As we can see from table 8.2 that removing boundary minutiae considerably reduced the number of false minutiae from minutiae extraction results.

Figure 8.10: Composite Image after post-processing

Table 8.1: Average Number of Minutiae before and after post-processing

DB Used	After Extraction	After Removing Spurious Ones	After Removing Boundary Minutiae
FVC2004DB4	218	186	93
FVC2004DB3	222	196	55

8.3.2 Reference Point Detection

For reference point extraction we used complex filters as described earlier. For a database size of 300, reference point was found with success rate of 67.66 percent.

8.4 Gender Estimation and Classification

8.4.1 Gender Estimation

Average ridge density was calculated along with minimum and maximum ridge densities shown in table 8.3. Mean ridge density was used to divide the database into two parts. This reduced database size to be searched by half. Based on the information available about the gender of enrolled student, we can apply our gender estimation algorithm which will further increase the speed of identification.

8.4.2 Classification

Fingerprint classification was performed on both original and enhanced images. Results were more accurate on the enhanced image. We used same algorithm as in sec 6.2 to classify the fingerprint into five classes - arch, left loop, right loop, whorl and

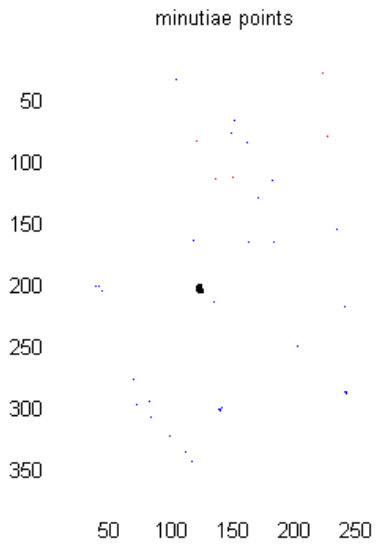

Figure 8.11: Plotted Minutiae with Reference Point(Black Spot)

Table 8.2: Ridge Density Calculation Results

Window Size Taken	Minimum Ridge Density	Maximum Ridge Density	Mean Ridge Density	Total Time Taken	Average Time Taken
36	6.25	9.50	7.87	193.76 sec	1.46 sec

unclassified. This classification was used to divide the database into five parts which would reduce the database to be searched to one-fifth and ultimately making this identification process five times faster. Results of classification are shown in table 8.4, 8.5 and 8.6.

8.5 Enrolling

At the time of enrolling personal details like name, semester, gender, age, roll number etc. were asked to input by the user and following features of fingerprint were saved in the database -

(1)Minutiae Set

(2)Key

(3)Ridge Density

(4)Class

Total and average time taken for enrolling fingerprints in database is shown in table

Table 8.3: Classification Results on Original Image

Class (1-5)	No. of Images
1	2
2	2
3	3
4	4
5	121

Table 8.4: Classification Results on Enhanced Image

Class (1-5)	No. of Images
1	8
2	3
3	3
4	6
5	112

8.7. All the personal details were stored in the MS Access database and were modified by running sql queries inside matlab. Fingerprint features were stored in txt format inside a separate folder. When txt file were used, the process of enrolling was faster as compared to storing the values in MS Access DB. It was due to the overhead of connections, running sql queries for MS Access DB.

8.6 Matching

Fingerprint matching is required by both verification and identification processes.

8.6.1 Fingerprint Verification Results

Fingerprint verification is the process of matching two fingerprints against each other to verify whether they belong to same person or not. When a fingerprint matches with the fingerprint of same individual, we call it *true accept* or if it doesn't, we call it *false reject*. In the same way if the fingerprint of different individuals match, we call it a *false accept* or if it rejects them, it is *true reject*. False Accept Rate (FAR) and False Reject Rate (FRR) are the error rates which are used to express matching trustability.

FAR is defined by the formula :

Table 8.5: Time taken for Classification

Image Taken	Average Time(sec)	Total Time(sec)
Original	0.5233	69.07
Enhanced	0.8891	117.36

Table 8.6: Time taken for Enrolling

No. of Images	Storage Type	Average Time(sec)	Total Time(hrs)
294	MS Access DB	24.55	2.046
60	MS Access DB	29.37	0.49
150	TXT files	15.06	1.255

$$FAR = \frac{FA}{N} * 100, \tag{8.1}$$

FA = Number of False Accepts, N = Total number of verifications

FRR is defined by the formula :

$$FRR = \frac{FR}{N} * 100, \tag{8.2}$$

FR = Number of False Rejects.

FAR and FRR calculated over six templates of Verifinger DB are shown in table 8.8. This process took approximately 7 hours.

8.6.2 Identification Results and Comparison with Other Matching techniques

Fingerprint identification is the process of identifying a query fingerprint from a set of enrolled fingerprints. Identification is usually a slower process because we have to search over a large database. Currently we match minutiae set of query fingerprint with the minutiae sets of enrolled fingerprints. In this project, we store key in the database at the time of enrolling. This key as explained in sec 5.3 helps in

Table 8.7: Error Rates

FAR	FRR
4.56	12.5
14.72	4.02

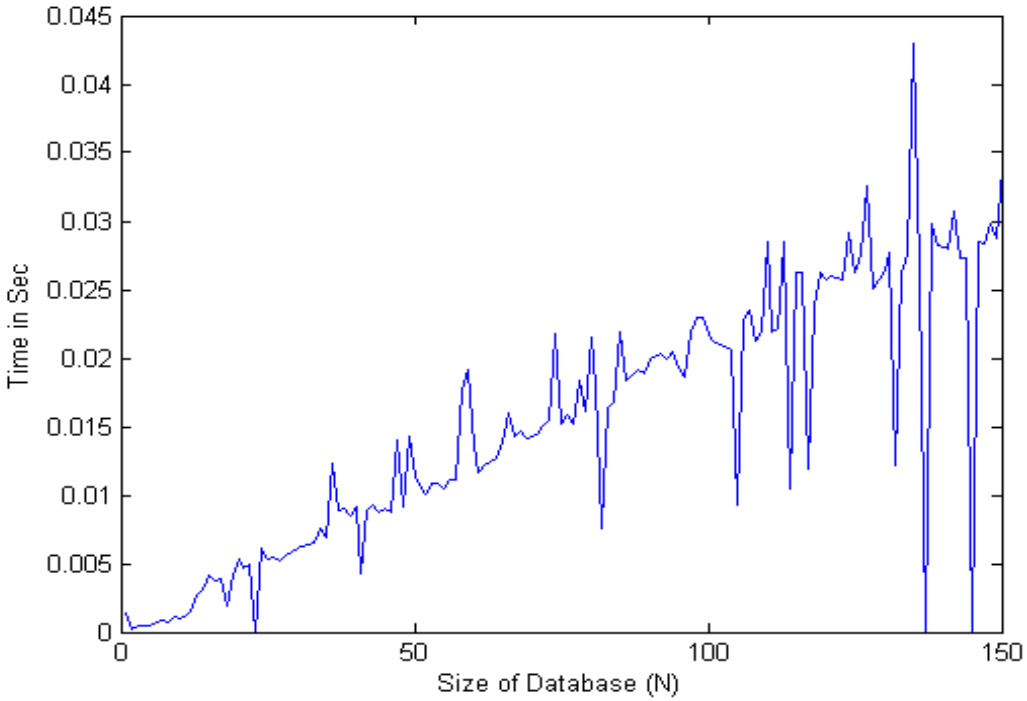

Figure 8.12: Graph: Time taken for Identification vs Size of Database(key based one to many identification)

reducing matching time over non-matching fingerprints. For non-matching enrolled fingerprints, we don't perform full matching, instead a key matching. Among one or many keys which matched in one iteration of one to many matching, we allow full minutiae set matching. Then if any full matching succeeds, we perform post matching steps. This identification scheme has lesser time complexity as compared to conventional n^2 one to one identification. Identification results are shown in table 8.9. The graph of time versus N is shown in figure 8.13. Here N is the index of fingerprint to be identified from a set of enrolled fingerprints. Size of database of enrolled fingerprints was 150. So N can vary from 1 to 150. The caverns in the graph indicated failed key matches while peaks indicated successful key matches as it took more time because full matching was also performed. As we can see that time was increasing linearly with size of database. This is so because size was 150 which is of the order of n. As the size will increase e.g. millions or thousands, the slope of graph will decrease

considerably and this can be predicted very well from our time complexity analysis. On large database, the graph will finally saturate to line approximately parallel to x-axis.

We also ran existing n^2 one to one identification technique on the same platform. As

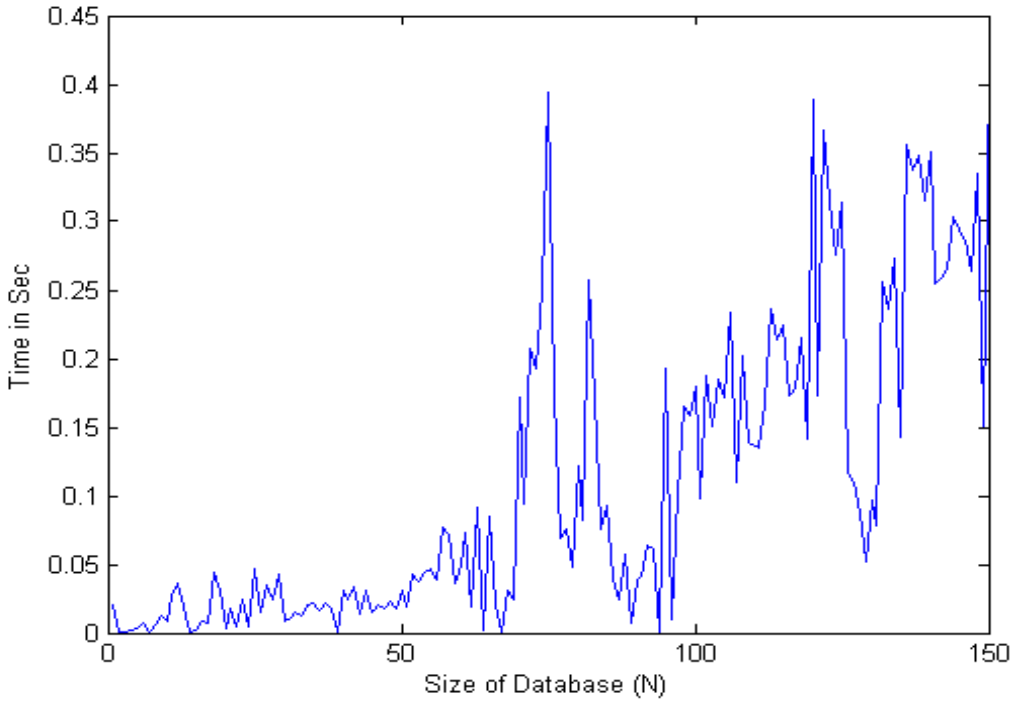

Figure 8.13: Graph: Time taken for Identification vs Size of Database (n^2 identification)

can be seen from figure 8.14, that time does not vary uniformly and linearly with size of database. This was because of less size of database and uneven values of n. Small values of n will result into less time elapsed for full minutiae match while large value of n will result into large elapsed time for full minutiae match. But on the graph for large databases, these imperfections will be negligible and we will get a linear graph in saturated state.

The graph of key based identification system was more uniform than that of n^2 matching based identification because in key based identification, we matched keys and size of keys was constant throughout the system. While in n^2 matching based identification, n may vary from 20 to 80 randomly and so n^2 will. As n^2 will change, matching

Table 8.8: Performance of ours and n^2 matching based identification techniques on a database of size 150

Total Matching Time	Average Matching Time	Processor Used	Type of Matching
0.1994 sec	1.33 ms	Core2Duo 1.83GHz	Exhaustive n^2 matching based One to One identification
0.0157 sec	0.105 ms	Core2Duo 1.83GHz	Key based One to Many identification

time will change. This gave rise to non-uniform graph. But this non-uniformity in graph will apparently vanish when size of database will be around thousands or millions.

8.7 Performance Analysis

As we can see from our fingerprint identification experiments, this key based identification takes very less time as compared to the existing n^2 matching based one to one identification. Results are shown in table 8.9. We draw an expected graph (figure 8.14) showing comparative results over database size of million fingerprints. Time taken by our identification technique will be roughly 0.000105*1000000=184.67 sec = around 3 minutes, while by one to one identification technique will be roughly 0.00133*1000000=1329.33 sec = around 1/3 hour. So when database is of the order of millions, our identification technique runs in minutes, while other identification techniques take hours to complete.

The matching results are so good that we can use this matching technique on the large databases like that of a country like India. India is working under her ambitious MNIC(Multipurpose National Identity Card) project. We can use our matching technique in identification system of MNIC project.

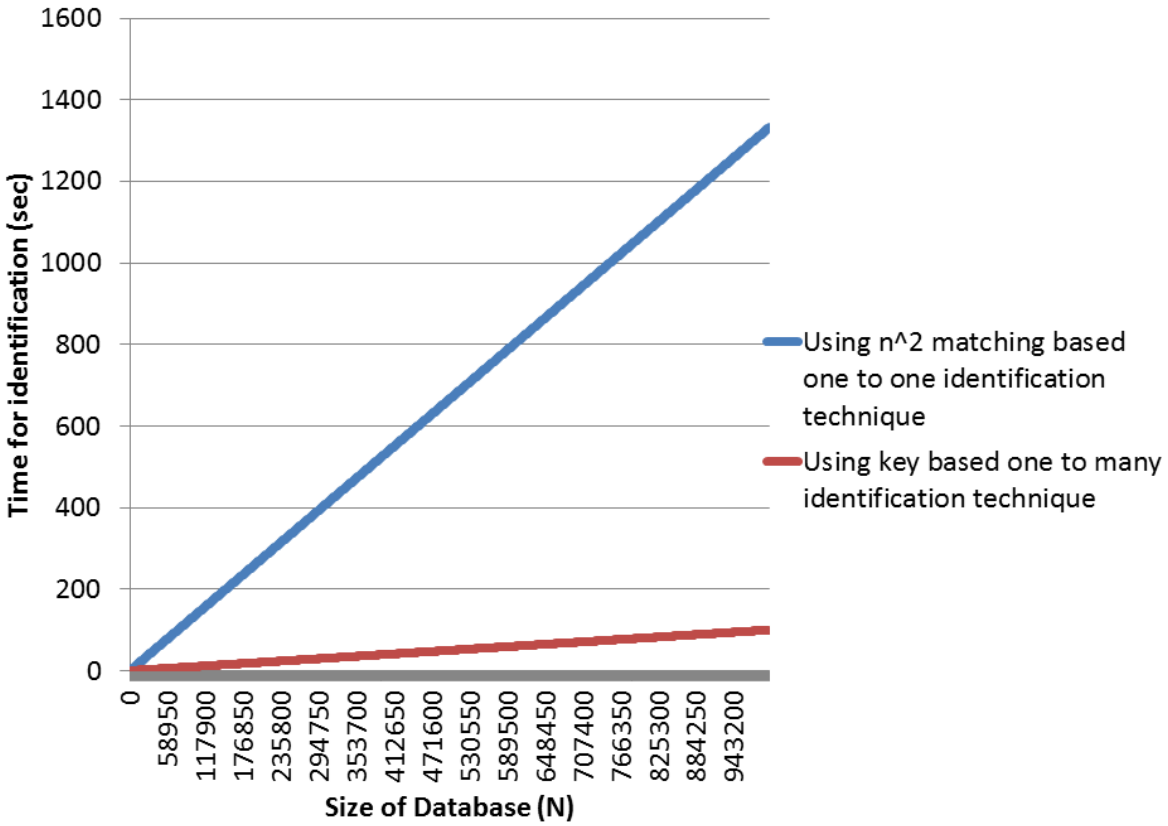

Figure 8.14: Expected Graph for comparison : Time taken for Identification vs Size of Database(1 million)

Chapter 9

Conclusion

This project mainly comprised of development of attendance management system and fingerprint identification system. Attendance management is very helpful in saving valuable time of students and teachers, paper and generating report at required time. This project presented a framework using which attendance management can be made automated and on-line. A general implementable approach to attendance management was proposed using LAN. Further, an idea for using portable devices alongwith wireless LAN or mobile 3G network was suggested.

Fingerprint Identification System used for student identification is faster in implementation than any other fingerprint identification systems. For fingerprint recognition, prevalent enhancement techniques like Gabor Filters[1], minutiae extraction[1] using Crossing Number concept followed by spurious and boundary minutiae removal, fingerprint classification[4], reference point detection[2], etc. are employed. Also, various new concepts are invented in this fingerprint identification system like gender estimation and key based one to many matching. Fingerprint classification and gender estimation are employed to partition the database in order to reduce search space. Key based matching made identification process very fast as compared to conventional identification techniques as shown by our identification analysis in section 8.6.2 and 8.7. We used FVC2004, FVC2000 and Verifinger databases for experiments. Time complexity of our identification system was O(n+N) which is far better than O(n^2N) of existing identification systems.

9.1 Outcomes of this Project

1. A Scientific approach was developed during project work.

2. Skills and self-confidence in coding and working with softwares like Matlab were developed.

3. An applicable attendance management system was designed for educational organizations. Ideas were presented for making whole system online using portable device and 3G mobile technology.

4. An improved and faster fingerprint identification system was developed for student identification purpose.

5. Various new algorithms like gender estimation, key based one to many matching were invented in this project.

6. Our identification system was compared with existing n^2 matching based identification systems. Our system took 0.0157 sec while n^2 matching based identification system took 0.1994 sec in worst case for database size of 150. It was then estimated that for a database of size millions, our system will take only around 3 minutes at maximum while existing one will take around half an hour for identification of an individual in worst case.

7. The future expectations from this project is to actually implement such system for one or more classes if sufficient funds are provided to us.

8. Our fingerprint identification system can be used in implementation of MNIC project of India.

Chapter 10

Future Work and Expectations

Regarding our fingerprint identification system, we are planning introduce more indexing techniques like ridge density tolerance etc. for making the search more faster. Also the key used will be more efficient when complex key will be used. We are trying to reduce matching error rates. Student attendance system is designed using LAN in this project. We have thought of using wireless LAN. The problem of using wireless LAN is that wireless devices are costly and implementation is hard because the wireless devices work in small area. Our institute NIT Rourkela is spread over large area so we may not use wireless network now. As an alternate, we may use mobile network (sec 2.5) which would be sufficiently suitable because now-a-days 3G network provides much speed. It would meet necessary throughput and data fetching speed requirements.

10.1 Approach for Future Work

1. Two computers connected via LAN and a fingerprint scanner will be used initially. One computer will serve the purpose of server for storing reports which may be MS Access, MS Excel or SQL/Oracle database. Other one will be storing the enrolled database, will have software for automatic attendance management and will be connected to USB fingerprint scanner.
2. Software for automatic attendance management that will run on nodes could either be developed in the Matlab or Java. We can run java code on matlab, also java is better for handling network communications. So except fingerprint related functions

(which are already coded in Matlab), rest of automatic attendance management software will be designed using java.

3. A website will be hosted on the server for online access to attendance reports. For this purpose, html, JSP or ASP dotnet would be used.

4. Fingerprint identification system will be improved further using more indexing techniques like ridge density tolerance etc.

5. Instead of using database available on internet, we would be using database of students.

Bibliography

[1] Raymond Thai. "Fingerprint Image Enhancement and Minutiae Extraction". Technical report, The University of Western Australia.

[2] Kenneth Nilsson and Josef Bigun. "Localization of corresponding points in fingerprints by complex filtering". Pattern Recognition Letters 24, page 2135 2144, October 2003.

[3] Vinod C. Nayak, Tanuj Kanchan, Stany W. Lobo, and Prateek Rastogi etc. "Sex differences from fingerprint ridge density in the Indian population". *Journal of Forensic and Legal Medicine*, 17(1):84 – 86, September 2007.

[4] Mary Jane and Aliman Manalo. "Development of a Fingerprint Classification Scheme For Improved Fingerprint Identification". Technical report, University of the Philippines, Diliman.

[5] N.K. Ratha, K. Karu, S. Chen, and A. K. Jain. "A Real-Time Matching System for Large Fingerprint Database". *IEEE Trans. PAMI*, 18(8):799 – 813, 1996.

[6] L. Hong, Y.Wan , and Anil K. Jain. "Fingerprint Image Enhancement: Algorithm and performance algorithm". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):777 – 789, May 1998.

[7] L. Hong. *"Automatic Personal Identification Using Fingerprints"*. PhD thesis, Michigan State University, 1998.

[8] C.J. Lee and S.D. Wang. "Fingerprint feature extration using Gabor filters". *Electronic Letters*, 35(4):288 – 290, 1999.

Appendix A

Matlab functions

These are few of many Matlab functions used in implementation of this project. For Fingerprint Enhancement, Minutiae Extraction, Key Extraction, Gender Estimation, One to many matching. Refer [4] for classification.

```
% To implement enhancement techniques on the image
% input argument :  im = image
% For function used in this function refer to


function [ thinned,binim,normalized,freq,medfreq,oriented
,gaborim,ridgefilterim2] = enhancement(im)

if nargin == 0
    im = imread('C:\Users\Rishabh\Desktop\Project\fingerprint
database\FVC2000\DB4_B\105_1.tif');
%   im = rgb2gray(imagenew);
end

show(im,1);

%Segmentation and Normalization
%mask is segmentation mask, normalized is image after normalization

blksze = 16; thresh = 0.1;
[normalized, mask] = segmentation(im, blksze , thresh );
show(normalized,2);

%orientation
[oriented, reliability] = ridgeorient(normalized,1,3,3);
%show(oriented ,3 );
plotridgeorient(oriented, 20, im, 3);

%ridge frequency
blksze = 36 ;
[freq , medfreq] = ridgefreq(normalized, mask, oriented, blksze, 5, 4.4, 15);

show(freq,4);

%Gabor Filter
[~,gabout1] = gaborfilternew(im,1.6,2.3,16,0);
[~,gabout10] = gaborfilternew(im,1.6,2.3,16,pi/16);
[~,gabout2] = gaborfilternew(im,1.6,2.3,16,pi/8);
[~,gabout11] = gaborfilternew(im,1.6,2.3,16,3*pi/16);
[~,gabout3] = gaborfilternew(im,1.6,2.3,16,3*pi/8);
[~,gabout12] = gaborfilternew(im,1.6,2.3,16,5*pi/16);
[~,gabout13] = gaborfilternew(im,1.6,2.3,16,7*pi/16);
[~,gabout14] = gaborfilternew(im,1.6,2.3,16,9*pi/16);
[~,gabout15] = gaborfilternew(im,1.6,2.3,16,11*pi/16);
[~,gabout16] = gaborfilternew(im,1.6,2.3,16,13*pi/16);
[~,gabout17] = gaborfilternew(im,1.6,2.3,16,15*pi/16);
[~,gabout4] = gaborfilternew(im,1.6,2.3,16,pi/4);
[~,gabout5] = gaborfilternew(im,1.6,2.3,16,pi/2);
[~,gabout7] = gaborfilternew(im,1.6,2.3,16,5*pi/8);
[~,gabout6] = gaborfilternew(im,1.6,2.3,16,3*pi/4);
[~,gabout8] = gaborfilternew(im,1.6,2.3,16,7*pi/8);
[~,gabout9] = gaborfilternew(im,1.6,2.3,16,pi);
```

```
gaborim=uint16(gabout1+gabout2+gabout3+gabout4+gabout5+gabout6+gabout7+gabout
8+gabout9+gabout10+gabout11+gabout12+gabout13+gabout14+gabout15+gabout16+gabo
ut17);
show(gaborim,5);

ridgefilterim2 = ridgefilter( gaborim , oriented, freq, 0.7,0.7, 1);
show(ridgefilterim2,6);

%binarization
binim = ridgefilterim2 > 0;
show(binim,7);
%thinning

thinned = bwmorph(binim,'thin',Inf);
show(thinned,8);
end
```

```
%% Describtion of Gabor filter Function :

%% I : Input image
%% Sx & Sy : Variances along x and y-axes respectively
%% f : The frequency of the sinusoidal function
%% theta : The orientation of Gabor filter

%% G : The output filter as described above
%% gabout : The output filtered image

function [G,gabout] = gaborfilternew(I,Sx,Sy,f,theta)

if isa(I,'double')~=1
    I = double(I);
end

for x = -fix(Sx):fix(Sx)
    for y = -fix(Sy):fix(Sy)
        xPrime = x * cos(theta) ;
        yPrime = y * cos(theta);
        G(fix(Sx)+x+1,fix(Sy)+y+1) = exp(-.5*((xPrime/Sx)^2 + (yPrime/Sy)^2))
*cos(2*pi*f*xPrime);
    end
end

Imgabout = conv2(I,double(imag(G)),'same');
Regabout = conv2(I,double(real(G)),'same');

gabout = sqrt(Imgabout.*Imgabout + Regabout.*Regabout);
```

```
% Minutiae Extraction
% input arg : thinned = thinned image ; refx,refy = reference point
% output arg : minu_count = no. of minutiae ; minutiae = minutiae matrix
function [minu_count,minutiae]= raymondthaiextraction(thinned,img,refx,refy)

% minutiae counter
minu_count = 1;
minutiae(minu_count, :) = [0,0,0,0];
% loop through image and find minutiae, ignore 10 pixels for border
for x=10:size(thinned, 1) - 10
    for y=10:size(thinned,2) - 10
        if (thinned(x, y) == 1) % only continue if pixel is white
            % calculate CN from Raymond Thai
            CN = 0;
            for i = 1:8
                CN = CN + abs (P(thinned, x, y, i) - P(thinned, x, y, i +
1));
            end
            CN = CN / 2;

            if (CN == 1) || (CN == 3)
                theta = FindTheta(thinned, x, y, CN);
                minutiae(minu_count, :) = [x, y, CN, theta];
                minu_count = minu_count + 1;
            end
        end % if pixel white
    end % for y
end % for x

% make minutiae image
minutiae_img = uint8(ones(size(thinned ,1),size(thinned, 2), 3)*255);

for i=1:minu_count - 1
    x1 = minutiae(i, 1);
    y1 = minutiae(i, 2);
    if (minutiae(i, 3) == 1)
        minutiae_img(x1, y1,:) = [255, 0, 0]; % red for ridge endings
        minutiae_img(x1+1, y1,:) = [255, 0, 0];minutiae_img(x1+1, y1+1,:) =
[255, 0, 0];
        minutiae_img(x1, y1+1,:) = [255, 0, 0]; % for making bigger spots...
    else
        minutiae_img(x1, y1,:) = [0, 0, 255]; % blue for ridge bijections
         minutiae_img(x1+1, y1,:) = [0, 0,255];minutiae_img(x1+1, y1+1,:) =
[0, 0, 255];
        minutiae_img(x1, y1+1,:) = [0, 0, 255]; % for making bigger spots...
    end
end % for loop through minu_count

% merge thinned image and minutia_img together
combined =  uint8(zeros(size(thinned ,1),size(thinned, 2), 3));
for x=1:size(thinned,1)
    for y=1:size(thinned,2)
        if (thinned(x,y))
            combined(x,y,:) = [255,255,255];
        else
```

```
            combined(x,y,:) = [0,0,0];
        end % end if
        if ((minutiae_img(x,y,3) ~= 0) || (minutiae_img(x,y,1) ~= 0))
            combined(x,y,:) = minutiae_img(x,y,:);
        end

    end % end for y
end % end for x

show(img,11);
subplot(1,1,1), subimage(minutiae_img), title('minutiae points')
show(img,12);
subplot(1,1,1), subimage(combined)
end
```

```
% function to remove spurious minutiae
% input arg:
% in : image
%
% output arg:
% finalList : minutiae list after removing spurious minutiae
% all_min_list : minutiae list before removing spurious minutiae

function     [finalList,all_min_list] =
realminutiae(in,minutiaeList,edgeWidth,minu_count,~)

[ridgeOrderMap,totalRidgeNum] = bwlabel(in);
finalList = minutiaeList;
numberOfMinutia=minu_count;
suspectMinList = [];


for i= 1:numberOfMinutia-1
for j = i+1:numberOfMinutia
d =( (minutiaeList(i,1) - minutiaeList(j,1))^2 + (minutiaeList(i,2)-
minutiaeList(j,2))^2)^0.5;
if d < edgeWidth
suspectMinList =[suspectMinList;[i,j]];

end;
end;
end;
[totalSuspectMin,dummy] = size(suspectMinList);
for k = 1:totalSuspectMin
typesum = minutiaeList(suspectMinList(k,1),3) +
minutiaeList(suspectMinList(k,2),3);
if typesum == 1
% branch - end pair
if
ridgeOrderMap(minutiaeList(suspectMinList(k,1),1),minutiaeList(suspectMinList
(k,1),2) ) ==
ridgeOrderMap(minutiaeList(suspectMinList(k,2),1),minutiaeList(suspectMinList
(k,2),2) )
finalList(suspectMinList(k,1),1:2) = [-1,-1];
```

```
finalList(suspectMinList(k,2),1:2) = [-1,-1];
end;
elseif typesum == 2
% branch - branch pair
if
ridgeOrderMap(minutiaeList(suspectMinList(k,1),1),minutiaeList(suspectMinList
(k,1),2) ) ==
ridgeOrderMap(minutiaeList(suspectMinList(k,2),1),minutiaeList(suspectMinList
(k,2),2) )
finalList(suspectMinList(k,1),1:2) = [-1,-1];
finalList(suspectMinList(k,2),1:2) = [-1,-1];
end;
elseif typesum == 0
% end - end pair
a = minutiaeList(suspectMinList(k,1),1:3);
b = minutiaeList(suspectMinList(k,2),1:3);
if ridgeOrderMap(a(1),a(2)) ~= ridgeOrderMap(b(1),b(2))
[thetaA,pathA,dd,mm] = getLocalTheta(in,a,edgeWidth);
[thetaB,pathB,dd,mm] = getLocalTheta(in,b,edgeWidth);
%the connected line between the two point
thetaC = atan2( (pathA(1,1)-pathB(1,1)),(pathA(1,2) - pathB(1,2)) );
angleAB = abs(thetaA-thetaB);
angleAC = abs(thetaA-thetaC);
if ( (or(angleAB < pi/3, abs(angleAB -pi)<pi/3 )) && (or(angleAC < pi/3,
abs(angleAC - pi) < pi/3)) )
finalList(suspectMinList(k,1),1:2) = [-1,-1];
finalList(suspectMinList(k,2),1:2) = [-1,-1];
end;
%remove short ridge later
elseif ridgeOrderMap(a(1),a(2)) == ridgeOrderMap(b(1),b(2))
finalList(suspectMinList(k,1),1:2) = [-1,-1];
finalList(suspectMinList(k,2),1:2) = [-1,-1];
end;
end;
end;

all_min_list=finalList;
A=finalList;
B = abs(A);
[~,loc] = ismember(B,A,'rows');
C = A(nonzeros(loc),:);
finalList=C;
end
```

```
% function to remove boundary minutiae
% input arg : edgewidth - distance between ridges , rest are as earlier
% output arg : new_min - minutiae after removing boundary ones

function new_min = remove_bounday_min( min_list , thinned ,edgewidth  )
num = size(min_list);
num=num(1,1);
s=20;new_min=[];
%limit = s*edgewidth*s/(36*2);limit = round(limit); %general formula
limit = 44; %an empirical value
for i = 1 : num
```

```
    matrix = find(thinned(min_list(i,1)-s/2:min_list(i,1)+s/2,min_list(i,2)-
s/2:min_list(i,2)+s/2)==1);
    total = size(matrix);
    total=total(1 , 1);

if total > limit
    new_min = [new_min;min_list(i,:)];
end
end
end
```

```
%     function for finding key
%     input arg : min_list - minutiae list
%                 refx,refy - location of reference point or centroid
%                 reforient - orientation of reference point
%     output arg : key - the extracted key


function key = extract_key(min_list, refx, refy, reforient)
key = [];
D=[];
for j= 1:10
[min_num,dummy]=size(min_list);
nearest = 1;
for i = 1:min_num
    %calculate distance between ref and points...
    D(i) = ((min_list(i,1)-refx)^2+(min_list(i,2)-refy)^2)^0.5;
    if (i-1 ~=0)
        if(D(i)< D(i-1))
            nearest = i;
        end
    end
end
key = [key; min_list(nearest,:)];
min_list(nearest,:)=[];
end
end
```

```
% Function to estimate gender using edgewidth of enrolled fingerprints
% input arg: edgewidth - mean ridge width of query fingerprint
% output arg: estimated_gender - 0/1 i.e. male/female.

function estimated_gender = gender_estimation( edgewidth )

DB = adodb_connect('PROVIDER=Microsoft.Jet.OLEDB.4.0; Data
Source=C:\Users\Rishabh\Desktop\Project\Student Attendence
system\adodb\myDatabase.mdb;');
sql = ' SELECT count(Serial) FROM MyImageDataTable ; ';
k=adodb_query(DB, sql);
k2=k.A; summ=uint16(k2);summ=summ-summ;k2=uint16(k2);
sql = 'SELECT ALL Roll,Reference_Point,Ridge_Distance FROM MyImageDataTable';
[Struct Table] = adodb_query(DB, sql);
%summx=sum(Struct.ridge_distance);
for i = 1:k2
   summ = summ+uint16(Struct.ridge_distance{i});
end
mean = summ./k2;
%%find female lower limit and male upper limit if real gender information
%%is stored in the database
DB.release;
if edgewidth <= mean
    disp('Probably a male candidate');
    estimated_gender=0;
end
if edgewidth > mean
        disp('Probably a female candidate');
        estimated_gender='1';
end
end
```

```
%Function To show classification results
function classification_results()
pat='C:\Users\Rishabh\Desktop\Project\fingerprint
database\2000DB4B+2004DB4A\Enrol';
           for i=1:22%templates
               for k=3:8%images in template

realpath=strcat(pat,'\');realpath=strcat(realpath,int2str(i));realpath=strcat
(realpath,' (');
           realpath=strcat(realpath,int2str(k));
           realpath=strcat(realpath,').tif');
           disp('Classification ');disp(i);
           image=imread(realpath,'tif');
           blksze = 16; thresh = 0.1;
[normalized, mask] = segmentation(image, blksze , thresh );
[oriented, reliability] = ridgeorient(normalized,1,3,3);
blksze = 36 ;
[freq , medfreq] = ridgefreq(normalized, mask, oriented, blksze, 5, 4.4, 15);
im=image;
[~,gabout1] = gaborfilternew(im,1.6,2.3,16,0);
[~,gabout10] = gaborfilternew(im,1.6,2.3,16,pi/16);
[~,gabout2] = gaborfilternew(im,1.6,2.3,16,pi/8);
[~,gabout11] = gaborfilternew(im,1.6,2.3,16,3*pi/16);
[~,gabout3] = gaborfilternew(im,1.6,2.3,16,3*pi/8);
[~,gabout12] = gaborfilternew(im,1.6,2.3,16,5*pi/16);
[~,gabout13] = gaborfilternew(im,1.6,2.3,16,7*pi/16);
[~,gabout14] = gaborfilternew(im,1.6,2.3,16,9*pi/16);
[~,gabout15] = gaborfilternew(im,1.6,2.3,16,11*pi/16);
[~,gabout16] = gaborfilternew(im,1.6,2.3,16,13*pi/16);
[~,gabout17] = gaborfilternew(im,1.6,2.3,16,15*pi/16);
[~,gabout4] = gaborfilternew(im,1.6,2.3,16,pi/4);
[~,gabout5] = gaborfilternew(im,1.6,2.3,16,pi/2);
[~,gabout7] = gaborfilternew(im,1.6,2.3,16,5*pi/8);
[~,gabout6] = gaborfilternew(im,1.6,2.3,16,3*pi/4);
[~,gabout8] = gaborfilternew(im,1.6,2.3,16,7*pi/8);
[~,gabout9] = gaborfilternew(im,1.6,2.3,16,pi);

gaborim=uint16(gabout1+gabout2+gabout3+gabout4+gabout5+gabout6+gabout7+gabout
8+gabout9+gabout10+gabout11+gabout12+gabout13+gabout14+gabout15+gabout16+gabo
ut17);
show(gaborim,5);
ridgefilterim2 = ridgefilter( gaborim , oriented, freq, 0.5, 0.5, 1);
tic;

%classification stage
A=imresize(ridgefilterim2,[300 300]);
A1 = flipud(A);
A2 = rot90(A1,-1);
warning off;
thresh = (min(min(A2)) + max(max(A2)))/2;
bw_print = A2>thresh;
bw = im2double(bw_print);
bw = fliplr(bw);
[theta_rad] = orient(bw);
```

```
theta_rad = ntheta(theta_rad);
theta_rad(theta_rad > 3*pi/4+0.1)=0;
theta_rad(theta_rad < 3*pi/4-0.1)=0;
binar = theta_rad;
[core_x1,core_y1] = find_core1(binar);
[class,deltax,deltay] = find_core_delta(core_x1,core_y1,binar);
disp('class ');
disp(class);


classi(i,k)=class;
timeio(i,k)=toc;
           end
           end
           classifytime=sum(sum(timeio));
avg=classifytime/132;
disp('class1');disp(sum(sum(classi==1)));
disp('class2');disp(sum(sum(classi==2)));
disp('class3');disp(sum(sum(classi==3)));
disp('class4');disp(sum(sum(classi==4)));
disp('class5');disp(sum(sum(classi==5)));
disp('time');disp(classifytime);
disp('avg time');disp(avg);
end
```

```
% function for for one to many identification

function totaltime=identifynew(p)

%using full match and key match
flag=0;

%p=input('Input p:');
tmp='C:\Users\Rishabh\Desktop\Project\Student Attendence system\Database\db\
(';
path=strcat(tmp,int2str(p));path=strcat(path,').tif');
im=imread(path,'tif');


%% Feature Extracion


%%Feature Extraction
        [thinned,binarized1,normalized,freq,medfreq,oriented,gaborim,ridgeim]
= enhancement(im);
        refx=1;refy=1;
        disp('');
        [minu_count,minutiae]= raymondthaiextraction(thinned,im,refx,refy);
        B = medfreq(medfreq~=0);
        M = mean(B(:));
        edgeWidth=1/M;
        x=minutiae(:,1);
        cx=round(mean(x));
        y=minutiae(:,2);
        cy=round(mean(y));
%         disp(cx);disp(cy);input('');
%         disp('Mean Ridge Distance ');
%         disp(edgeWidth);
        %estimated_gender = gender_estimation( edgeWidth );
        [finalListx,all_min_list] =
realminutiae(thinned,minutiae,edgeWidth,minu_count-1,thinned);%remove
spurious minutiae
        finalListq = remove_bounday_min( finalListx , thinned ,edgeWidth
);%remove boundary minutiae
        key=keyextract(finalListq,cx,cy);
        %disp('Total Number of Valid Minutiae ...');
        %plotnewminutiae(finalList ,image, thinned, refx,refy )
        %x=size(finalList);x=x(1,1);
        %disp(x);

%classification
A=imresize(im,[300 300]);
A1 = flipud(A);
A2 = rot90(A1,-1);
warning off;
thresh = (min(min(A2)) + max(max(A2)))/2;
bw_print = A2>thresh;
```

```
bw = im2double(bw_print);
bw = fliplr(bw);
[theta_rad] = orient(bw);
theta_rad = ntheta(theta_rad);
theta_rad(theta_rad > 3*pi/4+0.1)=0;
theta_rad(theta_rad < 3*pi/4-0.1)=0;
binar = theta_rad;
[core_x1,core_y1] = find_core1(binar);              %chng
[class,deltax,deltay] = find_core_delta(core_x1,core_y1,binar);

%%Matching
fmtime=0;
totaltime=0;
found=0;
successful_match=0;
successful_keys=0;
mtime=0;
keymatchtime=0;
k2=150;
j=1;
% M=input('Value of M?');
M=10;
while j <= k2
        if k2-j>=9
        limit=M;
        else
        limit=k2-j;
        end
    if flag==0
    for i=j:j+limit-1
        disp(i);
filenamex='key\';
filename1=strcat(filenamex,int2str(i));filename1=strcat(filename1,'key');file
name1=strcat(filename1,'.txt');
filename2=strcat(filenamex,int2str(i));filename2=strcat(filename2,'minutiae')
;filename2=strcat(filename2,'.txt');
filename3=strcat(filenamex,int2str(i));filename3=strcat(filename3,'edgewidth'
);filename3=strcat(filename3,'.txt');
filename4=strcat(filenamex,int2str(i));filename4=strcat(filename4,'class');fi
lename4=strcat(filename4,'.txt');
filename5=strcat(filenamex,int2str(i));filename5=strcat(filename5,'im');filen
ame5=strcat(filename5,'.mat');
keys=load(filename1,'key','-ASCII');
finalLists=load(filename2,'finalList','-ASCII');
edgewidths=load(filename3,'edgeWidth','-ASCII');
classs=load(filename4,'class','-ASCII');


%if class == classs

    %key match

    tic;
    match_status=key_match(keys,key);
    keymatchtime=keymatchtime+toc;
```

```
    if match_status == 1
        successful_keys=successful_keys+1;
        disp('Key Matching successful ');disp(i);tic;
    [match_score,full_match_status]=full_match(finalLists,finalListq);
    fmtime=fmtime+toc;
    if full_match_status == 1
        totaltime=keymatchtime+fmtime;
    disp('Fingerprint Identification Successful ');found=i;
    disp(i);disp('Time taken for Identification :');disp(totaltime);
    flag=1;break;


end

    end

    end
    end

j=j+M;
end

disp('Total successful key matches:');disp(successful_keys);
disp('Top match :');disp(found);
disp('Total time');disp(totaltime);
end
```

```
function [final]=fftenhance(image,f)

I = 255-double(image);

[w,h] = size(I);
%out = I;

w1=floor(w/16)*16;
h1=floor(h/16)*16;

inner = zeros(w1,h1);

for i=1:16:w1
   for j=1:16:h1
      a=i+15;
      b=j+15;
      F=fft2( I(i:a,j:b) );
      factor=abs(F).^f;
      block = abs(ifft2(F.*factor));

      larv=max(block(:));
      if larv==0
         larv=1;
      end;

      block= block./larv;
      inner(i:a,j:b) = block;
   end;
end;

final=inner*255;
final=histeq(uint8(final));
```

```
% for finding reference point
function [XofCenter,YofCenter] = corefinder(fingerprint)

global  matrice

x=[-16:1:16];
y=[-16:1:16];
dimx=size(x,2);
dimy=size(y,2);
% variance Gaussian, order complex filter
variance=sqrt(55);
order=1;
gamma=2;
filter_core=zeros(dimx,dimy);
filter_delta=zeros(dimx,dimy);
for ii=1:dimx
    for jj=1:dimy
        exponent=exp(-(x(ii)^2+y(jj)^2)/(2*variance^2));
        % filter core
```

```
        factor=x(ii)+i*y(jj);
        filter_core(ii,jj)=exponent*factor^order;
        % filter delta
        factor=x(ii)-i*y(jj);
        filter_delta(ii,jj)=exponent*factor^order;
    end
end
x=[-16:1:16];
y=[-16:1:16];
dimx=size(x,2);
dimy=size(y,2);
variance=sqrt(1.2);
filter=zeros(dimx,dimy);
for ii=1:dimx
    for jj=1:dimy
        exponent=exp(-(x(ii)^2+y(jj)^2)/(2*variance^2));
        filter(ii,jj)=exponent;
    end
end
% normalization
filter=filter/sum(sum(filter));
img=fingerprint;
img=double(img);
% complex field at 0 level
[gx,gy]=gradient(img);
num=(gx+i*gy).^2;
den=abs((gx+i*gy).^2);
pos=find(den);
num(pos)=num(pos)./den(pos);
z=zeros(size(img,1),size(img,2));
z(pos)=num(pos);
pos=find(den==0);
z(pos)=1;
if 1==0
    % complex field at  level 1
    z1=conv2fft(z,filter,'same');
    z1=dyaddown(z1,1,'m');
    num=z1;
    den=abs(z1);
    pos=find(den);
    num(pos)=num(pos)./den(pos);
    z1=zeros(size(z1,1),size(z1,2));
    z1(pos)=num(pos);
    pos=find(den==0);
    z1(pos)=1;

    z2=conv2fft(z1,filter,'same');
    z2=dyaddown(z2,1,'m');
    num=z2;
    den=abs(z2);
    pos=find(den);
    num(pos)=num(pos)./den(pos);
    z2=zeros(size(z2,1),size(z2,2));
    z2(pos)=num(pos);
    pos=find(den==0);
    z2(pos)=1;
    %*********************************
```

```
    %-------------------------------------
    % complex field at  level 3
    z3=conv2fft(z2,filter,'same');
    z3=dyaddown(z3,1,'m');
    num=z3;
    den=abs(z3);
    pos=find(den);
    num(pos)=num(pos)./den(pos);
    z3=zeros(size(z3,1),size(z3,2));
    z3(pos)=num(pos);
    pos=find(den==0);
    z3(pos)=1;

    z_f=conv2fft(z,filter_core,'same');
    z_f=abs(z_f);
    temp0=z_f;%temp0
    %-------------------
    z_1f=conv2fft(z1,filter_core,'same');
    z_1f=abs(z_1f);
    temp1=z_1f;%temp1
    %-------------------
    z_2f=conv2fft(z2,filter_core,'same');
    z_2f=abs(z_2f);
    temp2=z_2f;%temp2
    %--------------------
    z_3f=conv2fft(z3,filter_core,'same');
    z_3f=abs(z_3f);
    temp3=z_3f;%temp3
    %-------------------------------------------------------------------------


    [maximum_vector,position_vector]=max(temp3);
    [maximum,position]=max(maximum_vector);
    y_max=position;
    x_max=position_vector(position);
    maximum;

    x0=2*x_max;
    y0=2*y_max;

    dx=10;
    dy=10;

    positions=zeros(size(temp2));
    positions(max(1,x0-dx):min(size(temp2,1),x0+dx),max(1,y0-
dy):min(size(temp2,2),y0+dy))=1;
    temp2=temp2.*positions;

    [maximum_vector,position_vector]=max(temp2);
    [maximum,position]=max(maximum_vector);
    y_max=position;
    x_max=position_vector(position);
    maximum;

    x0=2*x_max;
```

```
    y0=2*y_max;

    dx=10;
    dy=10;

    positions=zeros(size(temp1));
    positions(max(1,x0-dx):min(size(temp1,1),x0+dx),max(1,y0-
dy):min(size(temp1,2),y0+dy))=1;
    temp1=temp1.*positions;

    [maximum_vector,position_vector]=max(temp1);
    [maximum,position]=max(maximum_vector);
    y_max=position;
    x_max=position_vector(position);
    maximum;

    x0=2*x_max;
    y0=2*y_max;

    dx=5;
    dy=5;

    positions=zeros(size(temp0));
    positions(max(1,x0-dx):min(size(temp0,1),x0+dx),max(1,y0-
dy):min(size(temp0,2),y0+dy))=1;
    temp0=temp0.*positions;

    [maximum_vector,position_vector]=max(temp0);
    [maximum,position]=max(maximum_vector);
    y_max=position;
    x_max=position_vector(position);
    maximum;

    disp('Coordinate x y');
    disp(x_max);
    disp(y_max);

    XofCenter=y_max;
    YofCenter=x_max;
    Outputprint=zeros(50);
end
```