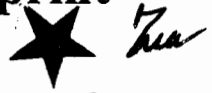


T00998

Personnel Authentication Through Fingerprint Biometrics



Acc. No. (FMS) T-998



Developed by
Nasir Rehan

Supervised by
Prof. Dr. Khalid Rashid

**Department of Computer Science
Faculty of Applied Sciences
International Islamic University, Islamabad
(2004)**

Doc. No. (FNU) T-998

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

In the name of ALMIGHTY ALLAH,
The most Beneficent, the most
Merciful.



**Department of Computer Science,
International Islamic University, Islamabad.**

Final Approval

Date: 30/5/04

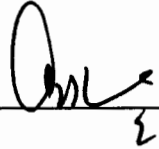
It is certified that we have read the thesis, titled “**Personnel Authentication Through Fingerprint Biometrics**” submitted by **Nasir Rehan** under University Reg. No. 23-CS/MS/01. It is our judgment that this thesis is of sufficient standard to warrant its acceptance by the International Islamic University, Islamabad, for the Degree of **Master of Science**.

Committee

External Examiner

Dr. Arshad Ali Shahid


Professor,
Department of Computer Science,
FAST- National University,
Islamabad.



Internal Examiner

Asim Munir

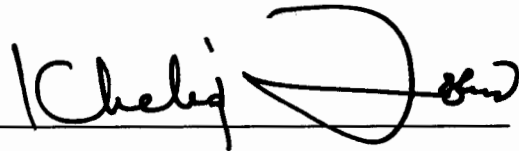
Lecturer,
Department of Computer Science,
International Islamic University,
Islamabad.



Supervisor

Prof. Dr. Khalid Rashid

Dean,
Faculty of Applied Sciences,
Faculty of Management Sciences,
International Islamic University,
Islamabad.



A thesis submitted to the
Department of Computer Science,
International Islamic University, Islamabad
as a partial fulfillment of the requirements
for the award of the degree of
MS Computer Science

Declaration

I hereby declare that this thesis & software neither as a whole nor as a part thereof has been copied out from any source. It is further declared that I have developed this software entirely on the basis of my personal efforts made under the sincere guidance of our teachers. No portion of the work presented in this thesis has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

Nasir Rehan
23-CS/MS/01

Dedication

To my Family, Friends and Teachers.

Abstract

This thesis describes the design and implementation of a fingerprint identification system for personnel authentication/identification. The purpose of the project is to look for a non-minutia approach for automated personnel identification system through fingerprint biometrics. Most of the existing automatic fingerprint identification and classification systems use representation scheme that are motivated by forensic experts in real time environment. They are either minutia based taking into account only the local anomalies (ridge ending & bifurcation) or correlation based techniques, which considers the direct use of gray-level information from the fingerprint image. Such systems are based on the fact that gray-level fingerprint image contains richer and discriminatory information than the minutiae locations therefore this gray level information can also be used as a distinguishing feature. So to yield a matching decision in a correlation based system the global pattern of ridges and valleys of template and input fingerprint images are aligned with each other. The task at hand is to come up with a fingerprint identification system that not only takes into account the local anomalies (i.e. minutia points) in the ridge structure but also caters the global pattern of ridges and furrows, inter-ridge distances and overall pattern of ridge flow. The resulting system should be practically feasible both in terms of efficiency and accuracy. Hence the filterbank-based representation matched all the desired specifications. The experimental results owing to modification in existing algorithms and application of performance tuning heuristics proved to be sufficiently good.

Project in Brief

Project Title:	Personnel Authentication Through Fingerprint Biometrics
Objective:	To develop an efficient fingerprint identification system for personnel authentication.
Undertaken By:	Nasir Rehan
Supervised By:	Prof. Dr. Khalid Rashid Dean, Faculty of Applied Sciences & Faculty of Management Sciences, International Islamic University, Islamabad.
Technologies Used:	Microsoft® Visual C++ 6.0
System Used:	Pentium® III, 550 MHz.
Operating System Used:	Microsoft® Windows® 2000 Professional
Date Started:	February, 2003
Date Completed:	December, 2003

Acknowledgements

All praise to the Almighty Allah, the most Merciful and the most Gracious, without whose help and blessings, I wouldn't have been able to complete the project.

I would like to thank Prof. Dr. Khalid Rashid, for taking keen interest in this project and accepting it under his supervision. During the course of the project he was kind enough to take time off from his busy routine and lend intellectual support required by the project.

My sincere thanks goes to my family and friends for their never fading faith, love and encouragement.

Nasir Rehan

Abbreviations

AFIS (P. 13)	Automated Fingerprint Identification System
KNN (P. 23)	K Nearest Neighbor
MATLAB (P. 68)	Matrix Laboratory
RB (P. 20)	Ridge Bifurcation
RE (P. 20)	Ridge Ending
AAD (P. 22)	Average Absolute Deviation

TABLE OF CONTENTS

<u>Ch. No.</u>	<u>Contents</u>	<u>Page No.</u>
1.	INTRODUCTION	1
1.1	BIOMETRICS	1
1.1.1	Requirments to Biometrics	2
1.1.2	Kinds of Biometric Systems	2
1.1.3	Biometric Applications	3
1.1.4	Biometric Process	4
1.1.5	Accepted and Studied Biometrics	4
1.1.6	Fingerprints	5
1.2	RESEARCH OBJECTIVE	6
2.	FINGERPRINT BIOMETRICS	7
2.1	FINGERPRINT CHARATERISTICS	7
2.1.1	Fingerprint Formation	7
2.1.2	Fingerprint Individuality	8
2.1.3	Fingerprint Features	8
2.1.3.1	Fingerprint Classification	9
2.1.3.2	Basic Ridge Patterns	10
2.1.3.3	Minutia Points	11
2.2	FINGERPRINT REPRESENTATION TECHNIQUES	13
2.2.1	Minutia Based Fingerprint Identification System	13
2.2.1.1	Orientation field Estimation	14
2.2.1.2	Segmentation	15
2.2.1.3	Binarization	16
2.2.1.4	Thinning	17
2.2.1.5	Minutia Detection	18
2.2.1.6	Postprocessing	19
2.2.1.7	Minutia Registration	19
2.2.1.8	Challenges in Minutia Based System	21
2.2.2	Filterbank Based fingerprint Identification System	21
2.2.2.1	Reference Point Location	22
2.2.2.2	Tessellation	22
2.2.2.3	Gabor Filtration	24
2.2.2.4	Feature Vector Generation	25
3.	CONCEPTUAL DESIGN AND	26
3.1	SYSTEM ARCHITECTURE	26
3.1.1	User Interface	26
3.1.2	System Database	26
3.1.3	Enrollment Module	26
3.1.4	Identification Module	27
3.2	SYSTEM COMPONENTS	27
3.2.1	Inputs	28
3.2.1.1	Fingerprint Image	28

3.2.1.2 Configuration File	28
3.2.1.3 Data Store	28
3.2.2 Processes	29
3.2.2.1 Feature Extraction	29
3.2.2.2 Classification	29
3.2.2.3 Feature Registration	29
3.2.2.4 Feature Matching	29
3.2.2.5 File Handler	29
3.2.3 Output	29
3.2.3.1 Graphical Display	30
3.3 DEVELOPMENT ENVIRONMENT	30
3.4 DESIGN METHODOLOGY	30
3.4.1 Object Oriented Design	30
3.4.2 Extentisibility of Data Structure and Functionality	30
4. SYSTEM DEVELOPMENT AND IMPLEMENTATION	31
4.1 STEPS FOR ENROLLMENT MODE	31
4.2 STEPS FOR IDENTIFICATION MODE	31
4.3 PROCESSES IN DETAIL	32
4.3.1 Feature Extraction	32
4.3.2 Fingerprint Classification	33
4.3.2.1 K Nearest Neighbor Classifier	33
4.3.3 Feature Registration	34
4.3.4 Feature Matching	35
4.4 ALGORITHM AND PSEUDO CODE	36
4.4.1 Algorithm for Reference Point Calculation	37
4.4.2 Pseudo Code for Reference Point Calculation	37
4.4.3 Algorithm for Tessellation	38
4.4.4 Pseudo Code for Tessellation	39
4.4.5 Algorithm for Normalization	40
4.4.6 Pseudo Code for Normalization	40
5. UTILITY CLASSES	42
5.1 CAFISALGO CLASS	42
5.2 CNORMALIZATION CLASS	45
5.3 CGABORFILTERING CLASS	46
5.4 CCLASSIFICATION CLASS	42
5.5 CMATCHING CLASS	49
5.6 CAFISSHARED CLASS	50
5.7 CFILEHANDLER CLASS	53
6. AFIS APPLICATION	55
6.1 SPLASH SCREEN	55
6.2 MAIN INTERFACE	56
6.3 AFIS TOOLBAR	56
6.4 IMAGE DISPLAY AREA	57
6.5 IMAGE VIEW TABS	57

6.6 LOG WINDOW TABS.....	58
6.6.1 Message Log.....	58
6.6.2 Database Detail Log.....	58
6.6.3 Close Matching Log.....	59
6.7 PERSON INFORMATION DIALOG BOX.....	59
6.8 OPTIONS DIALOG BOX.....	60
6.9 PERFORMING SYSTEM OPERATIONS VIA INTERFACE.....	60
7. SYSTEM EVALUATION.....	61
7.1 FINGERPRINT DATABASE COLLECTION.....	61
7.1.1 Training Set and Test Set.....	61
7.2 FAR AND FRR AS PERFORMANCE MEASURES.....	62
7.3 EXPERIMENTAL RESULTS.....	63
7.4 LIMITATIONS.....	64
7.4 CONSLUSION.....	65
BIBLIOGRAPHY AND REFERENCES.....	67

Chapter 1
Introduction

1. INTRODUCTION

Associating an identity with an individual is called identification (authentication). Automatic personal identification, in recent years have been greatly emphasized upon regarding the privacy and security of information utilized in today's world of electronic banking, e-commerce and access controls. Accurate automatic personal identification is now needed in a wide range of civilian applications involving the use of passports, cellular telephones, automatic teller machines, driver licenses, etc. An engineering approach to solve the problem of authentication of a person's identity is to reduce it to a problem of authentication of a concrete entity related to a person, for example:

- (i) Person's knowledge of some piece of information (Knowledge-based which includes passwords, PINs (Personal Identification Number), etc.).
- (ii) Person's possession (Token-based such as passport, driver license, ID Card, etc).

Traditional knowledge-based and token-based identifications are prone to fraud because PINs may be forgotten or guessed by an imposter and the tokens may be lost or stolen. Therefore, traditional knowledge-based and token-based approaches are unable to satisfy the security requirements of our electronically inter-connected information society. As an example, a large part of the annual \$450 million MasterCard credit card fraud is due to the identity. A perfect identity authentication system will necessarily have a biometric component [1]. Eventually, a foolproof identity authentication system will have all the three components (knowledge-based, token-based, and biometrics). In this thesis, we have only focused on the biometric component of an automatic identification system in general, and a fingerprint-based biometric identification system in particular.

1.1 Biometrics

The biometrics and biometry have been used historically to refer to a field statistical method used to analyze biological and environmental phenomena. The word "biometric" has recently been adopted by the Information Technology sector. Biometrics though being used in a wide array of applications but still a precise definition is difficult to establish. The most general definition of a biometric is:

"A physiological or behavioral characteristic that can be used to identify or verify the identity of an individual"[1].

Though there are numerous biometric measures that can be used to help derive an individual's identity, but as the above definition states, they are mostly classified into two distinct categories:

Physiological - These are biometrics, which are derived via direct measurement of a part of a human body. The most prominent and successful of these types of measures to date are fingerprints, face recognition, iris-scans and hand scans.

Behavioral - These extract characteristics are based on an action performed by an individual; they are an indirect measure of the characteristics of the human. The main feature of a behavioral biometric is the use of time as a metric. Established behavioral measures include keystroke-scan, speech patterns, etc.

Biometric methods used to identify people and animals have been used for thousands of years. Subconsciously we are able to differentiate our brothers and sisters by looking at their faces and recognizing subtle differences. Our brains are able to remember those people whom we have met before and differentiate them from strangers. Our brains are adapted to this process that we are able to uniquely identify a lost friend in a football stadium given enough time and patience. There is a medical condition for those people who are unable to remember the faces of those they meet; this condition is referred to as prosopagnosia. The ability to differentiate is not limited to people. Only Farmers are able to identify cattle by looking at a cow's marking and watching her behave in a herd. However, people who had never seen a cow would feel difficulty in identifying them, as they would all look alike to him. To alleviate the problems incurred in identifying someone without prior meeting or proving that a business transaction had occurred the Babylonians recorded the fingerprints in clay tablets. Whilst in China the Chinese began to leave fingerprints on important documents. So in short Biometrics is "Something you are" and cannot be misplaced on stolen.

1.1.1 Requirements to Biometrics

Biometric based identification system is subject to certain arguments, which need to be addressed before an actual implementation of that system takes place. These are as follows:

- i) **Universality** - all individuals should have it.
- ii) **Uniqueness** - no or very little probability that two persons are alike in terms of this characteristic.
- iii) **Permanence** - invariance with time.
- iv) **Collectability** - this biometric feature can be measured quantitatively.
- v) **Performance** - high identification accuracy rate within a short time span.
- vi) **Acceptability** - acceptance by people.
- vii) **Circumvention** - how easy it is to fool the system by fraudulent technique.

1.1.2 Kinds Of Biometric Systems

Two kinds of biometric systems are in operation: **Identification** and **Verification**.

In *identification-based systems*, a biometric signature of an unknown person is presented to the system. The system compares the new biometric signature with the database of biometric signatures of known individuals. On the basis of the comparison,

the system then reports or estimates the identity of the unknown person from this database. Systems that rely on identification include those that the police use to identify people from fingerprints and mug shots. Civilian applications include those that check for multiple applications by the same person for welfare benefits and driver's licenses.

In *verification-based systems*, a user presents a biometric signature and claim that a particular identity refers to him i.e. its his/her biometric signature. The algorithm either accepts or rejects the claim. Alternatively, the algorithm can return a confidence measurement of the claim's validity. Verification applications include those that authenticate identity during point-of-sale transactions or that of control access to computers or secure buildings.

Performance statistics for verification applications differ substantially from those for identification applications. The main performance measure for identification-based system is the system's ability to identify a biometric signature's owner. More specifically, the performance measure equals the percentage of queries in which the correct answer can be found in the top few matches. For example, law enforcement officers often use an electronic mug book to identify a suspect. The input to an electronic mug book is a mug shot of a suspect, and the output is a list of the top matches. Officers may be willing to examine only the top twenty matches. For such an application, the important performance measure is the percentage of queries in which the correct answer resides in the top twenty matches. The performance of a verification-based system, on the other hand is traditionally characterized by two error statistics: false-reject rate and false-accept rate. These error rates come in pairs; for each false-reject rate there is a corresponding false alarm. A false reject occurs when a system rejects a valid identity; a false accept occurs when a system incorrectly accepts an identity.

1.1.3 Biometric Applications

Biometrics is a rapidly evolving technology, which has been widely used in forensic application such as criminal identification and prison security. It has the potential to be widely adopted in a very broad range of civilian applications such as:

- i) **Banking Security** - such as electronic fund transfers, ATM security, check cashing, and credit card transactions,
- ii) **Physical Access Control** - such as airport access control,
- iii) **Information System Security** - like access to databases via login privileges,
- iv) **Government Benefits Distribution** - such as welfare disbursement programs,
- v) **Customs & Immigration** - such as *INS* Passenger accelerated Service System (INSPASS) which permits faster immigration procedures based on hand geometry,
- vi) **National ID Systems** - To provide a unique ID to the citizens and integrate different government services,
- vii) **Voter & Driver Registration** - To provide registration facilities to voters and drivers.

1.1.4 Biometric Process

Majority of biometric devices use a multi-stage process to identify or verify a person's identity. These can be broadly categorized as:

- **Data acquisition** - Process of acquiring the initial raw data from the respective sensing device (scanner, camera, etc.). This is the basis of all following computations, and the quality of the raw data has a severe impact on the performance of the complete system. For biometrics, this implies that paying attention to high quality capture data should enhance recognition performance. It is known, that significant variability can be introduced into subsequent samples of the same person due to noise or variability introduced by the human-machine interface of the acquisition device.
- **Preprocessing** - Preprocessing of data serves for quality enhancement of the acquired data. A series of steps such as mechanisms for noise removal from images or changing the data representation to a positional invariant one may belong to this step.
- **Feature extraction** - Together with the matching algorithm this step forms the basis for the systems recognition performance. The features and the respective matching algorithm must be optimized for each other. Biometric feature extraction implies the task of determining the personal characteristics that show the lowest intra-class and the highest inter-class variability simultaneously to guarantee, that variations in the original person's characteristics can be tolerated while still separating from obtrudes. In a practical biometric system the choice of the feature set (as well as the matching algorithm) can be further on determined by the application scenario and its computational constraints.
- **Classification** - This is the sequence of matching and decision. Matching is the process of calculating a similarity or dissimilarity measure based on a current feature representation of the biometrics data of a user and the respective reference data set. The matching process must compensate the statistical variability introduced into the biometrics signals during the acquisition process. Using statistical models of the data distributions the decision on the match can finally be performed.

1.1.5 Accepted & Studied Biometrics

Currently, there are mainly nine different biometric techniques that are either widely used or under investigation. These are:

- **Face**
- **Fingerprint**

- *Hand geometry*
- *Hand vein*
- *Iris*
- *Retinal pattern*
- *Signature*
- *Voice print*
- *DNA*

Of all the above-mentioned biometrics, DNA, signature, and fingerprints are accepted in court of law. Table 1.1 illustrates a brief comparison of these nine biometric techniques.

Biometrics	Universality	Uniqueness	Permanence	Collectability	Acceptability	Circumvention	Performance
Face	High	Low	Medium	High	Low	High	Low
Fingerprint	Medium	High	High	Medium	High	Medium	High
Hand Geometry	Medium	Medium	Medium	High	Medium	Medium	Medium
Hand Vein	Medium	Medium	Medium	Medium	Medium	Medium	High
Iris	High	High	High	Medium	High	Low	High
Retinal Scan	High	High	Medium	Low	High	Low	High
Signature	Low	Low	Low	High	Low	High	Low
Voice Print	Medium	Low	Low	Medium	Low	High	Low
Facial Thermogram	High	High	Low	High	Medium	High	High

Table 1.1 Comparison of Nine Biometrics

1.1.6 Fingerprints

Personal identification through the use of fingerprints has been in existence for thousands of years with individual impressions found on pottery which belong to generations that existed in past. The earliest recorded paper on the subject was published in 1684 and described the systematic study of ridge, pore and furrow structure in fingerprints [1]. In early 20th Century fingerprint recognition was accepted by law enforcement agencies as a valid method of personal identification and has become a standard for forensic testing, worldwide [2]. In early 1960's FBI, Home Office and Paris police departments all invested a large amount of effort in developing an automatic method of fingerprint identification. They had a tremendous success. Now automatic fingerprint identification is used by law enforcement agencies worldwide as a first line in forensic identification. In fact it has become so successful that it considered being the representative of the whole field of biometrics. Also involved are the economical benefits. The leading non-AFIS technology in the biometric market, fingerprint is poised to remain the leading technology through 2007. Because of the range of environments in which fingerprint can be deployed, its years of development, and the strong companies involved in the technology's manufacture and development, fingerprint revenues are

projected to grow from \$144.2m in 2002 to \$1,229.8m in 2007. Fingerprint revenues are expected to comprise approximately 30% of the entire biometric market.

1.2 Research Objective

The ability to identify an individual efficiently and accurately is an important task. Controlled environments such as banks, military installations and even airports demand the capability to quickly detect threats and provide different levels of access to different user groups. Recent events such as September 11, 2001 have brought biometrics a lot of attention as a method of identification.

Project Scope - Fingerprint identification system being the most developed and emphasized among the various practically employed biometric systems, has yet to make a long journey to prove its effectiveness in terms of speed and accuracy. Though various fingerprint identification algorithm have been developed and tested over the years but none had been able to yield satisfactory results. For example a state-of-the-art fingerprint identification & classification algorithm reports accuracies of 92.2% for five and 94.5% for four class classifications. These claims prove to be false when tested in the laboratory environment. Though this is a big achievement but in situations where the desired error rate is negligibly small, then there comes the need for further refinement of existing technologies or look for alternate ones. Our project is based on an emerging fingerprint identification technique i.e. Filterbank based fingerprint identification technique put forth by Salil Prabhaker in his PhD thesis [2]. This not only takes into account the local anomalies in the ridge structure (i.e. minutia points) but also the global pattern of ridges and furrows, inter-ridge distances, and overall patterns of ridge flow. This representation for oriented texture of fingerprints was inspired by Daugman's work on iris recognition and the success of the Gabor filterbank, as reported by Jain and Farrokhnia [3].

The project will enable us to establish an upper bound on the performance of filterbank based fingerprint systems, point out its deficiencies and come up with solution for increasing the efficiency and accuracy for fingerprint identification systems.

Chapter 2

Fingerprint Biometrics

2. Fingerprint Biometrics

Among all biometric techniques, fingerprint-based identification is the oldest method, which has been successfully used in numerous applications. Everyone is known to have unique, immutable fingerprints.

A smoothly flowing pattern formed by alternating ridges and valleys on the underside of finger is often referred to as fingerprints. Formation of ridges, valleys, and pores along ridges depend on the initial conditions (genetic and physical) of the embryonic development. A fingerprint is believed to be unique to each person as well as each finger. Fingerprints of even identical twins are different. The following is a quotation from a paper by Ashbaugh [1]:

"Closely related people have a greater chance of having similar genetic code that controls the appearance and development of volar pads. This results in pattern shapes that may appear to be passed from parent to child. However, even with those closely related, the physical and genetic variations seldom coincide to such a degree that all pattern areas display similar ridge configuration."

2.1 Fingerprint characteristics

2.1.1 Fingerprint Formation

Fingerprints are fully formed at about seven months of fetus development. Finger ridge configurations do not change throughout the life of an individual except due to accidents such as bruises or cuts on the fingertips. This property makes fingerprints a very attractive biometric identifier. Biological organisms in general are the consequence of the interaction of genes and environment. It is assumed that the phenotype is uniquely determined by the interaction of a specific genotype and a specific environment. Physical appearance and fingerprints are part of an individual's phenotype. In the case of fingerprints, the genes determine the general characteristics of the pattern. Fingerprint formation is similar to the growth of capillaries and blood vessels in angiogenesis. The general characteristics of the fingerprint emerge, as the skin on the fingertip begins to differentiate. However, the flow of amniotic fluids around the fetus and its position in the uterus change during the differentiation process. Thus, the cells on the fingertip grow in a microenvironment that is slightly different from hand to hand and finger to finger. The finer details of the fingerprints are determined by this changing microenvironment. Small differences in microenvironment are amplified by the differentiation process of the cells. There are so many variations during the formation of fingerprints that it would be virtually impossible for two fingerprints to be alike. Since the fingerprints are differentiated from the same genes, they will not be totally random patterns either. It could be said that the fingerprint formation process is a chaotic system rather than a random one.

2.1.2 Fingerprint Individuality

Fingerprint identification is based on two basic premises:

- i) **Persistence** - the basic characteristics of fingerprints do not change with time,
- ii) **Individuality** - the fingerprint is unique to an individual.

The validity of the first premise has been established by the anatomy and morphogenesis of friction ridge skin, while the second premise has been generally accepted to be true based on empirical results. The underlying scientific basis of fingerprint individuality has not been formally tested. As a result, fingerprint evidence is now being challenged in several court cases. A scientific basis for establishing fingerprint individuality will not only determine the admissibility of fingerprint identification in the courts of law but will also establish an upper bound on the performance of an automatic fingerprint identification system. The distinguishing nature of physical characteristics of a person is due to both the inherent individual genetic diversity within the human population as well as the random processes affecting the development of the embryo. It is only in identical twins where there may be more similarity of biometric properties. Typically, most of the physical characteristics such as body type, voice, and face are very similar for identical twins and automatic identification based on face and hand geometry is unlikely to distinguish them. It is, however, claimed that identical twins can be distinguished based on their fingerprints, retina, thermogram and iris patterns.

Monozygotic twins are a consequence of division of a single fertilized egg into two embryos. Thus, they have exactly identical DNA except for the generally undetectable micro mutations that begin as soon as the cell starts dividing. Fingerprints of identical twins start their development from the same DNA, so they show considerable generic similarity. However, identical twins are situated in different parts of the womb during development, so each fetus encounters slightly different intrauterine forces from their siblings. As a result, fingerprints of identical twins have different micro details, which can be used for identification purposes. Thus, there is anecdotal evidence that minutiae configurations are different in identical twins.

2.1.3 Fingerprint Features

Each person's fingerprint possesses unique characteristics that are contained within each pattern type. It is not simply the pattern or the combination of patterns that establishes identification, but the myriad of minute detail such as ridge endings, lakes, sweat pores, ridge structure and similar characteristics and their sequential relationship to each other that creates the uniqueness of each print.

There are two main types of characteristics that are used in identifying the fingerprints of a person, namely Global and Local Features.

Global Features - These are the characteristics that can be seen with the naked eye. It is here the concept of fingerprint classification takes place. They include Loop, arch, whorl (see figure 2.1) and the other groupings are based on the types of patterns in the fingerprint ridges. While not sufficient for identification, this categorization makes it easier to search large databases of fingerprints. Little description regarding fingerprint classification should be taken into consideration.

2.1.3.1 Fingerprint Classification

Fingerprint classification provides an important indexing mechanism in a fingerprint database. An accurate and consistent classification can greatly reduce the fingerprint matching time for a large database. In 1899, Edward Henry and his two assistants established the "*Henry System*" of fingerprint classification [4]. The Henry system classifies fingerprints into three main categories being *Arch*, *Loop* & *Whorl* as shown in Figure 2.1.

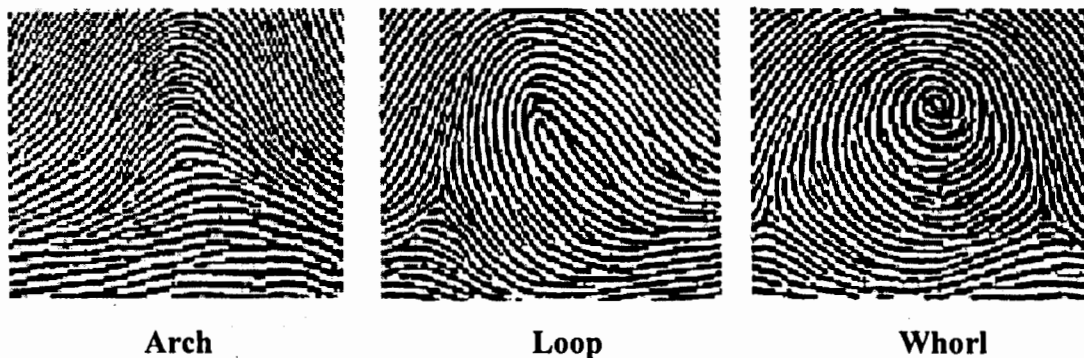


Figure 2.1 Three Main Classes of Fingerprint

Each of these categories is then further divided resulting in a total of more than twenty categories. Due to small interclass separability among these fingerprint types, it is extremely difficult to design twenty-class classifier with high accuracy. As a result, most automatic systems reduce the number of fingerprint types to a subset of classes defined in the Henry system. For example, the commercial systems typically provide ulnar, radial loops, accidental, whorl, double loop, and arch classification while academic institutes have typically concentrated on a five-class classification that includes whorl, left loop, right loop, arch, and tented arch as shown in figure 2.2.

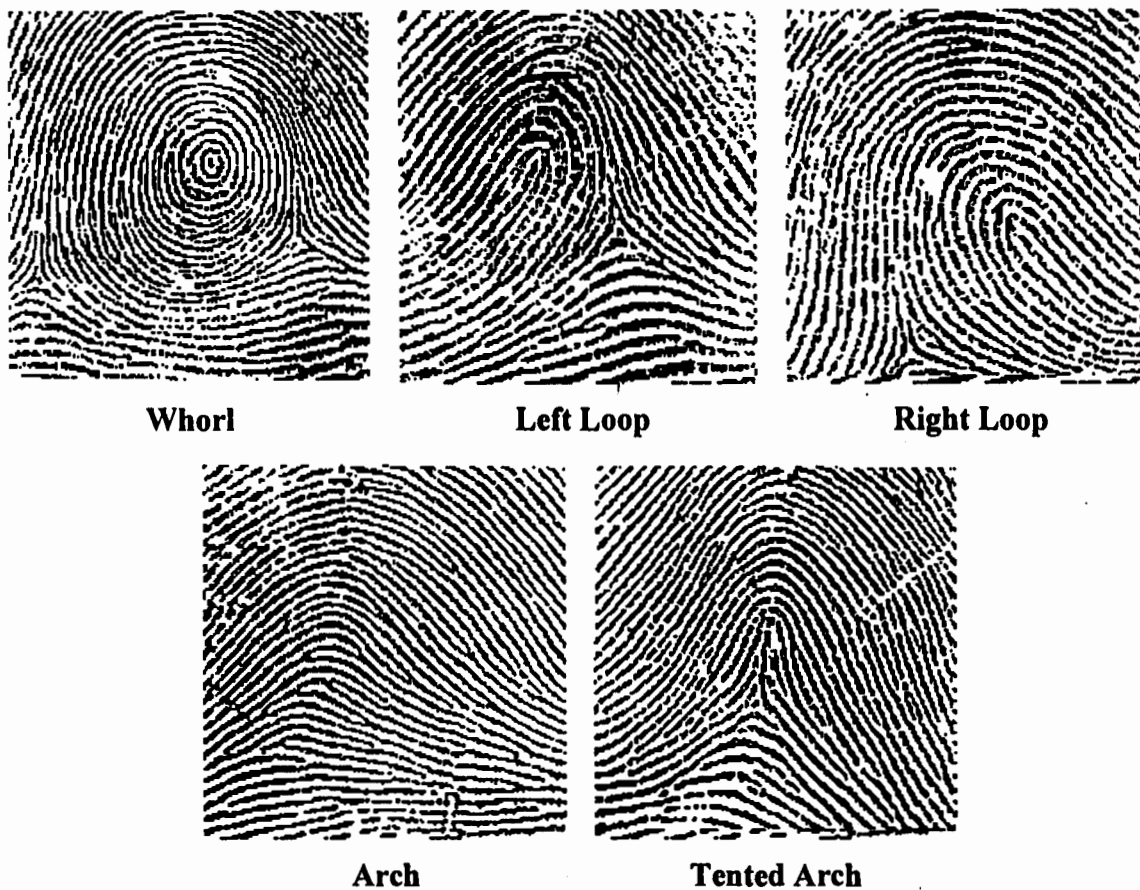


Figure 2.2 Five Fingerprint Classes Used in Automated Fingerprint Identification Systems

The natural proportion for occurrence of these five major classes of fingerprints is 0.3252, 0.3648, 0.1703, 0.0616, and 0.0779 for whorl, right loop, left loop, arch, and tented arch respectively.

Within the global categories are sub-characteristics that help to further distinguish one print from another.

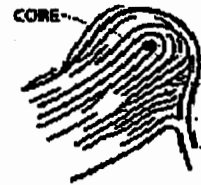
2.1.3.2 Basic Ridge Patterns

Loop, arch, whorl and other groupings are based on the types of patterns in the fingerprint ridges. While not sufficient for identification, this categorization makes it easier to search large databases of fingerprints. A brief description of these patterns is provided below:

Pattern Area - Part of the fingerprint that contains all the global features. One significant difference between particular fingerprint recognition algorithms is that some use the entire fingerprint for analysis and identification, not just the pattern area.



Core Point - Located at the approximate center of the finger impression that is used as a reference point for reading and classifying the print.



Delta - Point on the first bifurcation, abrupt ending ridge, meeting of two ridges, dot, fragmentary ridge, or any point upon a ridge at or nearest the center of divergence of two type lines, located at or directly in front of their point of divergence.



Type Lines - The two innermost ridges that start parallel, diverge and surround or tend to surround the pattern area. Often they are broken; the ridge immediately outside that line is considered to be its continuation.



Ridge Count - The number of ridges in the Pattern Area. To establish the ridge count, an imaginary line is drawn from the Delta and the Core and each ridge that touches this line is counted.



Local Features – These are the minutia points. It is possible for two or more individuals to have identical global features but still have different and unique fingerprints because they have local features - minutia points - that are different from those of others.

2.1.3.3 Minutia Points

Fingerprint ridges are not continuous or straight ridges. Instead they are broken, forked, changed directionally, or interrupted. The points at which ridges end, fork and change are called minutia points, and these minutia points provide unique, identifying information.

There are five different characteristics of minutia points in fingerprints:

- 1) **Type** - There are a number of types of minutia points. The most commonly used minutia types in automatic fingerprint identification systems are ridge endings and ridge bifurcations. These along with others are defined as:

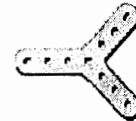
a) **Ridge Ending** -- occurs when a ridge ends abruptly.

A



b) **Ridge Bifurcation** -- the point at which a ridge divides into two or more branches.

B



c) **Ridge Divergence** -- the spreading apart of two lines which have been running parallel or nearly parallel.

C



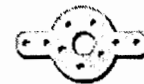
d) **Dot or Island** -- a ridge that is so short that it appears as a dot.

D



e) **Enclosure** -- a ridge that divides into two and then reunites again creating an enclosed area of ridge-less skin.

E



f) **Short Ridge** -- an extremely short ridge, but not so short that it appears as a Dot or Island

F



- 2) **Orientation** -- Each minutia point faces in a particular direction. This is the Orientation of the minutia point.

- 3) **Spatial Frequency** -- Spatial frequency refers to how far apart the ridges are in the neighborhood of the minutia point.
- 4) **Curvature** -- Refers to the rate of change of ridge orientation.
- 5) **Position** -- Refers to its x, y location, either in an absolute sense or relative to fixed points like the Delta and Core points.

2.2 Fingerprint Representation Techniques

With the passage of time many popular fingerprint representation schemes have evolved from an intuitive system developed by forensic experts who visually match the fingerprints. These schemes are either based on predominantly local landmarks (i.e. minutiae based fingerprint identification systems) or exclusively global information (correlation based fingerprint identification systems). The minutiae based automatic identification techniques first locate the minutiae points and then match their relative placement in a given finger and the stored template. A good quality inked fingerprint image contains between 60 to 80 minutiae, but different fingerprints and acquisition of the same finger at different time interval have different numbers of minutiae. The matching algorithm is based on inexact graph matching techniques. The point pattern-based representation considers the minutiae points as a two dimensional pattern of points. Correlation based technique considers direct use of gray-level information from the fingerprint image. Since a gray-level fingerprint image contains much richer, more discriminatory information than only the minutiae locations, the gray level information can also be used as distinguishing feature. The global pattern of ridges and valleys are aligned during decision level matching between template and input fingerprint image.

Most of the existing automatic fingerprint identification and classification systems use representations that are motivated by representations used by forensic experts i.e. minutia based systems. So a brief description of most commonly used minutia based system is provided and then filterbank-based technique, which is the core of this particular AFIS system, is also explained.

2.2.1 Minutia-Based Fingerprint Identification System

Most fingerprint representation systems follow a minutiae-based approach. A minutiae-based fingerprint identification system first extracts the minutiae points (ridge endings, bifurcation) from the fingerprint images. Then, the decision is based on the correspondence of the minutiae types and locations between the two fingerprint images (i.e. template fingerprint image and current input fingerprint image). Figure 2.3 illustrates the most commonly used minutia points.



Figure 2.3 Two most widely used minutia features
 a) Ridge Ending b) Ridge Bifurcation

Minutiae-based fingerprint identification systems use a large number of successive processing steps. In general, the following steps can be identified in a minutiae-based system as:

- *Orientation field estimation*
- *Segmentation*
- *Fingerprint image binarization*
- *Thinning*
- *Minutiae detection/extraction*
- *Postprocessing*
- *Minutiae Registration*

A brief description of the above-mentioned processing steps is given below:

2.2.1.1 Orientation Field Estimation

The orientation field of a fingerprint image represents the directionality of ridges in the fingerprint image. A number of methods have been proposed to estimate the orientation field of fingerprint images. The orientation is determined by dividing the fingerprint image into a number of non-overlapping blocks (preferably of size 10) and an orientation representative of the ridges in the block is assigned to the block based on the analysis of grayscale gradients within the block. This is achieved by applying the least mean square estimation algorithm. The complete steps for the orientation estimation algorithm [5] are illustrated by figure 2.4. The information thus obtained is actually the angle or direction, which is associated with the minutiae that are detected at later stages.

(a) Divide the input fingerprint image into blocks of size $W \times W$.

(b) Compute the gradients G_x and G_y at each pixel in each block [4].

(c) Estimate the local orientation at each pixel (i, j) using the following equations [28]:

$$V_x(i, j) = \sum_{u=i-\frac{W}{2}}^{i+\frac{W}{2}} \sum_{v=j-\frac{W}{2}}^{j+\frac{W}{2}} 2G_x(u, v)G_y(u, v), \quad (1)$$

$$V_y(i, j) = \sum_{u=i-\frac{W}{2}}^{i+\frac{W}{2}} \sum_{v=j-\frac{W}{2}}^{j+\frac{W}{2}} (G_x^2(u, v) - G_y^2(u, v)), \quad (2)$$

$$\theta(i, j) = \frac{1}{2} \tan^{-1} \left(\frac{V_x(i, j)}{V_y(i, j)} \right), \quad (3)$$

where W is the size of the local window; G_x and G_y are the gradient magnitudes in x and y directions, respectively.

(d) Compute the consistency level of the orientation field in the local neighborhood of a block (i, j) with the following formula:

$$C(i, j) = \frac{1}{N} \sqrt{\sum_{(i', j') \in D} |\theta(i', j') - \theta(i, j)|^2}, \quad (4)$$

$$|\theta' - \theta| = \begin{cases} d & \text{if } (d = (\theta' - \theta + 360) \bmod 360) < 180, \\ d - 180 & \text{otherwise,} \end{cases} \quad (5)$$

where D represents the local neighborhood around the block (i, j) (in our system, the size of D is 5×5); N is the number of blocks within D ; $\theta(i', j')$ and $\theta(i, j)$ are local ridge orientations at blocks (i', j') and (i, j) , respectively.

(e) If the consistency level (Eq.(5)) is above a certain threshold T_c , then the local orientations around this region are re-estimated at a lower resolution level until $C(i, j)$ is below a certain level.

Figure 2.4 Hierarchical Orientation Field Estimation Algorithm

2.2.1.2 Segmentation

The task of a fingerprint segmentation algorithm is to decide which part of the image belongs to the foreground, originating from the contact of a fingertip with the sensor, and which part to the background, which is the noisy area at the borders of the image.

Accurate segmentation is especially important for reliable extraction of features like minutiae and singular points. Therefore, the main goal of the segmentation algorithm is to discard the background, and thus reduce the number of false features. The simplest approach segments the foreground by global or adaptive thresholding. A novel and

reliable approach to segmentation by Ratha et al. [6] exploits the fact that there is significant difference in the magnitudes of variance in the gray levels along and across the flow of a fingerprint ridge. Typically, the block size for variance computation spans 1-2 inter-ridge distance. An algorithm for the segmentation of fingerprints by Asker M and Sabih H [7] gracefully segments the fingerprint image. The method uses three pixel features which being coherence, mean and variance.

2.2.1.3 Binarization

Binarization is a method of transforming grayscale image pixels into either black or white pixels corresponding to a ridge and valley respectively. The process can be carried out using a multitude of techniques. Binarization is relatively easy to achieve as compared with other image processing techniques. The approaches to binarized image use either global or adaptive thresholding. The various methods of Binarization are explained in [8]. Description of the two most important Binarization techniques is as follows:

2.2.1.3.1 Global Thresholding

In global thresholding, the main or highest black and white pixel values within a grayscale image are determined. The pixels that range or lie within these two extremes or highest values is used to separate the black and white colors. Global binarization involves the formulation of a histogram consisting of the number of pixels versus the pixel value.

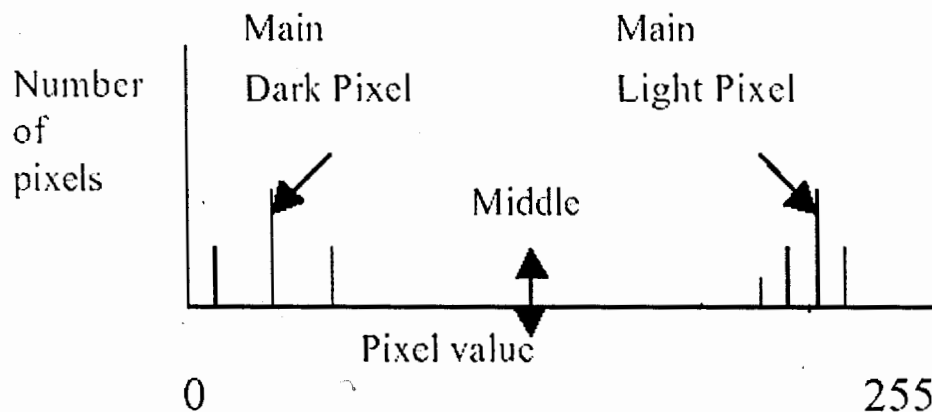


Figure 2.5 Histogram for calculation of most numerous light and dark pixel values.

The two major peaks found are assumed to be the most commonly used dark pixel and the most common light pixel. The middle range pixel will then be used to discriminate between black and white pixels for binarization.

2.2.1.3.2 Adaptive Thresholding

This is also known as contrast enhancement binarization. The method involves passing a low pass filter over the image and using the resulting grayscale pixel number to discriminate between a black or white pixel. The low pass filter does not process edges that are one pixel wide in the image. Low-pass filtering involves a spatial convolution process within a window. These windows can be quite large. The window size used is 3*3 pixels wide.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Figure 2.6 Convolution matrix used for low-pass filtering

The image is convolved with this filter, producing a new image. The process is repeated on the entire image, row by row until a new, binarized image is formed.

2.2.1.4 Thinning

Various morphological operations are performed on binarized images. The most important of them include thinning and spur removal. A thinned fingerprint image is one in which each ridge is only one pixel in width. Thinning is achieved by successive deletion of pixels from different sides of each image. (North, South, East, West). Each of the four sides are eroded away according to some set template.

NORTH	SOUTH	EAST	WEST
0 0 0	1 1 X	0 X 1	X X 0
X 1 X	X 1 X	0 1 1	1 1 0
X 1 1	0 0 0	0 X X	1 X 0
X 0 0	X 1 X	0 0 X	X 1 X
1 1 0	0 1 1	0 1 1	1 1 0
X 1 X	0 0 X	X 1 X	X 0 0

Figure 2.7 Eight matrices used for Thinning

If the image template matches, the middle pixel is removed. The two north images are processed, and then the other compass points are overlaid. Once all eight matrices have been sampled on the entire image, the process is repeated on the newly formed image. Processing only stops when no more points can be deleted. Similarly spur operation is performed on thinned image to remove bifurcations and ends caused by

thinning. It may also remove real end points from their original locations. A detailed explanation of how successive deletion works can be found in *A. Rosenfeld & A.C. Kak's* textbook [9].

2.2.1.5 Minutia Detection

Once the thinned ridge map is available, the minutia location is not a tiresome job then. The technique uses a sample window, 3 x 3 pixels wide, to detect key features such as ridge endpoints and ridge bifurcation. Ridge pixels with one ridge pixel neighbor are identified as ridge endings and those with three ridge pixel neighbors are identified as ridge bifurcations.

CN	Characteristic
0	Isolated Point
1	End Point
2	Continuing Point
3	Bifurcation Point

Figure 2.8 Table for checking characteristics of ridges

P ₄	P ₃	P ₂
P ₅	P	P ₁
P ₆	P ₇	P ₈

The matrix checks for pixel values in each neighborhood of pixel P₁ and then compares the outcome with the characteristics table.

Figure 2.9 Matrix for traversing through thinned image

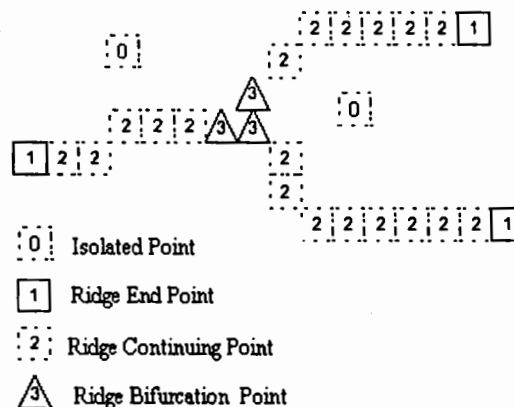


Figure 2.10 Thinned ridged map with minutiae being marked

However, the entire minutia thus detected are not genuine due to image processing artifacts and the noise in the fingerprint image. The excess bifurcations and endpoints associated with noise must be removed to maintain accuracy levels.

2.2.1.6 Postprocessing

In this stage, typically, genuine minutiae are gleaned from the extracted minutiae using a number of heuristics. For instance, too many minutiae in a small neighborhood may indicate noise and they could be discarded. Very close ridge endings oriented anti-parallel to each other may indicate spurious minutia generated by a break in the ridge either due to poor contrast or a cut in the finger. Two very closely located bifurcations sharing a common short ridge often suggest extraneous minutia generated by bridging of adjacent ridges as a result of dirt or image processing artifacts.

Minutia deletion is carried out on certain heuristics. A minutia point is selected and all those minutiae that lie within the specified radius of that selected minutia point are marked. And finally all those marked minutia points including the center (selected) minutia point are all deleted.

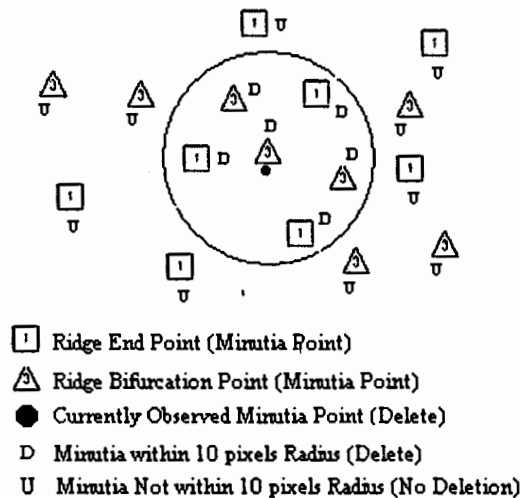


Figure 2.11 Depicting minutia deletion strategy

2.2.1.7 Minutia Registration

Feature extraction stage in minutia-based system is concluded by the registration of the acquired minutia points. Many features regarding a particular minutia point are recorded in the database. Database include information like total number of minutiae points detected, position (co-ordinates) of each minutia point, minutiae type (ridge ending, bifurcation), it's angle (direction) and most important of all the minutia's position relative to other minutia detected in the same fingerprint. A fingerprint can have up to 60-80 minutiae detected, and a valid match is generally accepted between two fingerprints if both the prints have 8 to 17 common minutia points.

Total Minutiae Points	Minutia Position (x, y)	Minutia Angle/ Direction	Minutia Type (RE/RB)
-----------------------	-------------------------	--------------------------	----------------------

Figure 2.12 Most Common Minutiae Information in Database

A microscopic approach, called minutia matching is commonly used for matching in minutia based fingerprint systems. Each minutia has a position relative to other minutia in a fingerprint. These positions are recorded as a template. For the purpose of matching, the positions of minutia in an enrolled template are matched against those of a current user for authenticating his fingerprint.

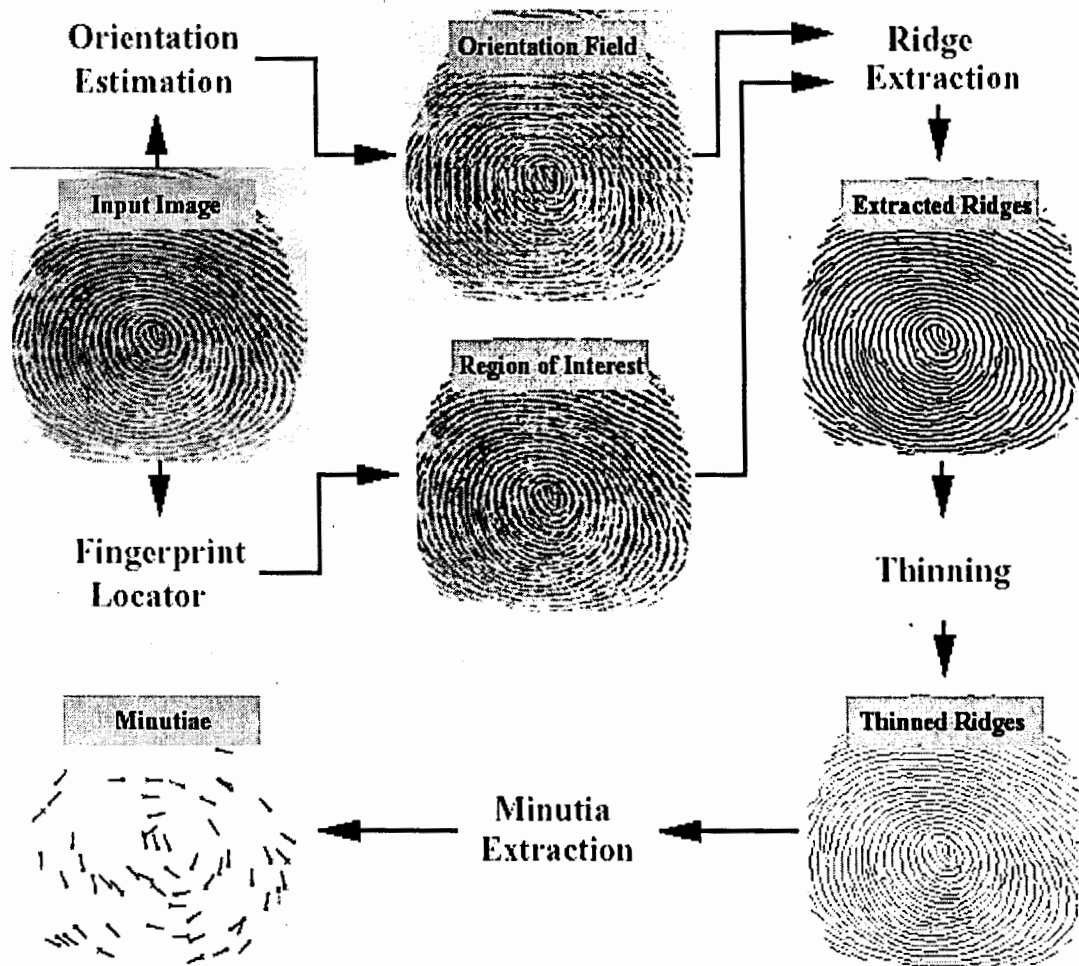


Figure 2.13 Flowchart of Minutia Extraction Algorithm

2.2.1.8 Challenges In Minutia-Based System

These are various factors or constraints, whose net affect adversely hinders the efficiency of minutia-based systems.

The segmentation of fingerprint-textured image still remains a difficult problem in fingerprint identification systems. Without significant segmentation, the system's efficiency is questioned due to the false minutia detected off the original fingerprint texture. The most successful approach towards fingerprint segmentation includes Gabor Filters, which is the prime characteristic of Filterbank based fingerprint system, but this has its own computational constraint which will affect the time within which the system generates the matching result.

The thinning algorithms haven't yet been designed to perfection. The correspondence between the original fingerprint image and the resulting thinned image has to be perfect. As it is from this stage the minutia points are determined and stored for future use in matching stage. The contemporary thinning algorithms engulf many true minutia points and generate many false minutia points.

The heuristics employed for the deletion of minutia points further add to the deficiency of minutia-based systems. Many of the valid minutia points are deleted and certain false minutiae points survive for registration in the system database.

As a result of the limited information content of the minutiae representation, non-minutiae representations of fingerprints should be explored. The main aim of the project is to follow a novel non-minutiae representation for fingerprints that combines both the global and the local information present in a fingerprint. The representation under consideration i.e. Filterbank based fingerprint representation system is based on considering the fingerprint images as oriented textures and is very different from the representations used by the forensic experts as it is more amenable to automatic systems (in terms of matching speed and storage size). The performance of this representation is evaluated for both fingerprint classification and matching applications.

2.2.2 Filterbank Based Fingerprint representation System

Filterbank based fingerprint identification system not only takes into account the local anomalies in the ridge structure (i.e. minutia points) but also the global pattern of ridges and furrows, inter-ridge distances, and overall patterns of ridge flow. This system is based on the fact that most textured images contain a limited range of spatial frequencies. Textured regions possessing different spatial frequency, orientation, or phase can be easily discriminated by decomposing the image into several spatial frequency and orientation channels. Computing the angle and coherence at each pixel in the image can derive symbolic description of a fingerprint image. Fingerprints can be represented/matched by using quantitative measures associated with the flow pattern (oriented texture) as features. Jain and Farrokhnia derived a global representation of

texture by decomposing the input image into different frequency and orientation components using a Gabor Filterbank [2]. This representation for oriented texture of fingerprints was inspired by Daugman's work on iris recognition and the success of the Gabor filterbank, as reported by Jain and Farrokhnia [3].

The four main steps in this filterbank based identification algorithm [2] are:

- **Reference point determination in fingerprint image,**
- **Tessellation of region around the reference point,**
- **Gabor Filtration (i.e. filtrating the tessellated area in six different directions using bank of Gabor Filters),**
- **Feature Vector Generation (i.e. Average Absolute Deviation (AAD))**

2.2.2.1 Reference Point Location

The point of maximum curvature of the concave ridges in the fingerprint image is taken as the reference point. The method locates the reference point more precisely than the algorithm proposed by Hong and Jain [1]. Center point location algorithm gracefully determines the point of most curvature by determining the normals of each fingerprint ridge, and then following them inward the center. The image is then cropped to a 223x223 pixels centered-around the reference point. The comprehensive description of steps for reference point determination algorithm is presented by Salil Prabhaker [2].

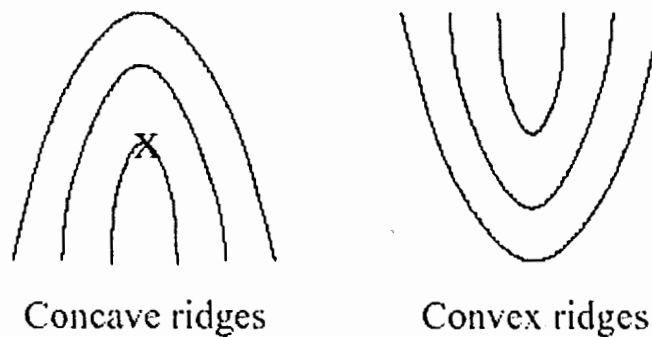


Figure 2.14 Most Concave Ridge Marked as Reference Point

2.2.2.2 Tessellation

The region of interest in the fingerprint image is the collection of all sectors around the center point where each sector is computed in terms of a particular radius and angle. This demarcation of the cropped area in sectors is called tessellation. The cropped fingerprint image is divided into 5 concentric bands centered on the reference point. Each band has a radius of 20 pixels, and a center hole radius of 12 pixels. Thus, the total radius of the sectorization is 111 pixels plus the central reference point. Each band is evenly divided into 12 sectors. The process of sectorization is carried out as for feature

extraction 6 equi-angular Gabor filters are used, which will align with the 12 wedges formed by the bands. In short each sector will capture information corresponding to each Gabor filter. The center band is ignored. All in all, there are a total of 60 sectors.



Figure 2.15 Reference point marked and area of interest Tessellated.

Another reason for sectorization is for normalization purposes. Each sector is individually normalized to a constant mean and variance to eliminate variations in darkness in the fingerprint pattern, due to scanning noise and pressure variations. And all the pixels outside of the sector map are considered to be one giant sector. This will yield in an image that is more uniform. Equation 2.1 is used for normalizing pixel intensities in each sector of the cropped fingerprint image.

$$N_i(x, y) = \begin{cases} M_0 + \sqrt{\frac{V_0 \times (I(x, y) - M_i^2)}{V_i}}, & \text{if } I(x, y) > M_i \\ M_0 - \sqrt{\frac{V_0 \times (I(x, y) - M_i^2)}{V_i}}, & \text{otherwise} \end{cases} \quad (2.1)$$

Where M_0 is constant mean and V_0 is constant variance, both having values 100, i is the sector number, M_i is the mean of sector, and V_i is the variance of each sector.

2.2.2.3 Gabor Filtration

The normalized image is then passed through a bank of Gabor filters. Each filter is performed by producing a 23x23 filter image for 6 angles ($0, \pi/6, \pi/3, \pi/2, 2\pi/3$ and $5\pi/6$), and convolved with fingerprint image. Spatial domain convolving is rather slow, so multiplication in the frequency domain is to be preferred. However this involves more memory to store real and imaginary coefficients.

The purpose of applying Gabor filters is to remove noise while preserving ridge valley structures and to provide information contained in a particular direction in image. A fingerprint convolved with a 0-oriented (0 degree) filter accentuates those ridges that are parallel to that angle (0 degree) and smoothes out the ridges of all other directions. Filters tuned to other angles work in similar manner. These six directional-sensitive filters capture most of the global ridge directionality information as well as the local ridge characteristics present in fingerprint. Equation 2.2 is the definition of Traditional Gabor filter.

$$G(x, y, f, \theta) = \exp\left\{-\frac{1}{2}\left[\frac{x^2}{\delta_x^2} + \frac{y^2}{\delta_y^2}\right]\right\} \cos(2\pi f x'), \quad (2.2)$$

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

$$y' = x \cos \theta - y \sin \theta$$

The parameters, δ_x and δ_y , are Gaussian constants whose values are empirically determined and set to 4.0. Very small value will make the filter ineffective in removing noise while too large a value will destroy ridge and furrow details. The filter frequency f is set to average inter ridge frequency ($1/k$), where k is the average inter-ridge distance which is determined to be on average 10 pixels in 500 dpi image.

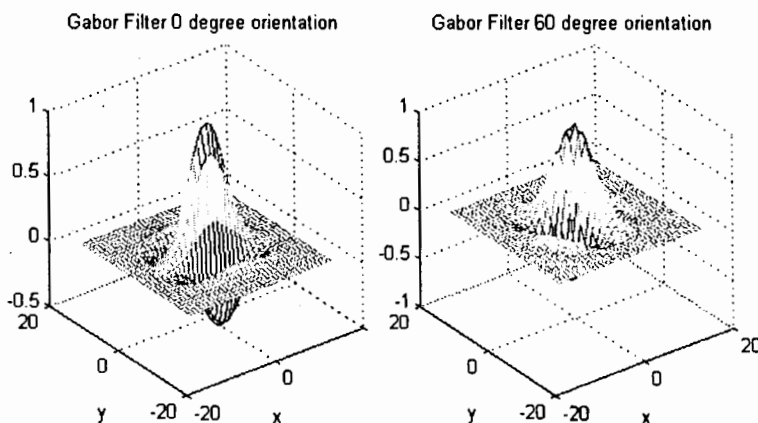


Figure 2.16 0 & 60 Degree Oriented Gabor Filter

2.2.2.4 Feature Vector

After getting the 6 filtered images, the variance of the pixel values is calculated in each sector. This will tell the concentration of fingerprint ridges going in each direction in that part of the fingerprint. A higher variance in a sector means that the ridges in that image are going in the same direction, as is the Gabor filter. A low variance indicates that the ridges are not, so the filtering smooths them out. The resulting 360-variance values (6 × 60) are the feature vector of the fingerprint scan. Equation 2.3 is used for variance calculation.

$$V_{i\theta} = \sqrt{\sum_{K_i} (F_{i\theta}(x,y) - P_{i\theta})^2} \quad (2.3)$$

Where $F_{i\theta}$ represents the individual pixel value in the i th sector. $P_{i\theta}$ is the mean of the pixel values. K_i is the number of pixels in the i th sector.

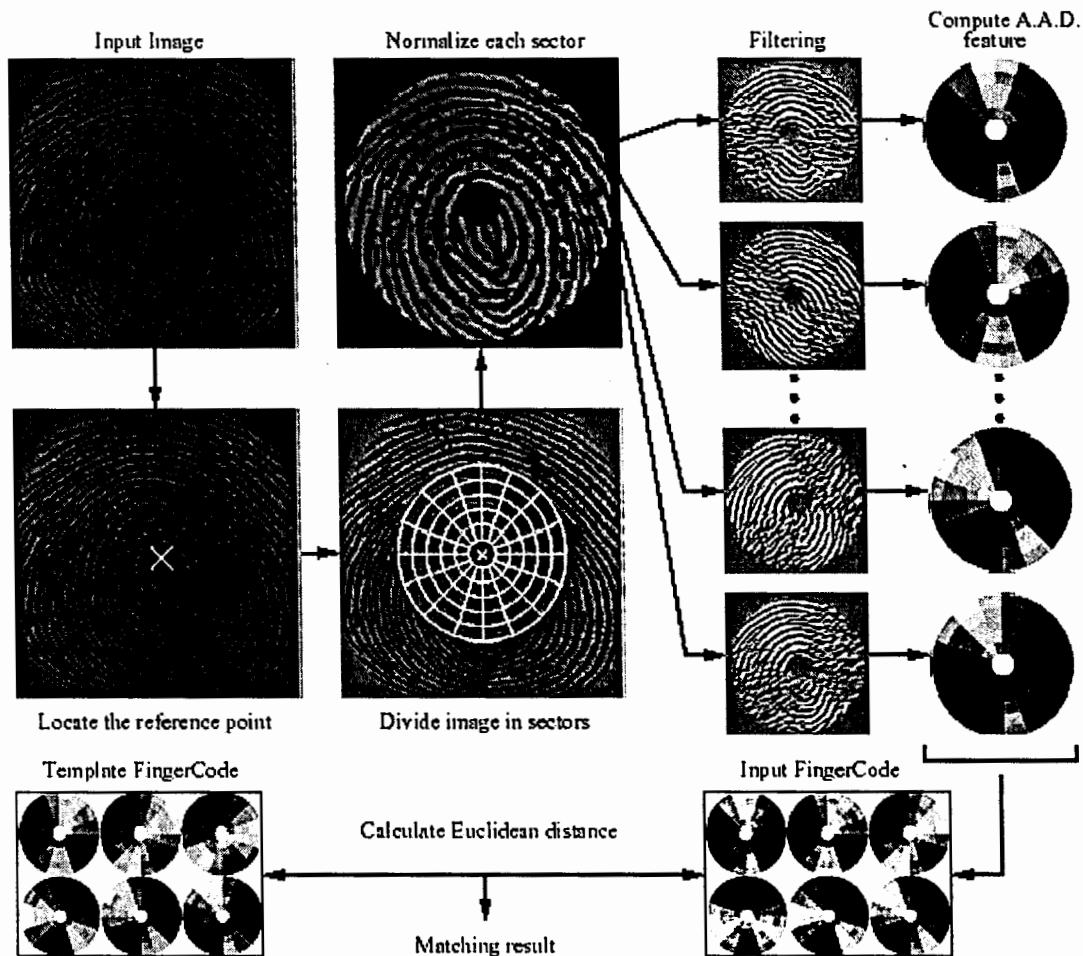


Figure 2.16 Flowchart of Filterbank based Feature Extraction Algorithm

T-998

Chapter 3

Conceptual Design and Development Environment

3. Conceptual Design And Development Environment

This chapter discusses the software architecture, overall conceptual design of the fingerprint identification system and also provides the reason for choosing a specific development environment and the design methodologies used during implementation.

3.1 System Architecture

The architecture of a fingerprint-based automatic identity authentication system is shown in the Figure 3.1. It consists of the following four components:

- *User Interface*
- *System Database*
- *Enrollment Module and*
- *Identification Module*

3.1.1 User Interface

This module provides mechanisms for a user to indicate his/her identity and input his/her fingerprints into the system.

3.1.2 System Database

The system database consists of a collection of records, each of which corresponds to an authorized person that has access to the system. Each record contains the following fields for authentication purpose:

- User name i.e. the person enrolled with the system
- Feature template of the fingerprint and
- Other information (e.g., specific user privileges).

3.1.3 Enrollment Module

The task of enrollment module is to enroll persons and their fingerprints into the system database. When the fingerprint images and the user name of a person to be enrolled are fed to the enrollment module, the feature extraction algorithm along with some other system required algorithms are performed and subsequently the feature vector picture of fingerprint is stored for that particular person.

3.1.4 Identification Module

The task of Identification module is to authenticate the current user who intends to access the system. The person to be identified provides his/her fingerprint image. Feature vectors are extracted from the provided fingerprint image and fed to a matching algorithm, which matches it against the stored records in the system database to establish the identity of the current user.

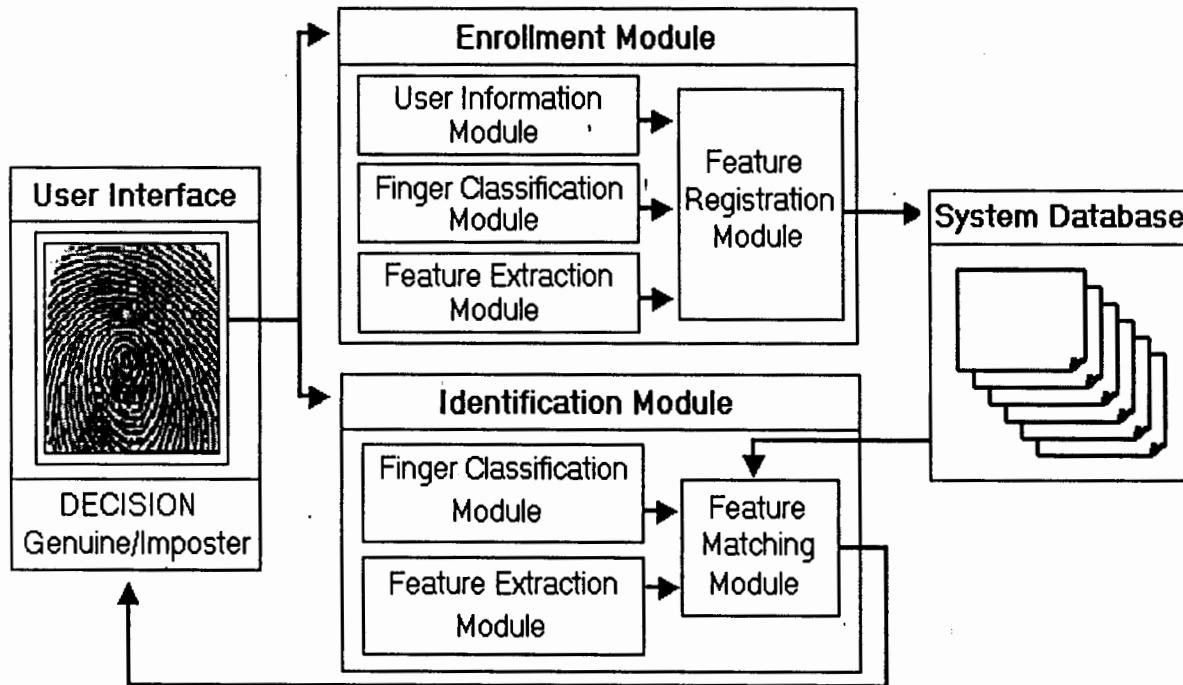


Figure 3.1: Fingerprint Identification System Architecture

3.2 System Components

The holistic view of the project can be broken down into three main components or subdivisions i.e. inputs, process and outputs. Each one of these subdivisions comprises of a number of modules used for processing and displaying the information flow within the program.

The overall conceptual design for the system is depicted by figure 3.2:

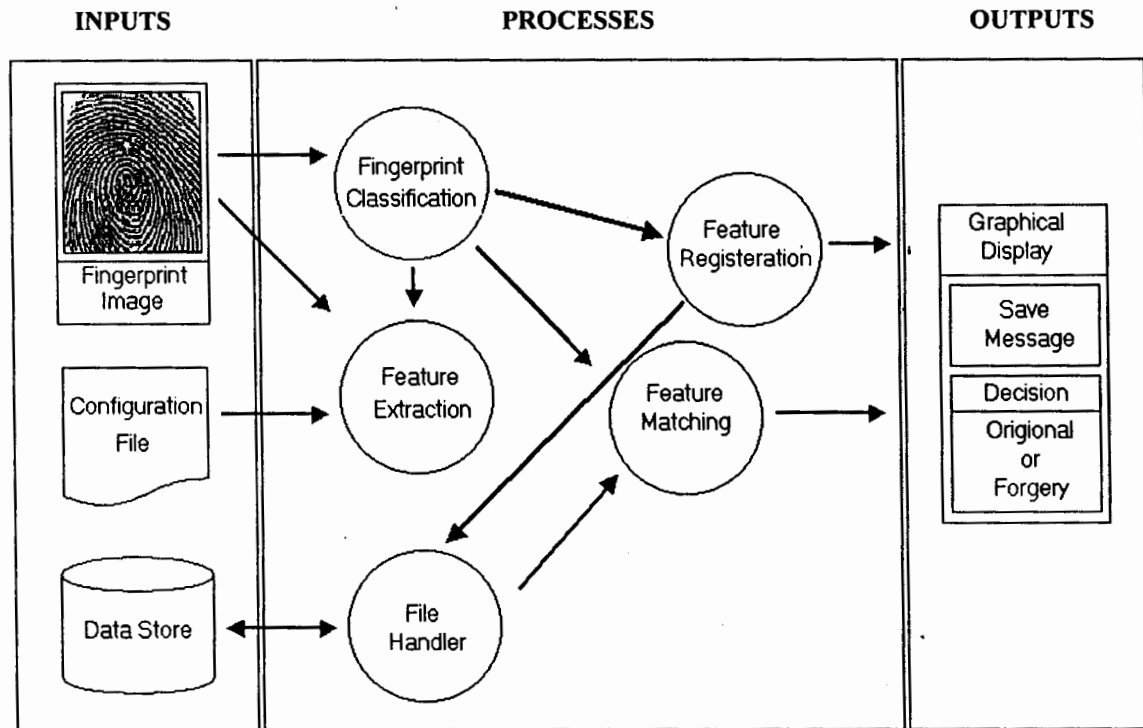


Figure 3.2 Overall Conceptual Design

Each module in the design has different purpose. These are described in more detail in the following sections:

3.2.1 Inputs

3.2.1.1 *Fingerprint Image*

A fingerprint image of the current user will be provided as the main input to the AFIS. Due to the unavailability of the sensor, a database of the fingerprint images is made available for the real time testing of the system. The images are in bitmap format and for the time being the system test will be restricted to this type only.

3.2.1.2 *Configuration File*

The configuration file will store parameters that will be used in the feature extraction and matching processes. By separating out the parameters used in the program, changes can be easily made to any of the assumptions made in the program.

3.2.1.3 *Data Store*

Data store is used to record all of the captured data about each subject in simple CFile format. Two different files are used for storing individual data. All Users personal

information i.e. his/her name, designation, privilege levels are stored in one file and the corresponding fingerprint data is stored in a separate file. Both of these files are connected together via unique primary key.

3.2.2 Processes

3.2.2.1 Feature Extraction

The most important of all the processes is feature extraction process. No matter what operational mode (enrollment / identification) the system is in, this process is bound to execution. Feature vectors or fingerprint representation values are extracted from the input fingerprint image, which are saved in the database or matched with the database records depending on the operational mode i.e. enrollment or identification mode respectively.

3.2.2.2 Classification

This module will perform the task of assigning one of the predefined classes to the input fingerprint image. In the absence of this module all the fingerprint images will be stored in a single bin in the database, the shortcoming would then be the comparison of the input image with all those enrolled with the system. But if we place the finger images into different classes corresponding to their global characteristics then the matching time will reduce significantly as the system has to search for the template from the class which matches with the current fingerprint image's class.

3.2.2.3 Feature Registration

Information from the feature extraction module and fingerprint classification module are collected in this module, which are finally written to the database. This module also manages information irrelevant to fingerprint image such as user's name, designation, privilege level, etc.

3.2.2.4 Feature Matching

This module performs the task of identification of the current user. The features extracted from the fingerprint are then compared with those stored in the database. The Euclidean distance of the corresponding features is determined and based on this module's outcome the system then forwards the final wording for the user identification.

3.2.2.5 File Handler

The file handler may be identified as the interface between the system and its data manipulation activities. It provides the data trafficking between the database and different process modules.

3. 2.3 Output

3. 2.3.1 Graphical Display

This is the application's main screen, which contains the user specific information. Image loading, operation performed on it and system controls are salient features of this visual display screen. Also informs the user of the various processes that are carried out as system operates in any of its working modes.

3.3 Development Environment

This project is developed using Microsoft Visual C++ 6.0 and is run and tested on Pentium III 550 MHz, with 128MB RAM. C++ is chosen as implementation language, because of the speed of execution of the code. It is possible to develop a program using Java, but this is not employed for implementation, as the language has slower execution speed. This is because Java is compiled into an intermediate form, Java Bytes rather than machine code, which have to be interpreted at program run time. Also using Visual C++ environment offers the possibility for the fingerprint identification application to be integrated into other programs developed by the Visual Information Processing group at Imperial. Visual C++ also offers excellent debugging tools, which aid quick testing, error locating, etc. Microsoft Foundation Classes (MFC), which can be used as part of Visual C++, offers the possibility to produce professional applications in the Windows environment through the use of the Windows API. The main disadvantage of using language is that it limits the application to the Windows environment, because Visual C++ applications based upon MFC will not run on the Linux operating system. Although this is a slight drawback it does not affect the project in any way.

3.4 Design Methodology

The following methodologies are used during implementation of the project:

3.4.1 Object Orientated Design

One of the reasons, Visual C++ is chosen for this project is because of its object-orientated capabilities. Within this program, common functionality should be grouped into classes, to allow reuse of code and a minimization of the amount of repeated code within the program.

3.4.2 Extensibility of Data Structures and Functionality

The design for overall architecture of the system and the individual classes should be as modular as possible in order to allow future extension of the project. Well-structured design will allow new functionality to be added to the overall program design with the minimal of difficulty.

Chapter 4

System Development and Implementation

4. SYSTEM DEVELOPMENT AND IMPLEMENTATION

As described in the previous section that almost all of the biometric system in general and fingerprint identification system in particular operates in two modes i.e. enrollment mode and identification mode. Each of these working modes has distinct processes within the system, which are carried out in predefined sequence based on the mode's operational requirements.

4.1 Steps for Enrollment Mode

The steps or various processes involved in the enrollment mode are as follows:

- *Fingerprint Image acquisition*
- *Feature Extraction*
- *Fingerprint Classification*
- *Feature Registration*

4.2 Steps For Identification Mode

The steps or various processes involved in the identification mode are as follows:

- *Finger Image acquisition*
- *Feature Extraction*
- *Fingerprint Classification*
- *Feature Matching*

As the two working modes of the system indicate that both have almost same processing steps except for the last step where one registers the feature generated (in enrollment mode) and other matches it with records in database (in identification mode).

4.3 Processes In Detail

4.3.1 Feature Extraction

Aim

The main aim of the feature extraction process is to extract the represent-able information from the fingerprint image, which will then be further utilized in the processes to follow. The feature extraction process is made up of a number of sub-processes, which come up with discrete information, that collectively make the feature extraction process feasible.

Procedural Design

The procedural design transforms structural elements of the program architecture into a procedural description of software components. The procedural design of the feature extraction process is given by figure 4.1.

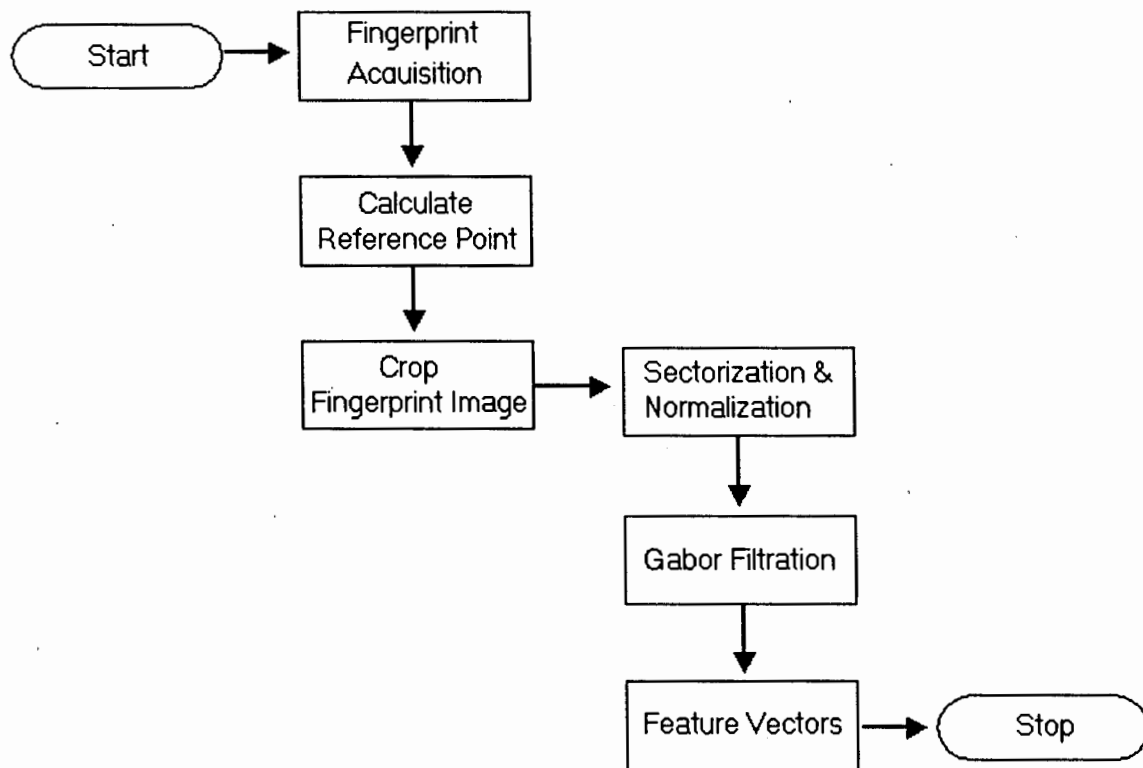


Figure 4.1 Procedural design for Feature Extraction Process

4.3.2 Fingerprint Classification

Aim

Automatic identity recognition based on fingerprints requires that the input fingerprint be matched with a large number of fingerprints stored in a database. Fingerprint classification provides an important indexing mechanism in a fingerprint database. An accurate and consistent classification can greatly reduce fingerprint-matching time for a large database. Fingerprint classification is a technique used to assign a fingerprint into one of the several pre-specified types as defined earlier in the second chapter of the report. An input fingerprint is first matched to one of the pre-specified types and then it is compared to a subset of the database corresponding to that fingerprint type. To increase the search efficiency, the fingerprint classification algorithm can classify a fingerprint into more than one class, as is the case in our project. The classification process is based on our filterbank fingerprint representation scheme used for feature extraction process. Few of the parameters in the feature extraction process are altered which then easily fit into our fingerprint classification algorithm. Information thus obtained is passed onto the K Nearest Neighbor Classifier.

Procedural Design

The procedural design of the Classification process is illustrated in figure 4.2.

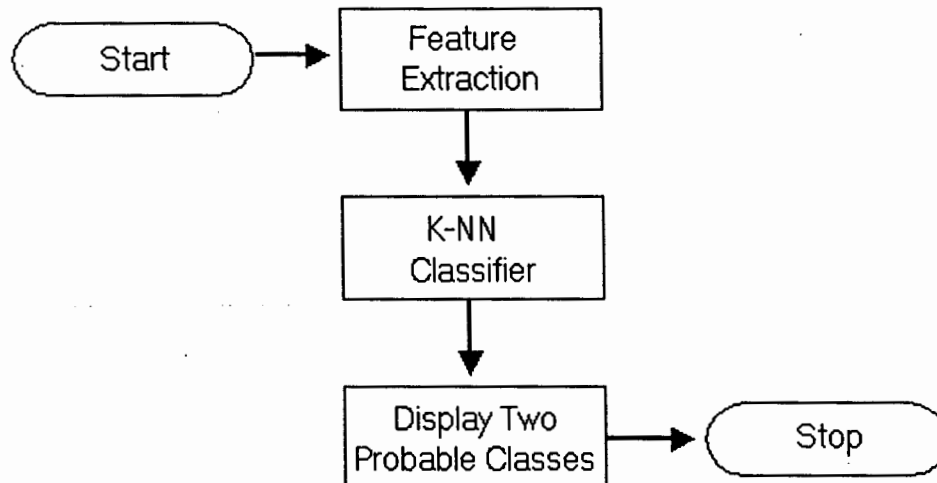


Figure 4.2 Procedural design for Classification Process

4.3.2.1 K Nearest Neighbor Classifier

Automatic classification of fingerprints is a difficult problem because of the small interclass variability and large intraclass variability among the five classes (Whorl, Left Loop, right Loop, Arch, Tented Arch) under consideration. In order to simplify the

classification task, we decompose the five-class problem into a set of 10 two-class problems (Possible combinations of all five classes in the set of two). We use a K-nearest neighbor classifier to find the two most probable classes for a given input pattern. The top two categories can be retrieved from the K-NN classifier corresponding to the classes which have the highest and the second highest count among the K nearest neighbors, i.e., the first recall and the second recall. K is set equal to 10. So the query in classifier comes with up with 10 classes from the training set, whose Euclid distance is the minimum. Of these 10 final two classes are picked corresponding to the two lowest Euclid values.

Procedural Design

Figure 4.3 below illustrates the procedural design of K-Nearest Neighbor Classifier.

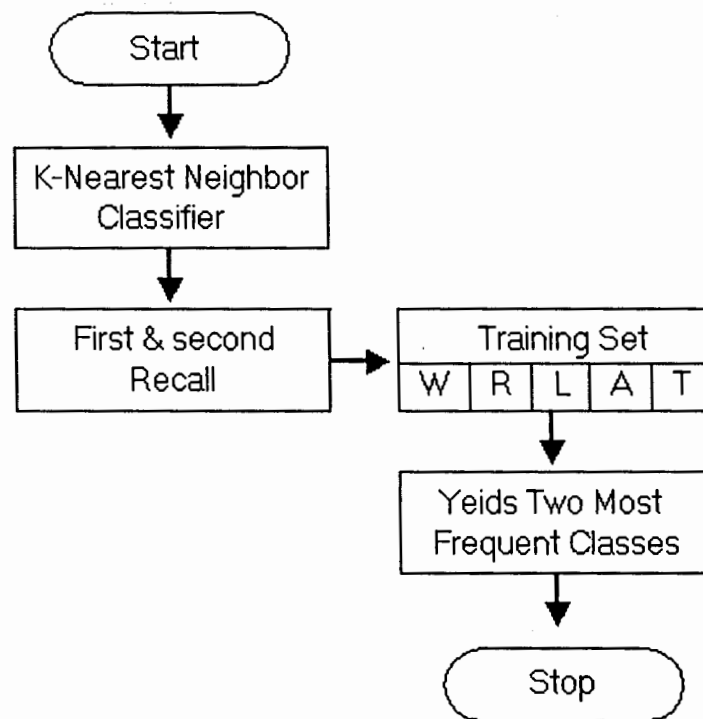


Figure 4.3 Procedural design for KNN Classifier Process

4.3.3 Feature Registration

Aim

As the user when comes for identification his fingerprint information has to be matched with a certain template. So all the users of the system first have to get them registered with the system. So the fingerprint image is first collected and its feature are extracted and then that information along with the user's personal information is saved in the system's database.

Procedural Design

The procedural design of feature registration is shown in the figure 4.4.

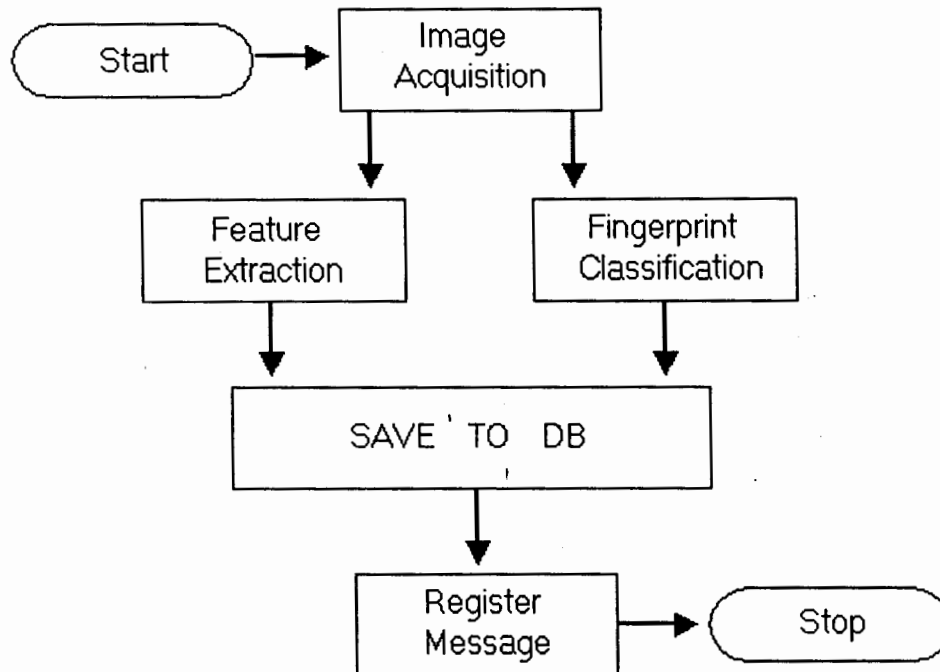


Figure 4.4 Procedural design for Feature Registration Process

4.3.4 Feature Matching

Aim

The final and the most important process in the system's identification mode is the feature matching process. It is here that the discrimination between the genuine and imposter is carried out. The features of the input image are compared with those stored templates in the database and the genuine user is identified if at all he/ she is registered with the system.

Procedural Design

The procedural design of the feature matching process is depicted in the following figure 4.5.

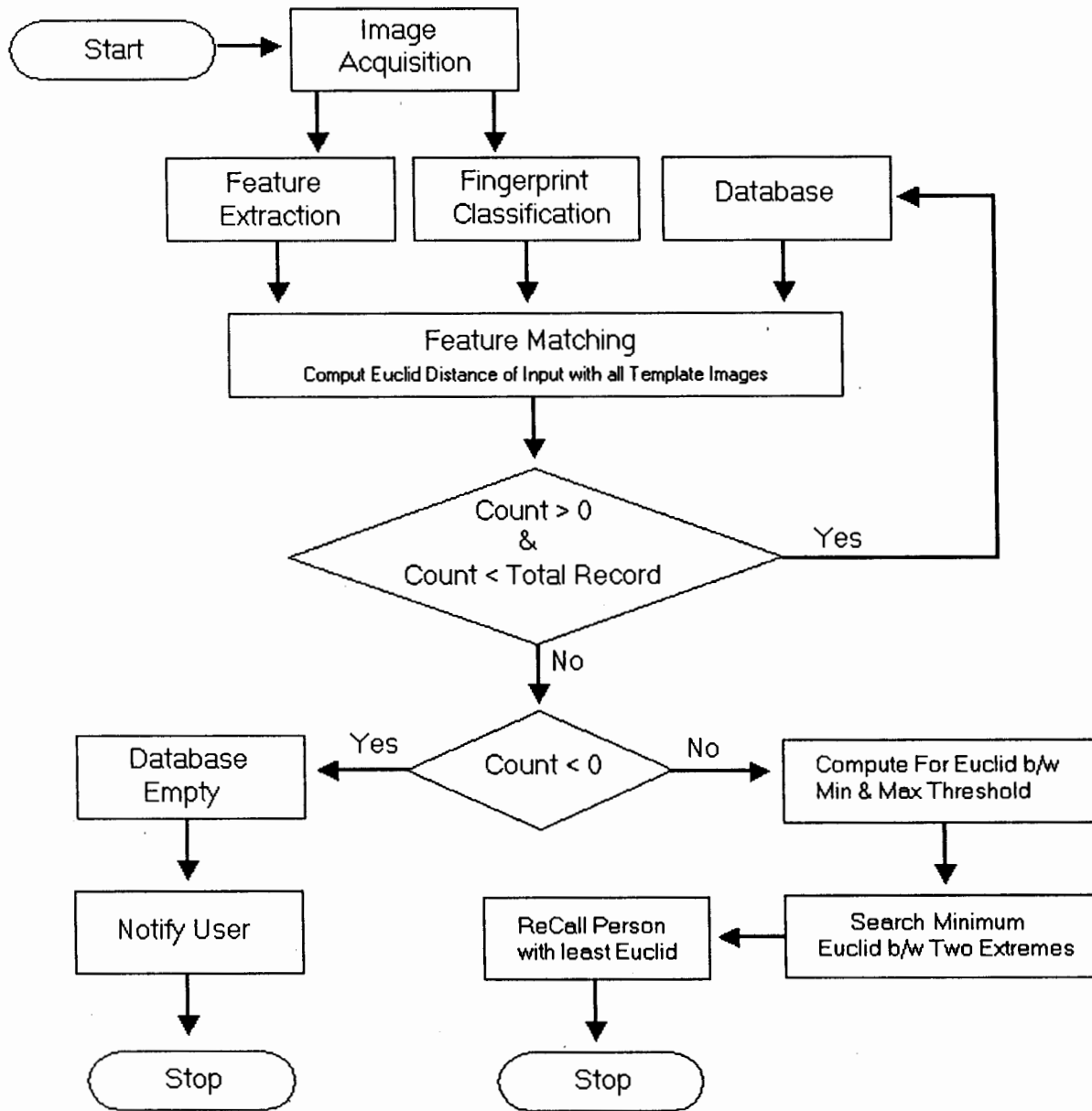


Figure 4.5 Procedural design for Feature Matching Process

4.4 Algorithm and Pseudo Code

As the project's main concern is fingerprint signature extraction and information thus generated is stored in the system for future identification purposes, so there need to be a mechanism through which features could be extracted from the fingerprint impression. The procedural design for feature extraction discussed earlier in section 4.3.1 clearly points out the processes

involved. These processes are representative of algorithms used for feature extraction.

4.4.1 Algorithm for Reference Point Calculation

Begin

Load fingerprint image and retrieve its dimension and pixel values;
Apply lowpass Wiener filter to remove noise from image;
Calculate gradient G_x & G_y at each pixel using Sobel operator;
Divide fingerprint image into non-overlapping blocks of size 10;
Compute slope perpendicular to local orientation of each block;

$$\Theta = \frac{1}{2} \tan^{-1} \left(\frac{\sum_{i=1}^{10} \sum_{j=1}^{10} 2G_x(i,j)G_y(i,j)}{\sum_{i=1}^{10} \sum_{j=1}^{10} (G_x^2(i,j) - G_y^2(i,j))} \right) + \frac{\pi}{2}$$

Binarize blocks with slope ranging from 0 and $\pi/2$;
Trace blocks with values ranging from 0 and $\pi/2$ to blocks with values not ranging from 0 and $\pi/2$;
Check for the block with highest number of markings;
Calculate the center point of this block to mark as the reference point;

End

4.4.2 Pseudo Code for Reference Point Calculation

```
//Load Fingerprint image and gray scale pixel information
ImageInfo = load(fingerprint );

//Apply lowpass Wiener filter of neighborhood size 5 by 5 to remove noise
WienerFilter(ImageInfo, 5,5);

//Apply Sobel filter to to calculate gradient  $G_x$  &  $G_y$ 
SobelOperator(ImageInfo,  $G_x$ ,  $G_y$ );

//Evaluate equation  $(2 \times G_x \times G_y)$  using window size/blocks of size 10 x 10 pixels
Calculate $2G_xG_y$  ( $2G_xG_y$ ,  $G_x$ ,  $G_y$ , 10);

//Evaluate equation  $(G_x^2 - G_y^2)$  using window size/blocks of size 10 x 10 pixels
Calculate $G_x^2G_y^2$  ( $G_x^2G_y^2$ ,  $G_x$ ,  $G_y$ , 10);

//Calculate orientation of each block
for count1 = 0 : ImageWidth/10
```

```

        for count2 = 0 : ImageLength/10
            .
            .
            .
            OrientImage[count1, count2 ] = angle;
            count2 = count2 + 1;
        end
        count1 = count1 + 1;
    end

//Binarize blocks with slope values between 0 and 90
    Binarize(OrientImage);

//Get x and y index of blocks with slope values ranging from 0 and 90
    GetPosition(Binarize);

//Mark the blocks and trace a path from blocks with slope values ranging from 0 and 90
//to block not ranging from 0 and 90
    for count1 = 0 : ImageWidth/10
        for count2 = 0 : ImageLength/10
            .
            .
            .
            MarkedImage[count1, count2 ] = value;
            count2 = count2 + 1;
        end
        count1 = count1 + 1;
    end

// Check the block with maximum markings and mark its center as the reference point
    maximum(MarkedImage,xPos, yPos);
    Referencepoint.x = xPos,yPos;
    Referencepoint.y = yPos;

    Return Referencepoint;

```

4.4.3 Algorithm for Tessellation

Begin

S_i = Collection of all the sectors
 i = Sector number for $i = 1 \dots 60$
 $B = 5$; Number of concentric bands
 $k = 10$; Number of sectors in each band
 $b = 20$; Width of each band
 θ = angle of tessellation
 (x_c, y_c) = Reference point

$$S_i = \{(x, y) | b(T_i + 1) \leq r < b(T_i + 2),$$

$$\theta_i \leq \theta < \theta_{i+1}, 1 \leq x \leq N, 1 \leq y \leq M\},$$

$$T_i = i \text{ div } k,$$

$$\theta_i = (i \text{ mod } k) \times (2\pi/k),$$

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

$$\theta = \tan^{-1}((y - y_c)/(x - x_c)),$$

End

4.4.4 Pseudo Code for Tessellation

```
//Get image width and height which are both equal
M = Image width = Image height

//Get current pixel position whose sector number is to be determined
Index = (xPos, yPos)

//Set maximum index value equal to image width and image height
MaxIndex = ( Image width, Image height )

//Check whether the pixel is within the cropped image
if( Index < (0,0) || Index > MaxIndex )
    return error;
end

//Calculate radius
rad = (xPos)2 + (yPos)2

//Check for inner or outer band
if( rad < 12 ) || ( rad > (111)2 )
    return message; //Out of region of interest
end

//Calculate angle for the current pixel
theta = calculateangle(xPos, yPos)
r = sqrt(rad)
ring = ( r - 12 ) / 20
arc = theta / (Pi / 6)
```



```
//Get the sector number for this particular pixel
sector = ring X 12 + arc
```

4.4.5 Algorithm for Normalization

Begin

```
Mo = 100; Empirically determined constant mean value
Vo = 100; Empirically determined constant variance value
I(x,y) = Pixel gray scale value at position x, y
i = Sector number for i = 1 . . . . 60
Mi = Mean pixel value for ith sector
Vi = Mean variance value for ith sector
Ni = Normalized pixel value
```

```
For i = 1 : ImageSize
```

$$N_i(x, y) = \begin{cases} M_0 + \sqrt{\frac{V_0 \times (I(x,y) - M_i)^2}{V_i}}, & \text{if } I(x, y) > M_i \\ M_0 - \sqrt{\frac{V_0 \times (I(x,y) - M_i)^2}{V_i}}, & \text{otherwise,} \end{cases}$$

End

4.4.6 Pseudo Code for Normalization

```
//Assign sector number to each pixel
ExecuteTessellateAlgo(Image Size)

//Calculate mean of each sector
for count = 1 : Total Sectors
    mean[count] = mean[count] / num[sector pixels];
    count = count + 1;
end

//Calculate variance of each sector
for count = 1 : Image Size
    variance[sector] += (pixelvalue[count] - mean[sector])2;
    count = count + 1;
end

for count = 1 : total Sectors
    variance[count] = variance [count] / num[sector pixels]
    count = count + 1;
end
```

```
//Perform normalization
for count = 1 : Image Size
    if(fabs[variance[sector] > 1.0)
        pixelvalue = pixelvalue[count] - mean[sector];
        if(pixelvalue < 0)
            pixelvalue[count] = M_O -sqrt(V_O / variance[sector])
                x (pixelvalue)2;
        else
            pixelvalue[count] = M_O +sqrt(V_O / variance[sector])
                x (pixelvalue)2;
        end
    else
        pixelvalue = M_O;
    end
    i = i +1;
end

//Return normalized image
return image;
```

Chapter 5

Utility Classes

5. Utility classes

The purpose of this section is to explain the functionality of a number of ubiquitous classes, which have been implemented in the project. Primarily their names suggest their representation of modules discussed in previous chapter, while others may act as utility classes that are used throughout the entire code either providing data handling, processing or providing utility functions to format the data in the desired manner. An overview of most important classes is given but before that a brief look at the class diagram (Figure 5.1), which narrates the overall class hierarchy and relationships.

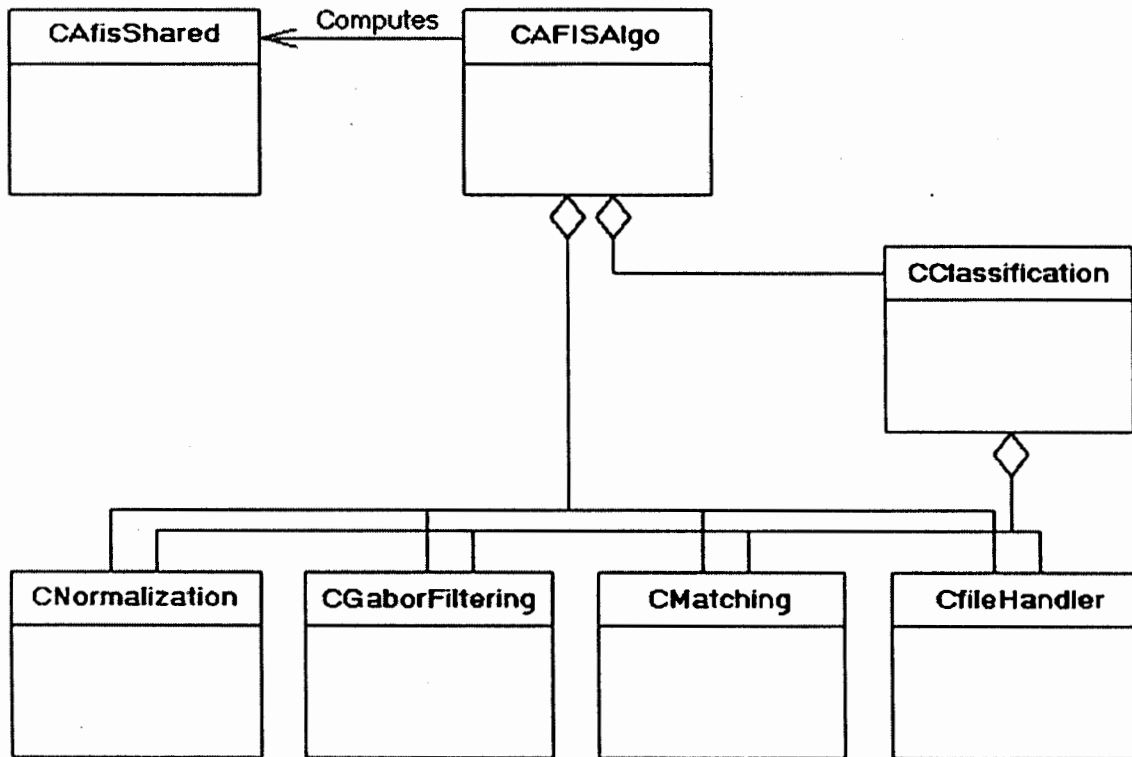


Figure 5.1 Class Diagram of AFIS Project

NOTE: Details of Member Function & Member Variables are given in individual class elaborated text

5.2 CAFISAlgo Class

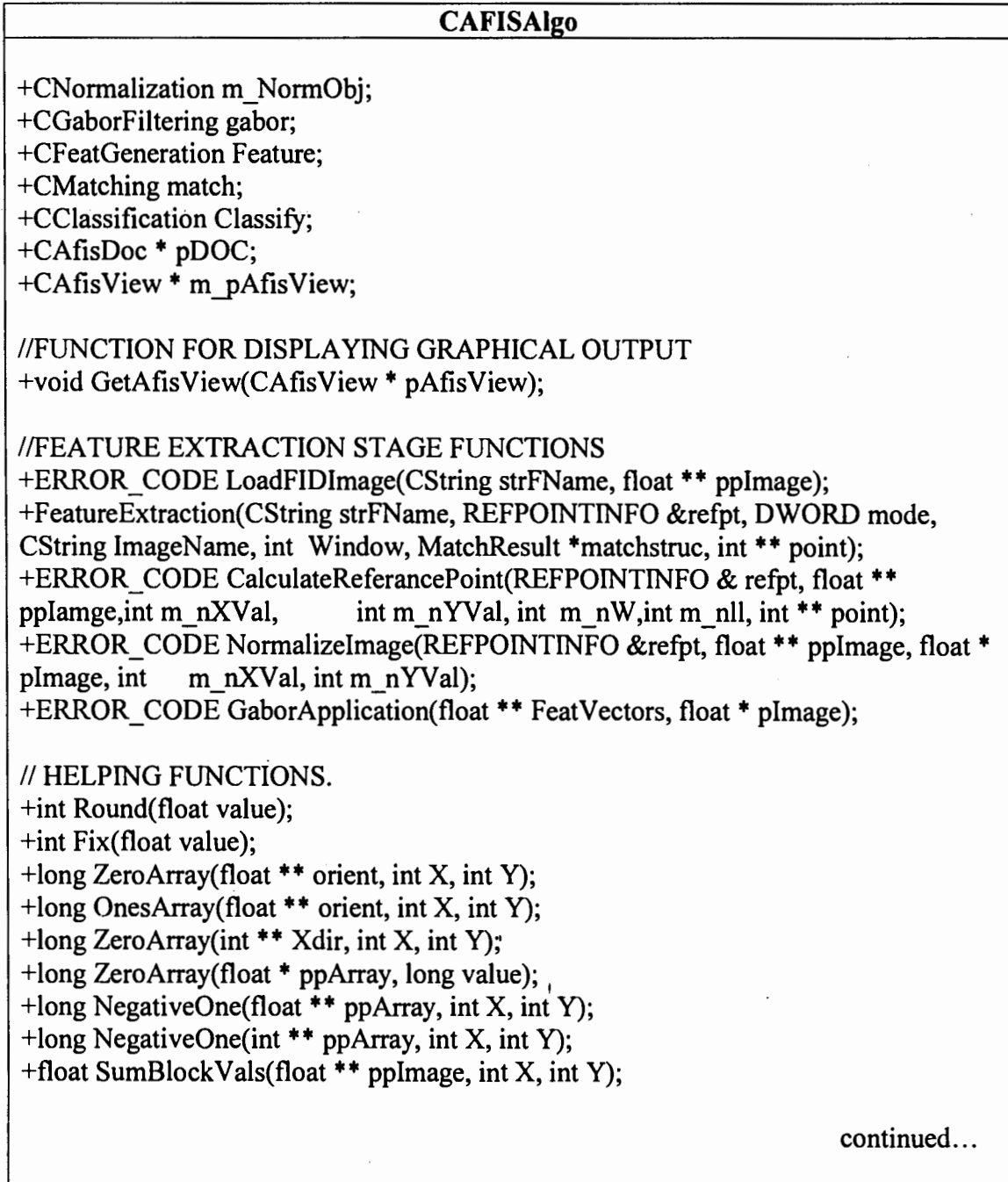
Responsibility

The most important, in fact the main class in our project is CAFISAlgo class. This class represents the feature extraction process executed in any case either during enrollment or identification. Once the data has been captured it needs to be stored inside the program in such a way that it can be easily manipulated and used in calculations.

Fingerprint image information being the data in our case, is collected through MFC based document view architecture using simple file and bitmap reading MFC functions. The image data is passed on to the CAFISAlgo class, which makes this information available to all the AFIS core classes.

Design

The following figure illustrates the structure of the CAFISAlgo class object and its relationship with other class objects in the project.



```

+long BinarizeImage(float ** pporient, int ** ppbinarize, int X, int Y);
+long Maxim(int ** binarize2, REFPOINTINFO & RefPtInfo, int X, int Y);
+ERROR_CODE Interchange(float ** ppImage1, float ** ppImage, int X, int Y);
+long FindNonZeroElementsPos(int ** ppbinarize, INTVECTOR & vecbi,
INTVECTOR & vecbj, int X, int Y);

//IMAGE MANIPULATION FUNCTIONS
+long FlipImage();
+long Smooth_Filter(float ** ppImage, int X, int Y, int filter_x, int filter_y);

//HELPING FUNCTIONS FOR SMOOTH FILTER
+ERROR_CODE filter2(float ** localMean, float ** ppImage, int X, int Y, float **
nOnes,int filter_x, int filter_y);
+ERROR_CODE AssignValue(float * pImage, int X);
+ERROR_CODE RoundArray(float ** ppImage1, float ** ppImage2, int X, int Y);
+BOOL AnyZero(float** ppImage, int X, int Y);
+BOOL AnyZero(float * pImage, int X);
+BOOL CompareArray(float ** ppImage1, float ** ppImage2, int X, int Y);
+ERROR_CODE ConvRow(float * hRow, float ** ppImage, float ** Row, int X, int
Y);
+ERROR_CODE ConvCol(float * hCol, float ** ppImage, float ** Col, int X, int Y);
+ERROR_CODE Convolution(float ** ppImage, float ** Result, int X, int Y);
+ERROR_CODE divideByTenThousand(float ** ppImage, int X, int Y, int val);
+ERROR_CODE MultiplyByTenThousand(float ** ppImage, int X, int Y, int val);
+long filter2(float ** ppFilter, int nfilterX, int nfilterY, float ** ppImage, int X, int Y,
float ** ppRetImage);
+long MatrixSquare(float ** ppImage, float ** ppRetImage, int X, int Y);
+float Mean2(float ** ppImage, int X, int Y);
+long maxValuedItems(float ** ppRetImage1, int X, int Y, int nthreshold=0);
+long maxValuedItems(float ** ppRetImage1, float ** ppImage2, int X, int Y);
+long Addition(float ** ppImage1, float ** ppImage2, float ** ppRetImage, int X, int
Y);
+long Subtraction(float ** ppImage1, float ** ppImage2, float ** ppRetImage, int X,
int Y);
+long Subtraction(float ** ppImage, float ** ppRetImage, int X, int Y, float
subtractedValue);
+long Multiplication(float ** ppImage1, float ** ppImage2, float ** ppRetImage, int
X, int Y);
+long division(float ** ppImage1, float ** ppImage2, float ** ppRetImage, int X, int
Y);
+long division(float ** ppImage, float ** ppRetImage, int X, int Y, float divisor);

```

continued...

```

// GRADIENT FILTER APPLICATION.
+ERROR_CODE Gradient(float ** ppImage, float ** ppGx, float ** ppGy, int X, int
Y);
+long Permute(float ** ppImage, int n, int p);

// ORIENTATION FIELD CALCULATION.
+ERROR_CODE Calculate_2GxGy_and_GxGxGyGy(float ** ppOrientNum, float **
ppOrientDen, float ** ppGx, float ** ppGy, int X, int Y);

// MEMORY ALLOCATION FUNCTION.

ERROR_CODE MemAlloc(float *** ppp, int X, int Y);
void MemDealloc(float *** ppp, int X, int Y);
ERROR_CODE MemAlloc(int *** ppp, int X, int Y);
void MemDealloc(int *** ppp, int X, int Y);

```

Functionality

The CAFISAlgo is a representative class for the feature extraction process, so it performs most of the important tasks, which can be categorized as:

- ✓ Reference point calculation i.e. center point determination in fingerprint images
- ✓ Tessellation and Normalization of Cropped image
- ✓ Gabor Filtration of the Normalized Image
- ✓ Feature Vector generation from the Gabor filtered image
- ✓ Memory allocation & de-allocation functions
- ✓ Data conversion functions
- ✓ Helper functions

Memory allocation, de-allocation functions, data conversion functions and the helper functions make up most of the processing tasks needed at some point in the project and therefore their presence in this class help to reduce the amount of code which in other case would have been duplicated to a great extent.

5.3 CNormalization Class

Responsibility

This class deals with the tessellation and normalization of the selected or tessellated image. The reference point and the original image in 2D array form is passed on to this class which extracts a squared image of 223 x 223 pixels with respect to the

reference point. The image is then sectorized based on the settings mentioned in the configuration file i.e. AfisShared file.

Design

The following figure illustrates the CNormalization class object.

CNormalization
<pre>+ERROR_CODE ExtractIndexImagewrtRefPoint(float ** ppImage, int X, int Y, float * pImage, float ** ppImage2); +ERROR_CODE PerformNormalization(float * pIndexImage, int X, int Y); +int whichsect(long index);</pre>

Functionality

This class crops the desired image from the original image and then performs the normalization i.e. gray scaling on that cropped image to remove the noise which may result in tedious feature vector generation in later stages. The tasks are performed via the following member functions

NOTE: For function parameter illustration, please refer to the respective class diagram.

- ✓ **ExtractIndexImagewrtRefPoint():** As the name indicates that this function extracts the desired or 223 x 223 pixel wide image from the original input image with respect to the reference point.
- ✓ **PerformNormalization():** This function then performs the normalization task. The mathematical formula for normalization is given in the third chapter.
- ✓ **Whichsect():** The normalization is carried out in sectors separately rather than the whole image collectively. As the sectors are not square and an angle is involved in sectorization so mathematical calculations are carried out to determine which pixel belongs to which sector and correspondingly the whole process is completed in various iterations.

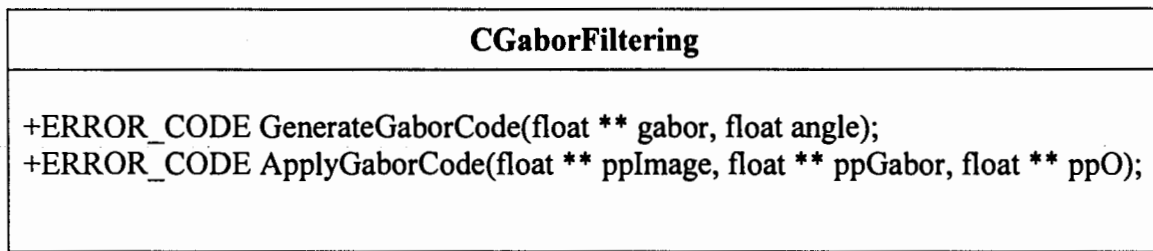
5.4 CGaborFiltering Class

Responsibility

This class, as the name suggests performs the Gabor filter application on the normalized image. Gabor filter properties have already been discussed in section 2.2.

Design

The following figure illustrates CGaborFiltering class object.



Functionality

This class simply generates the Gabor filter depending upon the angle value and then the Gabor and image convolution takes place.

NOTE: For function parameters illustration, please refer to the respective class diagram.

- ✓ **GenerateGaborCode():** This function generates the specific angle oriented gabor filter depending upon the angle parameter. The gabor generation equation 2.2 is already defined in section 2.2.2.3. This function executes for 6 times as 6 different angles are used for Gabor generation (0, 30, 60, 90, 120, 150);
- ✓ **ApplyGaborCode():** This function simply performs the convolution process involving both the original image and gabor filter. The original image is left intact. This function also executes for 6 times, which therefore add to the processing time and overall system's deficiency.

5.5 CClassification Class

Responsibility

As the name suggests that it has to do with the classification of fingerprint images in this fingerprint identification system. This process is questioned to system's performance when not a large number of people are registered with the system. The reason is that this involves all those sub-processes (tessellation & normalization, 4-angle gabor filtration, and matching processes involving training set database) of feature extraction stage. It involves almost all classes for its execution.

Design

The following figure illustrates the CClassification class object. .

CClassification
<pre> //ASSOCIATED CLASS'S INSTANCES +CGaborFiltering gabor; +CMatching match; +ERROR_CODE ExtractIndexImagewrtRefPoint(float ** ppImage, int X, int Y, float * pImage, float ** ppImage2); //+ERROR_CODE ExtractIndexImagewrtRefPoint(float ** ppImage, int X1, int X2, int Y1, int Y2, float * pIndexImage); +ERROR_CODE PerformNormalization(float * pIndexImage, int X, int Y); +int whichsect1(long index); +int whichsect2(long index); +ERROR_CODE GaborApplication(float ** FeatureVectors, float * pImage); +ERROR_CODE GenerateFeatureVectors(float * pIndexImage, int X, int Y, float ** FeatVectors, int count); +ERROR_CODE QuantizeVectors(float ** ppImage , int X, int Y); +ERROR_CODE FeaturesMatching(FVINFOVEC &DBInfovec, float ** FeatVectors, CLASSINFO * ClassInfoArray, int X); +double EuclidDistance(float ** featvector, FVInfo * Info); +ERROR_CODE WhichClass(REFPOINTINFO &refpt, float ** ppImage, float * pImage, CString * ClassOne, CString * ClassTwo); +ERROR_CODE FindKNN(CString * Class1, CString * Class2, CLASSINFO * ClassInfoArray, int X); //HELPER FUNCTIONS +long ZeroArray(float * ppArray, long value); +long ZeroArray(float ** orient, int X, int Y); +int Round(float value); </pre>

Functionality

As stated earlier, this class involves almost all the processes to be repeated again with slight deviation in parameters with respect to other classes, so one really feels the boredom in writing all that stuff again. The classification process detail has already been provided in earlier chapters.

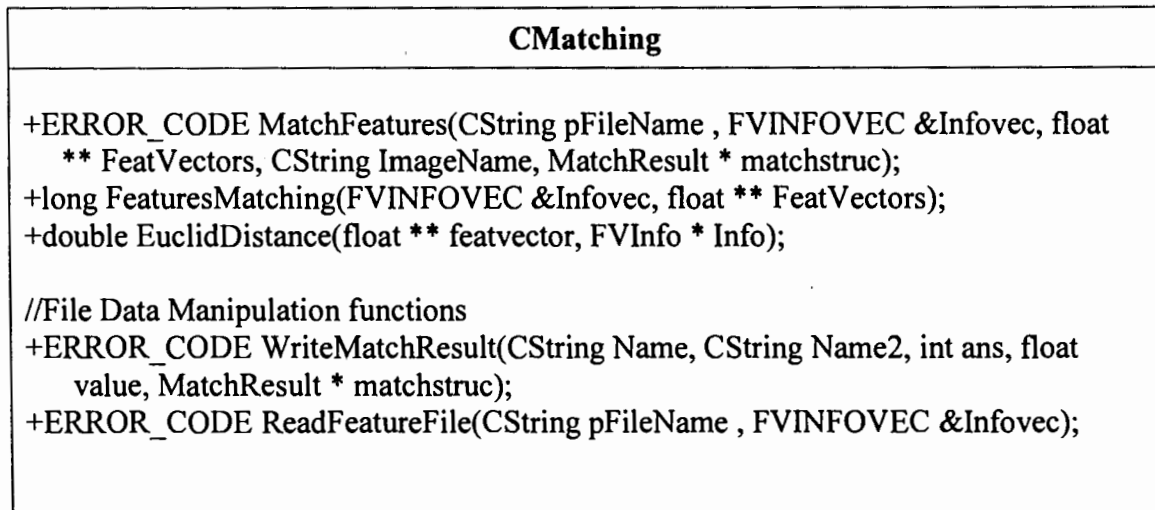
5.6 CMatching Class

Aim

The class name being self-explanatory, informs the reader of its purpose. The currently generated feature vectors of the input fingerprint image are matched with those stored in the database.

Design

The following figure illustrates the CMatching class object.



Functionality

The details of the functions which carry out the functionality of matching class are as follows:

NOTE: For function parameters illustration, please refer to the respective class diagram.

- ✓ **MatchFeatures():** This is the main function of this class which is invoked in CAFISAlgo class via this class's object declared in that class. Currently generated feature vectors, database file name (corresponding class's DB file if classification is enabled) and a structure for storing results are passed from the preceding class are passed on to this function which then carry out the required procedure.
- ✓ **FeaturesMatching():** This function actually match currently generated feature vectors with those stored in the database.
- ✓ **EuclidDistance():** This function performs the mathematical calculation for determining the Euclidean distance between the two sets of feature vectors. The equation for this process is given in the third chapter.

5.7 CAfisShared

Responsibility

This is the configuration class whose data is being shared by all the classes implemented in the project. All the algorithm's performance-tuning parameters are declared in this class. The purpose is that instead of changing the parameter's value in each module, this could be done at a global level without disturbing systems overall integrity.

Design

Figure 7.3 illustrates the CAfisShared class.

CAfisShared
<pre> /**This is the configuration files that specify all the parameters in the algorithms**\ // NORMALIZATION CONSTANTS #define INDEX_IMAGE_SIZE 49729 // keep in mind declare index_image of size 49729+1 #define INDEX_IMAGE_WIDTH_LENGTH 223 #define INDEX_GABOR_IMAGE 256 #define INDEX_IMAGE_RADIAN_MAX_VALUE 12544 #define BAND_SIZE 20 #define NO_OF_BANDS 5 #define SECTORS_PER_BAND 12 #define INNER_BAND_RADIUS 12 #define GABOR_ANGLES 6 #define NO_OF_SECTORS (NO_OF_BANDS * SECTORS_PER_BAND) #define V_0 100.0 #define M_0 100.0 // NORMALIZATION CONSTANTS FOR CLASSIFICATION #define CLASSIFICATION_SECTORS 8 #define CLASSIFICATION_ANGLES 4 #define NO_OF_CLASSIFICATION_SECTORS (NO_OF_BANDS * CLASSIFICATION_SECTORS) #define PI 3.14159265358979323846264338327 // GABOR GENERATOR CONSTANTS #define INTER_RIDGE_DISTANCE 10.0 // average inter-ridge pixel. </pre>
continued...

```

#define GABOR_SIZE 33
#define VAR 32.0
#define TOTAL_GABORS 6

//MATCHING CONSTANTS
#define MAXI_THRESHOLD 330000.0
#define DIFF_THRESHOLD 50000.0

// APPLICATION MODE CONSTANTS
enum AFIS_MODE {ENROLLMENT=0, RECOGNITION, GENERALIZATION};
static AFIS_MODE g_application_mode = ENROLLMENT;

//STL based vector class for data storage
#include <vector>
typedef vector<int> INTVECTOR;

// ERROR FUNCTIONS.
// It contain error string and error number.
typedef struct ERROR_DETAIL
{
    long nErr;
    char strError[256];
    ERROR_DETAIL()
    {
        nErr = 0; // success.
        strcpy(strError, ""); // error string.
    }
}

ERROR_CODE, *PERROR_CODE;

//DATA STORAGE CONSTANTS
// FID FUNCTIONS.
//Structure contains the x & y position of an element in an array.
typedef struct FIDPOINT
{
    int x;
    int y;
}FidPoint, *pFidPoint;

//Structure contains the maximum value and it's position in the array
typedef struct REFPOINTINFO
{

```

continued...

```

    FidPoint ptRef;
    float maxValue;
} RefPointInfo, *pRefPointInfo;
typedef vector<RefPointInfo, allocator<RefPointInfo> > REFPOINTINFOVEC;

//Structure for getting feature vectors of all the enrolled persons
typedef struct FVINFO
{
    //Further information i.e. to be added in a particular record
    CString PersonName;
    CString ClassName;
    float Features[NO_OF_SECTORS * GABOR_ANGLES];

double euclid;// Euclidian distance
    double tryeuclid;//for trial matching
}FVInfo, *pFVInfo;
typedef vector<FVInfo> FVINFOVEC;

//Structure for getting matching results.
typedef struct MatchResult
{
    CString PersonName;
    CString Matchedwith;
    double euclid;// Euclidian distance
    CString Result;
    CString dept;
    UINT EMpId;
}MATCHRESULT, *pMATCHResult;

//Structure for reading gabor values.
typedef struct GABORVAL
{
    float Features[INDEX_GABOR_IMAGE][INDEX_GABOR_IMAGE];
}GaborVal, *GabVal;
typedef vector<GABORVAL> GABORVEC;

//Structure for the 10 nearest neighboring classes based on euclidean distance
typedef struct CLASSINFO
{
    //Further information i.e. to be added in a particular record
    CString ClassName; //Class Name
    double euclid;// Euclidian distance
}CLASSInfo, *pCLASSInfo;

```

continued...

```

typedef struct IMAGE_INFORMATION
{
    long m_nXVal;
    long m_nYVal;
    float ** m_ppImage;
    IMAGE_INFORMATION()
    {
        m_nXVal = m_nYVal = 0;
        m_ppImage = NULL;
    }
}
IMAGE_INFO, *PIMAGE_INFO, **PPIMAGE_INFO;

```

Functionality

CAfisShared class simply provides the facility of declaring and then subsequent modification of the values of global or performance specific parameters of various algorithms used in the system. Along with the algorithm specific parameters there are certain application specific parameters too, which helps in smooth functioning of the system application.

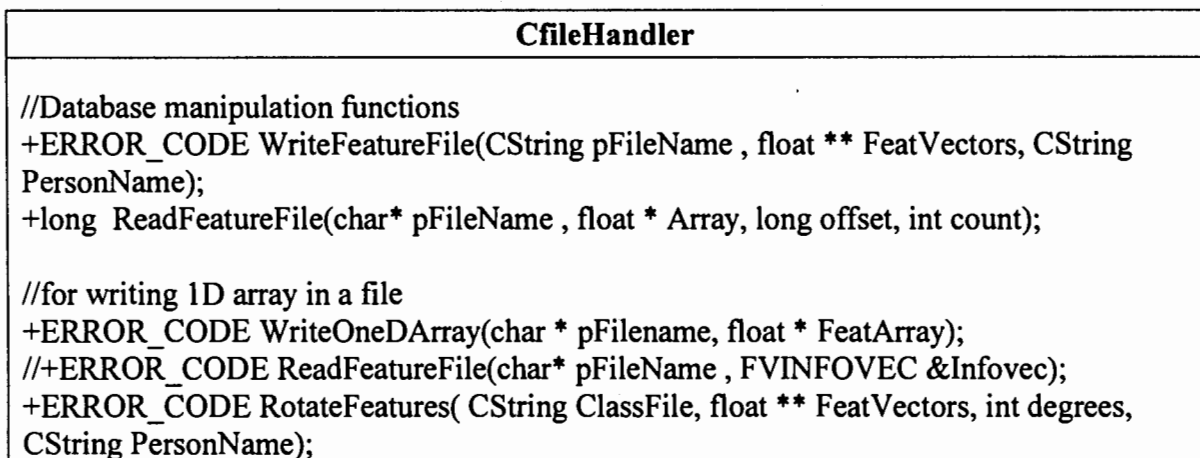
5.8 CFileHandler

Responsibility

This class contains all those file manipulation functions that are frequently used both in feature registration and matching phases/modes.

Design

The following figure illustrates the CFileHandler class object.



Functionality

CFileHandler contains functions for reading and writing data to the database files. There are functions that manipulate data between the database files and various data structures like 1D, 2D array of various data type along with structure data type.

Chapter 6

User Application

6. AFIS Application

As AFIS (Automated Fingerprint Identification System) is purely a biometric based security application, its real time application only requires the input fingerprint impression from the user through a scanner and the system generates a signal, which is acknowledged by the user either as access granted or access blocked. Anyhow for demo purpose a clear and concise graphical user interface is provided for easy interaction with the application and to show the working and features of the system.

6.1 Splash Screen

As the application is launched the very first screen that appears is the splash screen as shown in figure 6.1. It narrates certain information regarding the project title, its developer and supervisor, etc.

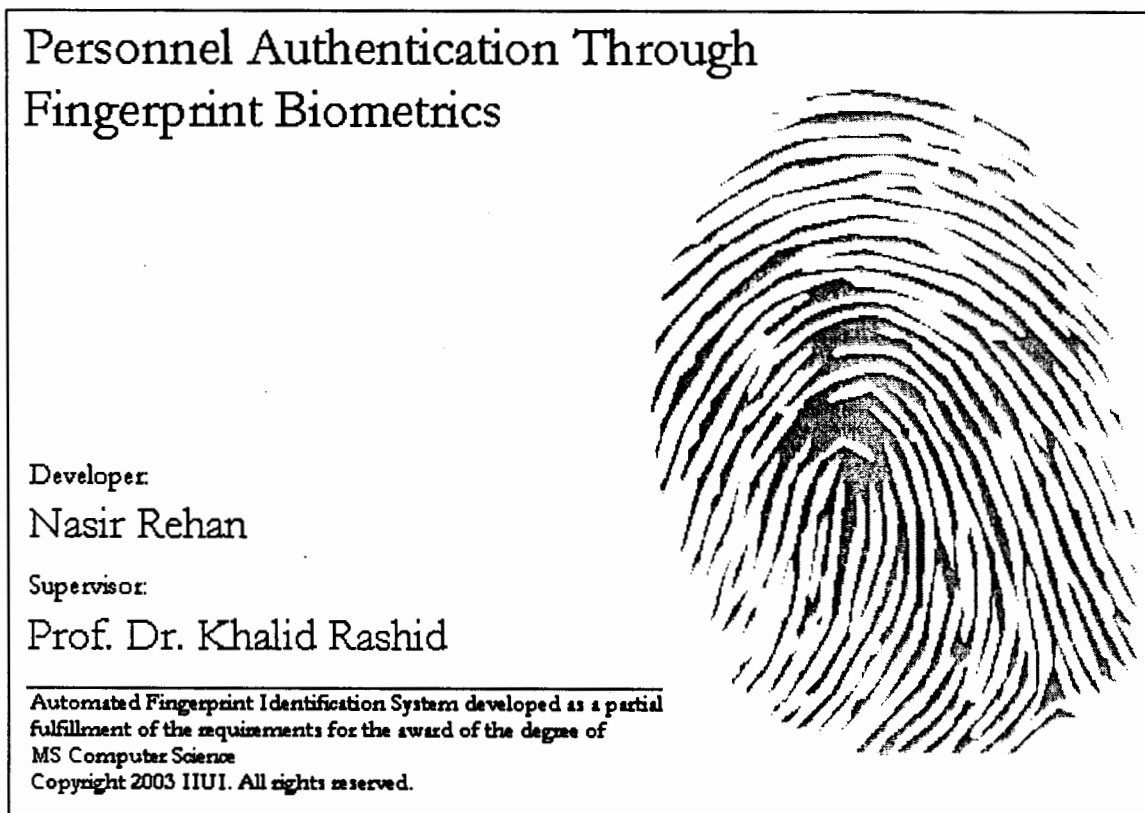


Figure 6.1 Splash Screen

6.2 Main Interface

The main window of our system through which all of the operations are performed is shown in figure 6.2.

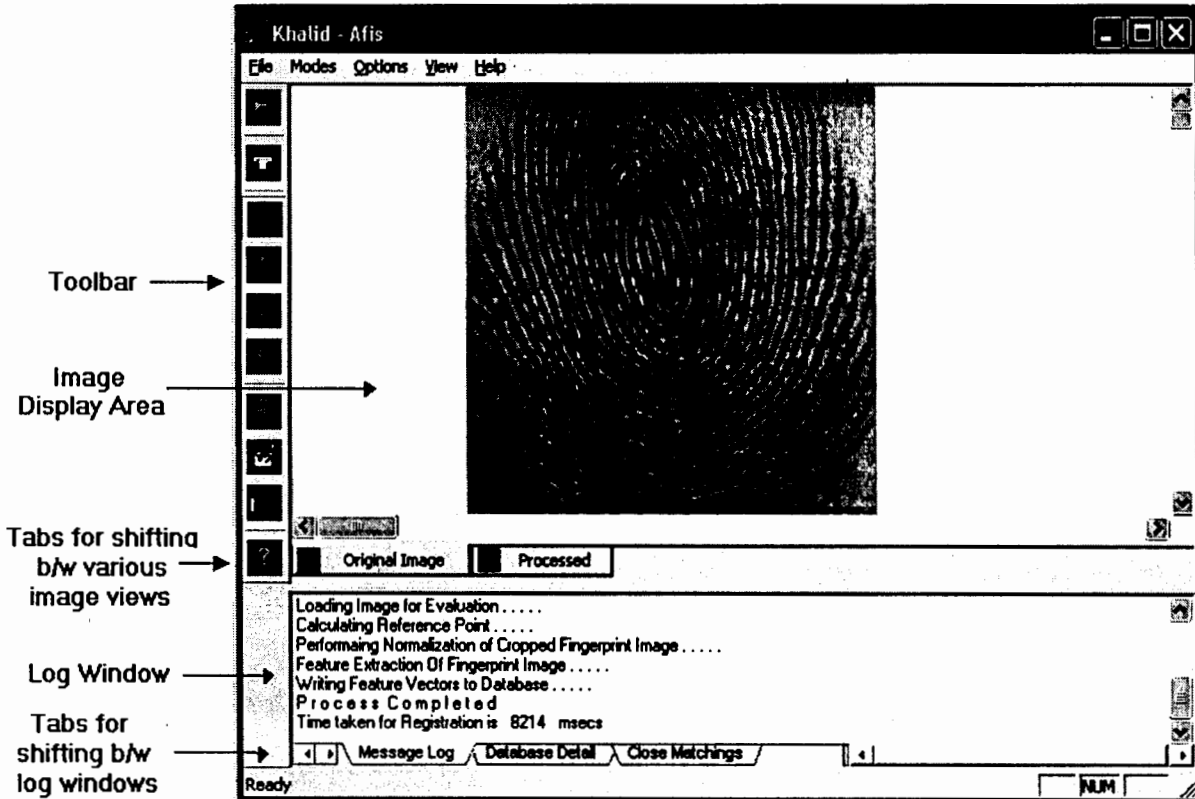


Figure 6.2 Main Interface of Application

The application's main window has various components and sections, which facilitate easy interaction with the software. The user needs to know the details for performing operations in the software application. These are all explained in detail in the following sections.

6.3 AFIS Toolbar

The toolbar at the left of the window provides a fast way of executing the various operations available to the user. Figure 6.3 is a close up of the toolbar icons.

Each of the icons on the toolbar are explained in sequence they appear on the toolbar from top to bottom.

1. **File Open Button** – This enables the user to load new fingerprint image from a folder for system operation
2. **Scanner Button** – This option enables the user to take input via scanner (not operational yet)
3. **Enrollment Button** – This starts the process of registering a certain person with the system.
4. **Recognition Button** – This is for starting identification process i.e. the current file load is matched with the database.
5. **Nearest 5** – This feature provides the facility of coming up with five closest matches to a particular person.
6. **Generalization** – This feature provides the facility of collectively enrolling all the fingerprint images placed in a certain folder.
7. **Settings Dialog** – Dialog box that provides the facility of manipulating performance settings of the system (semi-operational).
8. **Clear Database** – An option for clearing/deleting all the records present in the system.
9. **Clear Log-Window** – Option for flushing all the messages generated by system in the log window
10. **Help** – An option for facilitating user in better handling the application



Figure 6.3 Toolbar

6.4 Image Display Area

This Image display area depicts the current fingerprint impression loaded on to the application. When the user loads a new image to display, the screen is updated to display the new image in this region.

6.5 Image View Tabs

This is for navigating between various views, which are incorporated to allow the user to view the effect of various image-processing techniques as they are applied on the

original image. We have only provided two views one being the original image as it is loaded and then a reference point marked on that image.

6.6 Log Window Tabs

These tabs are used to switch between various log/output windows. These log windows inform the user of various processing tasks the system is going through in either of its operating modes.

6.6.1 Message Log

This window displays the text messages, which are representative of the various processing tasks performed by the system.

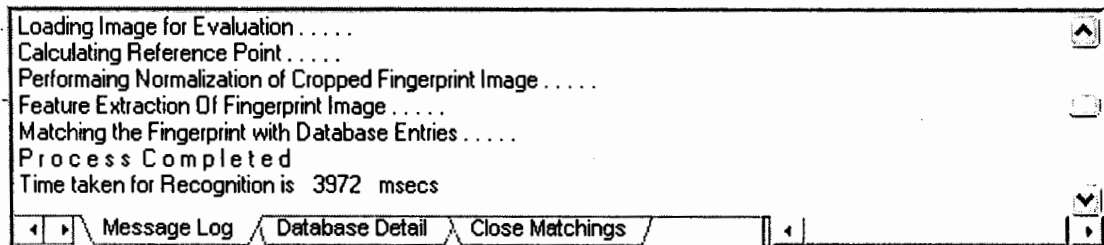


Figure 6.4 Message Log Window

6.6.2 Database Detail Log

This section of the log window displays the result of matching operation. The information required as an output during matching process is illustrated by figure 6.4.

Current Person	Matched With	Euclid Distance	Access
Abdul-1	Karim Khan	600315.807453	Granted
Adnan-1	Ali Khan	478836.556845	Granted
Arshad-2	Jahanzeb Khan	524721.531677	Granted
Abdul-1	Karim Khan	600315.807453	Granted

At the bottom, there is a progress bar and navigation arrows.

Figure 6.5 Database Detail Log Window

original image. We have only provided two views one being the original image as it is loaded and then a reference point marked on that image.

6.6 Log Window Tabs

These tabs are used to switch between various log/output windows. These log windows inform the user of various processing tasks the system is going through in either of its operating modes.

6.6.1 Message Log

This window displays the text messages, which are representative of the various processing tasks performed by the system.

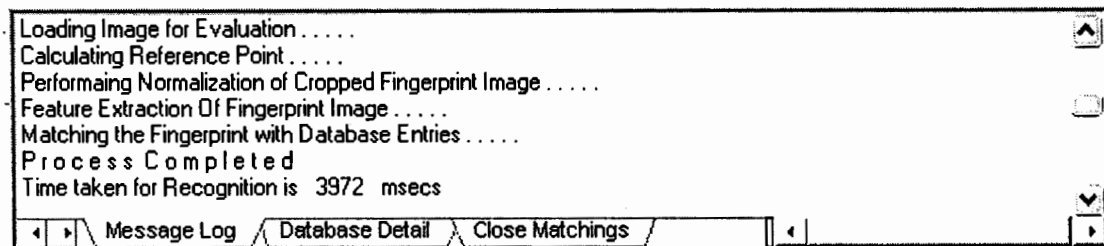


Figure 6.4 Message Log Window

6.6.2 Database Detail Log

This section of the log window displays the result of matching operation. The information required as an output during matching process is illustrated by figure 6.4.

Current Person	Matched With	Euclid Distance	Access
Abdul-1	Karim Khan	600315.807453	Granted
Adnan-1	Ali Khan	478836.556845	Granted
Arshad-2	Jahanzeb Khan	524721.531677	Granted
Abdul-1	Karim Khan	600315.807453	Granted

Figure 6.5 Database Detail Log Window

6.6.3 Close Matching Log

This section of the log window displays the result of the five nearest matchings to a particular person matching operation. The information required as an output during nearest matching process is illustrated by figure 6.5.

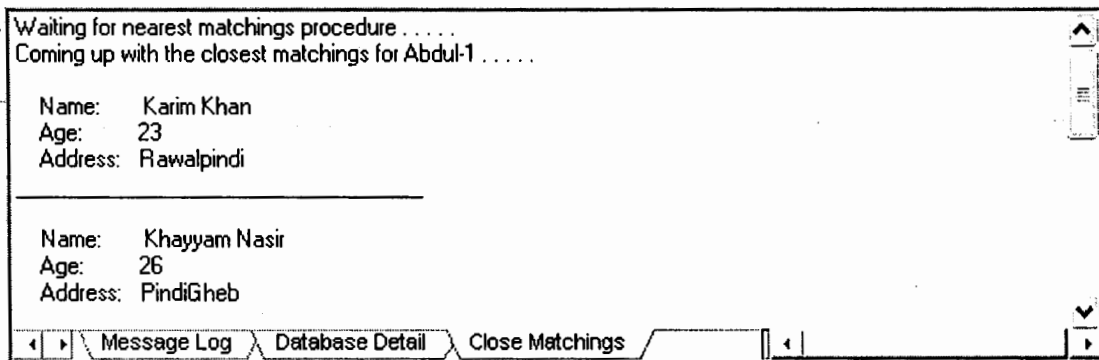


Figure 6.6 Close Matchings Log Window

6.7 Person Information Dialog Box

This dialog box appears when a person is enrolled with the system. This dialog box is used for getting some information about the person before actually proceeding with the feature extraction process. This simply takes person's name and some additional information which is to be stored in the database.

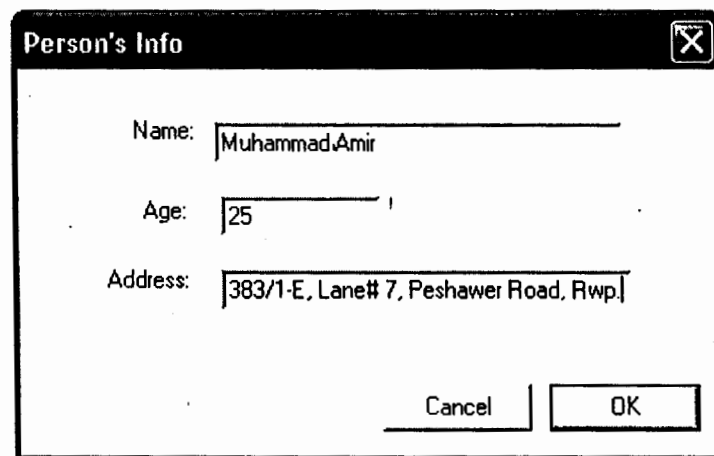


Figure 6.7 Persons Information Dialog Box

6.8 Options Dialog Box

This setting dialog box helps in manipulating various performance issues. The information content of this dialog box is illustrated by figure 6.7.

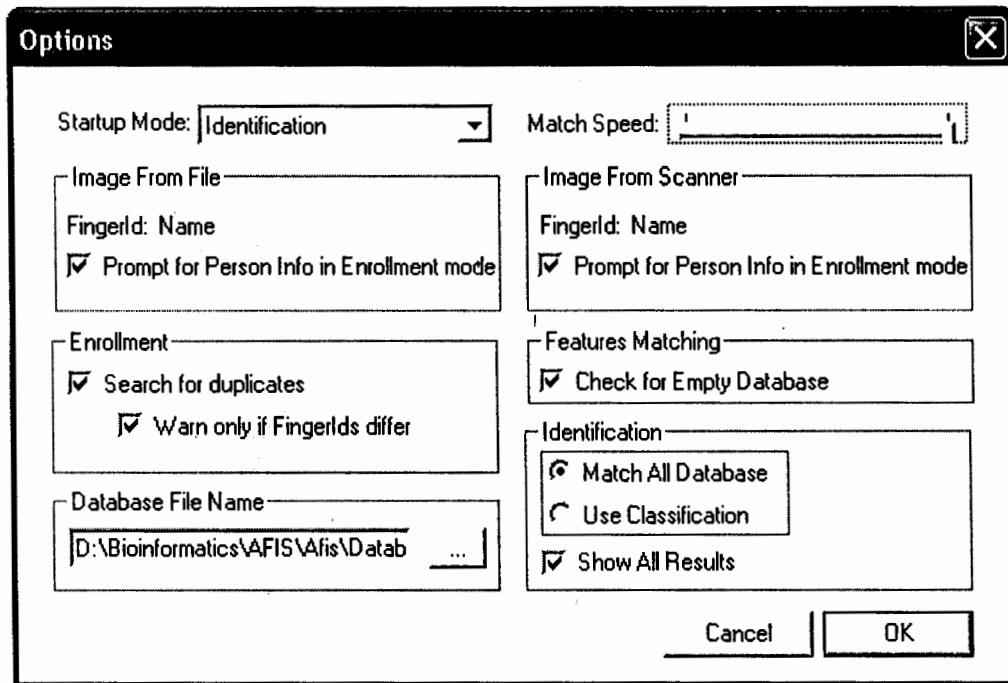


Figure 6.8 Options Dialog Box

6.9 Performing System Operations Via Interface

The user friendliness feature of the system can be judged by the number of clicks required to perform a desired operation. Our system hardly requires more than four clicks for any operation i.e. either for enrollment or identification. These steps can be categorized in chronological order:

- Select and load the new fingerprint impression by ***File Open Button***.
- Select the desired option by clicking any of the operational button i.e. either ***Enrollment button*** or ***Identification button***.
- Navigate between various views using ***Image View Tabs***.
- Navigate between log windows using the ***Log Window Tabs***.

Chapter 7
System Evaluation

7. System Evaluation

This chapter goes through all of the fingerprint identification's main sections, reference point location, sectorization & normalization, Gabor-filtration, feature vectors, classification, and identification processes and explains the outcome of each of the individual sections, then a holistic evaluation of the whole process is given as the system's efficiency is tested on the test images.

7.1 Fingerprint Database Collection

Real time testing of biometric projects using hardware apparatus does add a lot to the fame of the individuals developing these systems. So it was desired to arrange for the scanner and other required commodities essential for real time testing. For hardware availability, an organization had taken the responsibility but at the last moment they backed out. So a database of fingerprints has been arranged for system evaluation. This database was collected from the site of Neuroteknologija Ltd., which is a provider of biometric solutions. As these images are gathered using their scanners so it is assumed that if our system is connected to such a scanner and the images thus obtained are of the same quality and category then our results regarding system's efficiency are very much reliable.

In our pursuit of cost effective, reliable and efficient scanner we came across Kinetic Sciences Inc. Their KC-901 scanner did fulfill all our operational requirements. It's cost is affordable but it is quiet disheartening to know that they provide the hardware to their corporate partners and that too in large amount. The point to mention here is that there are certain issues related to sensors, which need to be taken into consideration while opting for a certain scanner. As we haven't employed the scanner in our real time testing of this system so discussion about these issues become irrelevant to us. Interested candidates can refer to Kinetic Sciences Inc. site for such information and also a comparison of various scanner scanners been provided which will help in choosing a particular scanner.

Fingerprint images collected from Neuroteknologija Ltd. contained a total of 125 fingerprint images. These are attributed to 25 people. Each person has 5 different fingerprint impressions of the same finger. Each image is a 400 x 400 pixel in size, 8-bit grayscale image scanned at 500dpi. The database is divided into two categories:

- **Training Set**
- **Test Set**

7.1.1 Training Set And Test Set

The best fingerprint image from all of the five impressions of each of the person are marked and selected under the training set. And the other four impressions are used

for testing so they are classified as test sets. The images in the training set are registered with the system and fingerprint images from the test set are used for matching. These images were of mixed qualities. Also their ratio in the database was not in natural proportion so it also adds to our benefit as it is expected that our system will perform better in real time environment. Each fingerprint image has little rotation factor associated with it. but our system's performance doesn't deteriorate, we don't have to worry about the rotation factor at this stage. If required this factor could easily be incorporated as our filterbank representation scheme does cater this requirement. Alteration in the pattern in which the feature vectors are stored in the database can answer this requirement. It will just add an extra burden to the matching phase, as more matching time will be consumed due to higher number of records per person in database.

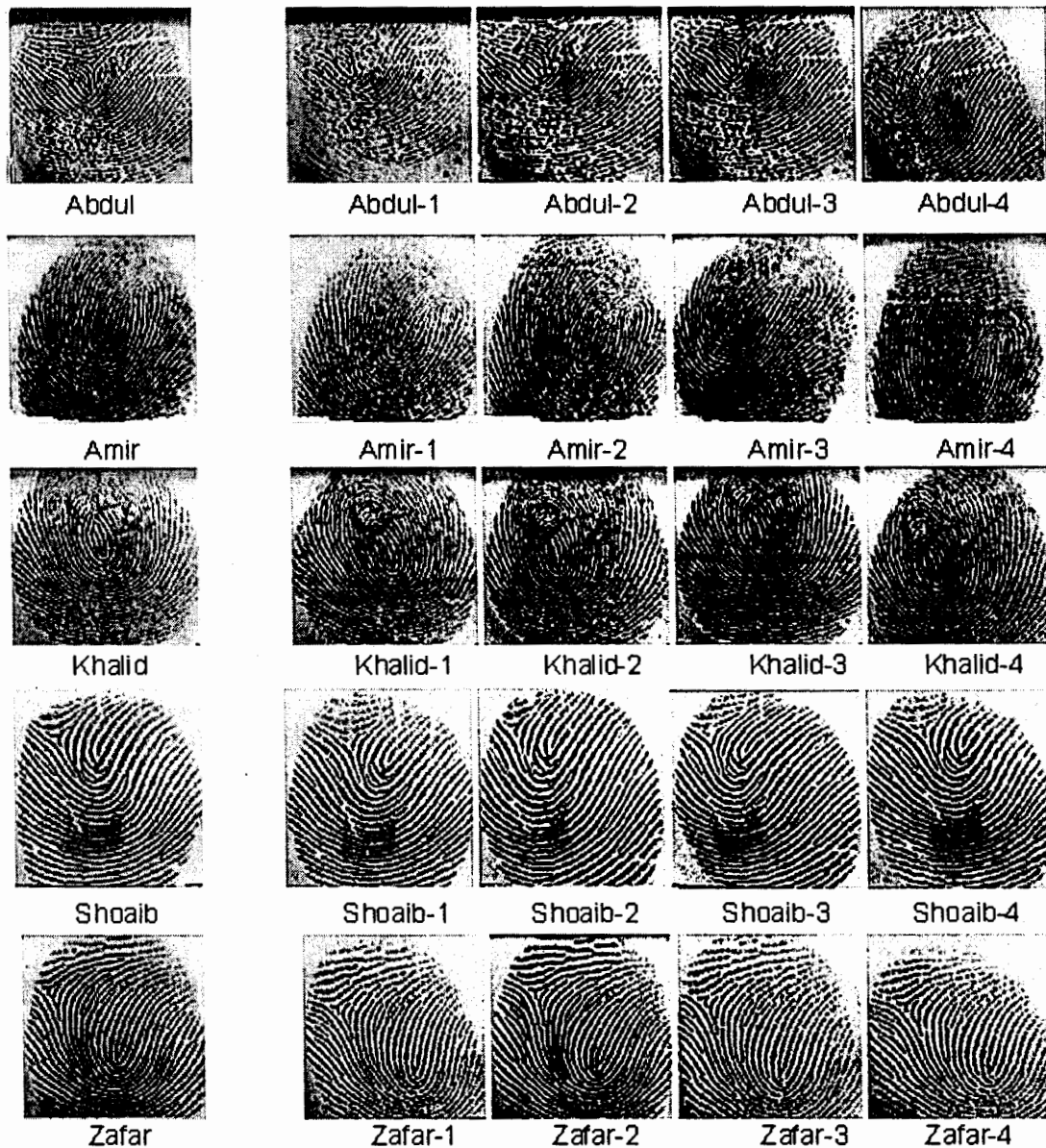


Figure 7.1 Sample Images from Database

7.2 FAR And FRR As Performance Measures

Two main properties of the system are worth to measure which are False Acceptance Rate (FAR i.e. how many impostors that are falsely accepted) and False Rejection Rate, (FRR, i.e. how many authorized users that are falsely denied). Naturally there is a tradeoff between FAR and FRR (see Figure 7.2). A decrease in one of them generally results in an increase in the other. The main reason for performing these tests is to choose tolerances for the verification/identification steps in the algorithm. The FAR and FRR rates are more or less a result thereof. Obviously the system should have as low FAR and FRR as possible. The main approach used is to select tolerances that result in no falsely accepted fingerprints and also minimize FRR among these tolerances. Since there are two tolerances to adjust there are many combinations to test. A small increase in one and a small decrease in the other may give a totally different result than with the original tolerances.

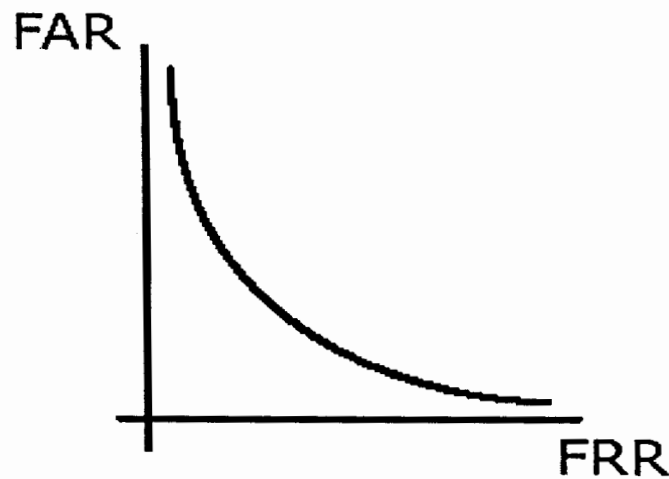


Figure 7.2 Asymptotic Behavior of FAR and FRR

Initially three security levels (High, Medium and Low) are supposed to be implemented to relate the tolerances to the intended use of the application. The difference between the High and the Medium level is that the tolerances of the High security level are set lower. Resulting in a lower FAR, i.e. a more secure system, and a correspondingly higher FRR. In order to motivate the Low security level, imagine the following scenario. The police force intends to heighten weapon security and a new fingerprint controlled gun holster, that only allows the authorized police man to use the weapon, is evaluated. If the policeman is rejected access to his gun in a critical situation his life may be in danger. Thus, it is vital for the policeman to be able to use his gun at all times and a false rejection is therefore not acceptable. This means that tolerances here are set higher, since low FRR is more important than a low FAR. This system is configured for operating at a medium security level.

7.3 Experimental Results

To establish the verification accuracy of our fingerprint representation and matching approach, each fingerprint image in the database is matched with all other fingerprints in the database. A matching is labeled correct if the matched pair is from the same finger and incorrect, otherwise. None of the genuine (correct) matching scores is zero indicating that the images from the same finger did not yield an identical FingerCode because of the rotation, distortion, and inconsistency in reference point location. For the available database, a total of 92 matchings are performed. The probability distribution for genuine (correct) matches as estimated to be 85 correct matches and the imposter distribution is estimated to be 7 as depicted by figure 7.3.

Enrolled Images	Total Matchings	Correct Matchings	False Matchings
23	92	85	7

Figure 7.3 Matching Results involving database of Test and Training Sets

If the Euclidean distance between two FingerCodes is less than a threshold, then the decision that “the two images come from the same finger (i.e. finger impressions of the same person)” is made, otherwise a decision that “the two images come from different fingers” is made. Different decision thresholds lead to different values of FAR and FRR as illustrated by figure 7.4 below.

Threshold Value	False Accept Rate (%)	False Reject Rate (%)
30	0.10	10.32
35	1.07	7.87
40	4.59	2.83

Figure 7.4 False Accept Rates & False Reject Rates with different Threshold Values

Note: These Threshold values correspond to security levels (i.e. High, Medium & Low)

7.4 Limitations

The filterbank approach suffers from a number of disadvantages and more research is needed in the following areas to improve the representation and matching:

- i) The registration is based on the detection of the reference point. Even though this multi-resolution (i.e. varying window size e.g. 5 & 10 pixels) reference point location algorithm is accurate and handles the poor quality fingerprint images

gracefully, it fails to detect the reference point in very low quality images leading to either a rejection of the image or even worse, a false rejection in the verification system.

- ii) The current implementation of the filterbank representation is not rotational invariant. The rotation is handled in the matching stage by rotating the FingerCode itself. However, due to quantization of the rotation space and generation of multiple alignment hypotheses, the false accepts increase.
- iii) The current implementation of filterbank representation extraction takes longer than a typical minutiae-extraction algorithm. More than 90% of the total compute time for verification for the images database is taken by the convolution of the input image with 6 Gabor filters. The convolution operation can be made significantly faster by dedicated DSP processors or performing the filtering in the frequency domain.
- iv) The current matching algorithm is very simple. An implementation of a smarter matching algorithm should be able to improve the verification performance. For example, the match resulting from each sector can be weighed differently based on image quality and a quantitative measure of the nonlinear distortion in the sector.

7.5 Conclusion

This project is a good learning experience for us. Besides learning new Signal Processing topics such as Pattern Recognition, Gabor Filtering, and classification schemes, we also learned that in real life, projects such as this one need other factors such as collective team effort, time management, and modular programming.

Given the chance to build a similar system again, we would like to come up with another technique for determining center point, which will be more consistent and robust. To speed up the system FFT based Gabor Filtering would be thought of as a replacement for the commonly used normal convolution process. Efforts may also be applied to save pre-FFTed Gabor filters that will reduce processing time significantly. By achieving this goal we'll be able to save 60% of the computation time.

We would like to implement the system in hardware with a DSP applications specific chip. This would serve the purpose of time-saver relieving much of the strain in memory transferring issues and debugging.

Throughout the development of this project, good design and providing functionality took preference over optimizations. After running the program it can be seen that all the image-processing techniques require large amounts of calculations, which could benefit performance wise from optimizing the code. This could be achieved

These experiments are carried out on a limited number of subjects, and it would make a good extension to be able to ascertain a larger data set to get a better measure of how effective the fingerprint mechanisms are for identification. So the need is for creating a larger fingerprint data set for test to be tried against.

Bibliography and References

Bibliography and References

Research Papers

- [1] A. K. Jain, L. Hong, S. Pankanti, and Ruud Bolle, "An Identity Authentication System Using Fingerprints," *Proceedings of the IEEE*, Vol. 85, No. 9, pp. 1365-1388, 1997.
- [2] Anil K. Jain, S. Prabhakar, Lin Hong, "Fingercode: A Filterbank for Fingerprint Representation and Matching," MSU CPS Technical Report Library.
- [3] Anil K. Jain and F. Farrokhnia, "Unsupervised Texture Segmentation Using Gabor Filters," *Pattern Recognition*, Vol. 24, No. 12, pp. 1167-1186, 1991.
- [4] E. R. Henry, *Classification and Uses of Fingerprints*, London: Routledge, pp. 54-58, 1900.
- [5] L. Hong and Anil. K. Jain, "Classification of Fingerprint Images," 11th Scandinavian Conference on Image Analysis, June 7-11, Kangerlussuaq, Greenland, 1999.
- [6] N. Ratha, Shaoyun Chen, and A. K. Jain, "Adaptive Flow Orientation-Based Feature Extraction in Fingerprint Images," *Pattern Recognition*, Vol. 28, No. 11, pp. 1657-1672, 1995.
- [7] Asker M. Bazen and Sabih H. Gerez "Segmentation of Fingerprint Images", University of Twente, Department of Electrical Engineering, Laboratory of Signals and Systems, Veldhoven, Netherland 2002.
- [8] Mr Anthony Lum "Fingerprint Recognition" Department Of Electrical Engineering, University Of Queensland, St Lucia, 4067. October 13 th 1999.
- [9] R.Klette and P.Zamperoni, *Handbook of Image Processing Operators*, John Wiley & Sons, New York, NY, 1995.
- [10] *Digital Image Processing*. Rafael C. Gonzalez and Richard E. Woods. 1993 Prentice Hall Press.
- [11] *Digital Image Processing* Jan Teuber, 1992 Prentice Hall Press.
- [12] R.Klette and P.Zamperoni," *Handbook of Image Processing Operators*", John Wiley & Sons, New York, NY, 1995.
- [13] *Software Engineering A Practitioner's Approach (Fourth Edition)* by Roger S. Pressman 1992, McGraw-Hill Companies Inc.

- [14] MATLAB (Online Documentation).
- [15] MSDN Library.

Research Paper

JOURNAL OF APPLIED SCIENCES
A Quarterly Publication of ANSInet

Nasir Rehan
383/1-E, Lane #7, Peshawar Road, Rawalpindi, 46000,
Pakistan

24-Jun-2004

Subject: Comments on Article No. 274-PJAS entitled " Multi-Matcher Based Fingerprint Identification System "

Dear Nasir Rehan,

First of all I would like to thank you for your trust on " JOURNAL OF APPLIED SCIENCES "

We have received the referees comments and according to them your manuscript has been accepted for publication in " JOURNAL OF APPLIED SCIENCES " after suggested modification.

For modification according to the referee's comments we are enclosing the referee's evaluated copy. Please modify it and send it back with revised copy as early as possible.

It is also requested to please send us soft copy on CD (please dont send revised soft copy on floppy disk or through e-mail) so that we can process revised manuscript further.

Your quick response would be highly appreciated.

ANSInet cordially invites to please visit the Journal's website <http://www.ansinet.net>

With best regard..



Muhammad Imran Pasha
Account Manager

Multi-Matcher Based Fingerprint Identification System

Nasir Rehan
nasirrehan@yahoo.com
Ph# 92-51-5470085

and

Khalid Rashid
drkhalid@iiu.edu.pk
Ph# 92-51-9257951

Department of Computer Science
International Islamic University, Islamabad
Pakistan.

Abstract

Fingerprint identification system being the most developed and emphasized among the various practically employed biometric systems, has yet to make a long journey to prove its effectiveness in terms of speed and accuracy. Of various options for increasing systems efficiency, one may be the combination or fusion of various contemporary fingerprint identification systems. This study reveals the possibility for introducing a fingerprint identification system based on different fingerprint matching systems, working in close collaboration. Two different fingerprint representation schemes are evaluated and a merger point for the two is suggested.

Key words: Biometrics, fingerprint, efficiency, evaluation, combination.

Introduction

Fingerprint identification systems are increasingly employed into business, trading and living fields for automatic personal identification. Though various fingerprint representation schemes have been developed and tested over the years but none had been able to yield satisfactory results. For example, a state-of-the-art fingerprint identification and classification algorithm [1] reports accuracies of 92.2% for five and 94.5% for four class classifications. These claims prove to be false when tested in the laboratory environment. Though these may be great achievements but in situations where the desired error rate is negligibly small, then the need for further refining existing technologies becomes inevitable. Among many directions for addressing accuracy issues, there is an option for combining or running in parallel the existing fingerprint identification algorithms and coming up with a final decision (i.e. differentiating between genuine and imposter). This study takes into account two different

fingerprint representation schemes and highlights fusion or merging point for both the representation schemes.

Minutiae-Based Fingerprint Identification System

Most fingerprint representation systems follow the minutiae-based approach. A minutiae-based fingerprint verification system extracts minutiae points (Fig. 1) from the fingerprint images and matching decision is based on the correspondence of minutiae (i.e. it's type and position) between the stored template and current input fingerprint image.



Fig. 1: Two most widely used minutia features
a) Ridge Ending b) Ridge Bifurcation

Minutiae-based fingerprint identification systems use a large number of successive processing steps. In general, following are the steps in a minutiae-based system:

- Orientation Field Estimation
- Segmentation
- Binarization
- Morphological Operations
- Minutiae Extraction
- Post-processing
- Registration of Minutiae Templates

A brief description of the above-mentioned processing steps is given below:

Orientation Field Estimation

The orientation field represents the directional flow of ridges and valleys in a fingerprint image. A number of methods have been proposed to estimate the orientation field of fingerprint image [2]. To calculate the ridge direction, the fingerprint image is divided into a number of non-overlapping blocks and an orientation representative of the ridges in the block is assigned to the block, based on the analysis of grayscale gradients within that block. The complete steps for orientation estimation algorithm are stated in [3].

Segmentation

The task of a fingerprint segmentation algorithm is to decide that which part of the image belongs to the foreground, originating from the contact of a fingertip with the sensor, and which part to the background. Accurate segmentation is especially important for reliable extraction of features like minutiae and singular points. Therefore, the main goal of the segmentation algorithm is to discard the background to reduce the number of false features originated due to noise. Segmentation algorithm by Asker M and Sabih H [4] gracefully segments the fingerprint image.

Binarization

The method of transforming grayscale pixel values to either black or white pixels corresponding to a ridge and valley, respectively is called binarization. The process can be carried out using a multitude of techniques [5]. Binarized image can be obtained using either global or adaptive thresholding. A brief description of the two is given below.

Global Thresholding

Global Thresholding involves the formulation of a histogram consisting of the number of pixels versus the pixel value. In this technique, the highest black and white pixel values within grayscale image are determined. The two major peaks found are assumed to be the most commonly used dark and light pixels. The middle range pixels will then be used to discriminate between black and white pixels for binarization (Fig. 2).

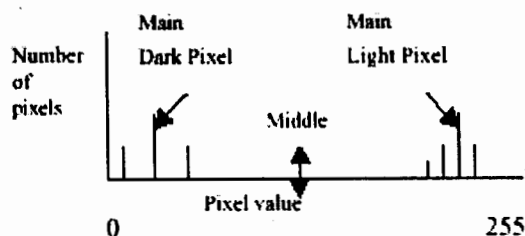


Fig. 2: Histogram for calculation of most numerous light and dark pixel values.

Adaptive Thresholding

The method, also known as contrast enhancement binarization, involves passing a low pass filter over the image and using the resulting grayscale pixel number to discriminate between black and white pixels. The low pass filter does not process edges that are one pixel wide in the image. Low pass filtering involves a spatial convolution process within a window (Fig. 3). These windows can be quite large. The image is convolved with this filter and the process is repeated until a new binarized image is obtained.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Fig. 3: Convolution matrix (3x3) for Low-Pass filtering

Morphological Operations

Various morphological operations are performed on binarized image. The most important of them include thinning and spur removal. A thinned fingerprint image is one in which each ridge is one pixel in width. Thinning is achieved by successive deletion of pixels from all sides of the image. Each of the four sides are eroded away according to some set template (Fig. 4) [6].

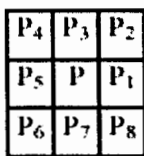
	NORTH	SOUTH	EAST	WEST
0 0 0	1 1 X	0 X 1	X X 0	
X 1 X	X 1 X	0 1 1	1 1 0	
X 1 1	0 0 0	0 X X	1 X 0	
X 0 0	X 1 X	0 0 X	X 1 X	
1 1 0	0 1 1	0 1 1	1 1 0	
X 1 X	0 0 X	X 1 X	X 0 0	

Fig. 4: Eight Matrices based Thinning Method

If the image matches with template, the middle pixel is removed. Initially the two north images are processed, and then the other compass points are overlaid. Once all eight matrices have been sampled on the entire image, the process is repeated again on the newly formed image. Processing only stops when no more pixels can be deleted. Similarly spur operation is performed on thinned image to remove false bifurcations and ends caused by thinning. This removes some real end points from their original locations too. *A.Rosenfeld and A.C.Kak* explained how successive deletion works [7].

Minutiae Extraction

Once the thinned ridge map is available, minutia location is not a tiresome job then. The technique uses a sample window (Fig. 5) to detect key minutia features (ridge endings and bifurcations).



The matrix checks for pixel values in each neighborhood of pixel P1 and then compares the outcome with the characteristics table.

Fig. 5: Matrix for traversing through Thinned Image

Ridge pixels with one ridge pixel neighborhood are identified as ridge endings and those with three ridge pixel neighbors are identified as ridge bifurcations (Fig. 6 and 7). However, the entire minutia thus detected are not genuine due to image processing artifacts and the noise in the fingerprint image. The excess bifurcations and endpoints associated with noise must be removed to maintain accuracy levels [6].

CN	Characteristic
0	Isolated Point
1	End Point
2	Continuing Point
3	Bifurcation Point

Fig. 6: Table for Checking Ridge Characteristics

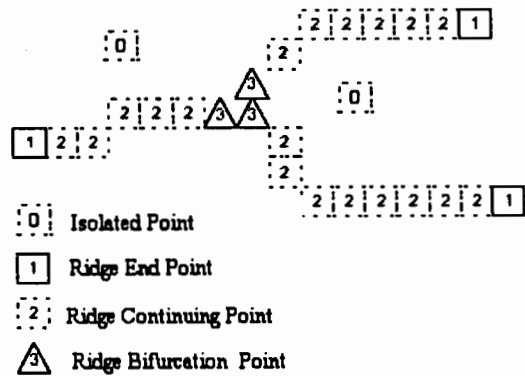


Fig. 7: Thinned Ridged map with Minutiae being Marked

Post-processing

At this stage, the genuine minutiae are gleaned from the extracted minutiae using a number of heuristics. For instance, too many minutiae in a small neighborhood may indicate noise and they could be discarded. Similarly very close ridge endings oriented anti-parallel to each other may indicate spurious minutia generated by a break in the ridge either due to poor contrast or a cut in the finger. Also two very closely located bifurcations sharing a common short ridge often suggest extraneous minutia generated by bridging of adjacent ridges as a result of dirt or image processing artifacts.

Thus all detected minutiae are passed through a process in which a minutia point is selected and all its neighbors within a specified radius are marked and these minutiae points including the center or selected minutia point are all deleted (Fig. 8). In the end there are only 70 - 90 minutiae points left for registration with the system.

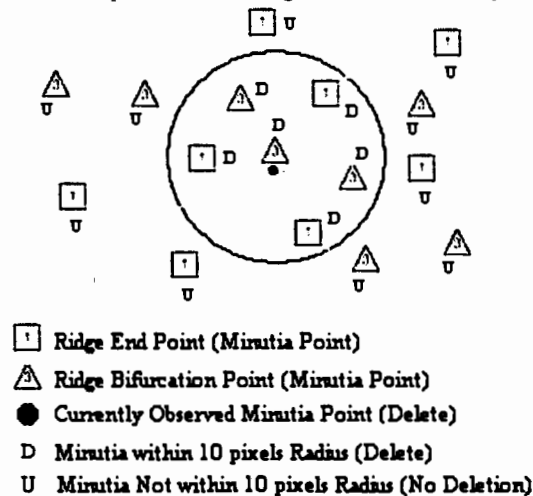


Fig. 8: Minutia Deletion Strategy

Minutiae Registration

Feature extraction stage in minutia-based system is concluded by registration of acquired minutiae. Many features regarding a particular minutia point are recorded in the database. These features include information like total number of minutiae points detected, position of each minutia point, minutiae type, its angle (direction) and most important of all, the minutia's position relative to other minutia points in the same fingerprint. A fingerprint can have up to 60-90 minutiae detected, and a valid match is generally accepted between two fingerprints if both prints have 8 to 17 common minutia points.

Total Minutiae Points	Minutia Position (x, y)	Minutia Angle/ Direction	Minutia Type (RE/RB)
-----------------------	-------------------------	--------------------------	----------------------

Fig. 9: Most common Minutiae Information in Database

A microscopic approach, called minutia matching is commonly used for matching in minutia based fingerprint systems. Matching procedure involve each minutia's relative position to other minutia in a fingerprint (Fig. 9).

Challenges Associated with Minutiae-based System

There are various factors or constraints, whose net affect adversely hinders the efficiency of minutia-based systems. Segmentation of fingerprint-textured images still remains a difficult problem in fingerprint identification systems. Without significant segmentation, the system's efficiency is questioned, as false minutiae are detected off the original fingerprint image. The most successful approach towards fingerprint segmentation includes Gabor Filters, which is the prime characteristic of Filterbank based fingerprint system [8], but this has its own computational constraint which will affect the time within which the system generates the matching result.

The thinning algorithms haven't yet been designed to perfection. The correspondence between the original fingerprint image and the resulting thinned image has to be perfect, as it is at this stage the minutia points are determined and stored for use in matching stage. The contemporary thinning algorithms engulf many true minutia points and generate many false minutia points.

The heuristics employed for the deletion of minutia points further add to the deficiency of minutia-based systems. Many of the valid minutia points are deleted and certain false minutiae points survive for registration in the system database.

Filterbank-based Fingerprint Identification System

Filterbank based fingerprint identification system not only takes into account the local anomalies (i.e. minutia points) in the ridge structure but also the global pattern of ridges and furrows, inter-ridge distances, and overall pattern of ridge flow. This system is based on the fact that most textured images contain a limited range of spatial frequencies. Textured regions possessing different spatial frequency, orientation and phase can be easily discriminated by decomposing the image into several spatial frequency and orientation channels. Jain and Farrokhnia derived a global representation of texture by decomposing the input image into different frequency and orientation components using a Gabor Filterbank. This representation for oriented texture of fingerprints was inspired by Daugman's work on iris recognition and the success of Gabor Filterbank [8, 9, 10].

The four main steps in this filterbank based identification algorithm involve [8]:

- Reference point determination in fingerprint image and cropping,
- Tessellation of region around the reference point,
- Filtration of the tessellated area in six different directions using bank of Gabor Filters,
- Computation of Average Absolute Deviation (AAD) to yield feature vectors, also called the FingerCode, inspired by Daugman's IrisCode.

A brief description of the above-mentioned processing steps is given below:

Reference Point Location

The point of maximum curvature of the concave ridges in the fingerprint image is taken as the reference point. Center point location algorithm [4, 8] gracefully determines the point of most curvature by determining the normals of each fingerprint ridge and then following them inwards towards the center. The algorithm has the following steps:

- Apply a pixel-wise adaptive 2-D Gaussian lowpass Wiener filter to reduce noise from the fingerprint image. The filter uses neighborhood of size 5×5 to estimate the local gradient mean and standard deviation.
- Divide the input fingerprint image into non-overlapping blocks of size 10×10 .

- Determine x and y magnitudes of the gradient (G_x and G_y) at each pixel in each block, by taking the average of the two neighboring pixels.
- Apply the same 2-D Gaussian lowpass filter on the x and y gradients as above to smooth out the gradients.
- With each block, compute the slope perpendicular to the local orientation of each block using equation (1).

$$\Theta = \frac{1}{2} \tan^{-1} \left(\frac{\sum_{i=1}^{10} \sum_{j=1}^{10} 2G_x(i,j)G_y(i,j)}{\sum_{i=1}^{10} \sum_{j=1}^{10} (G_x^2(i,j) - G_y^2(i,j))} \right) + \frac{\pi}{2} \quad (1)$$

- Only looking at blocks with slope values ranging from 0 to $\pi/2$, trace a path down until a slope is encountered that is not ranging from 0 to $\pi/2$ and mark that block (Fig. 10).

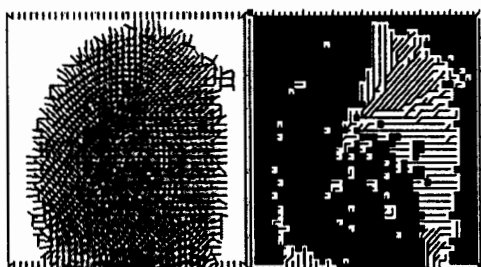


Fig. 10: Figure depicting inward slope determination

- The block that has the highest number of marks will compute the slope in the negative y direction and output an x and y position which will be the center point of the fingerprint (Fig. 11).



Fig. 11: Picking up block with higher number of markings.

Once this point is determined, fingerprint image is then cropped to a 223×223 pixels centered on the calculated reference point.

Tessellation

The region of interest in the fingerprint image is the collection of all sectors around the center point where each sector is computed in terms of a particular radius and angle. This demarcation of the cropped area in sectors is called tessellation. The cropped fingerprint image is divided into 5 concentric bands centered on the reference point. Each band has a radius of 20 pixels except for the inner most, which has a radius of 12 pixels. So the total radius of the sectorization is 111 pixels similar to that of cropped image. Each band is divided into 12 sectors (Fig. 12).



Fig. 12: Reference point marked and area of interest Tessellated.

The sectorization is performed as 6 equi-angular Gabor filters [8] are used for feature extraction. These Gabor filters are aligned with the 12 wedges formed by the bands. In other words, each sector will capture information corresponding to each Gabor filter. The center band is ignored because it has very small area to be of any use. The radius of the sectorization is chosen to avoid the effects of circular convolution in applying Gabor filters. Another reason for sectorization is the normalization process. Each sector is individually normalized to a constant mean and variance to eliminate variations in gray-scale values in the fingerprint pattern, which are incurred due to scanning noise and pressure variations.

Equation (2) is used for normalizing pixel intensities in each sector of the cropped fingerprint image.

$$N_i(x, y) = \begin{cases} M_0 + \sqrt{\frac{V_0 \times (I(x, y) - M_i^2)}{V_i}}, & \text{if } I(x, y) > M_i \\ M_0 - \sqrt{\frac{V_0 \times (I(x, y) - M_i^2)}{V_i}}, & \text{otherwise} \end{cases} \quad (2)$$

M_0 is constant mean and V_0 is constant the variance, both having values = 100. I is sector number, M_i is the mean of sector, and V_i is the variance of sector.

Gabor Filtration

The normalized image is then passed through a bank of Gabor filters. Each filter is performed by producing a 23 x 23 filter image for 6 angles (0, $\pi/6$, $\pi/3$, $\pi/2$, $2\pi/3$ and $5\pi/6$), and convolved with the fingerprint image. Spatial domain convolution is rather slow, so multiplication in the frequency domain is preferred. However, this involves more memory to store real and imaginary coefficients. The purpose of applying Gabor filters is to remove noise while preserving ridge valley structures and to provide information contained in a particular direction in image. A fingerprint convolved with a 0-oriented (0 degree) filter accentuates those ridges that are parallel to that angle (0 degree) and smoothes out the ridges of all other directions. Filters tuned to other angles work in similar manner. These six directional-sensitive filters capture most of the global ridge directionality information as well as the local ridge characteristics present in fingerprint [8].

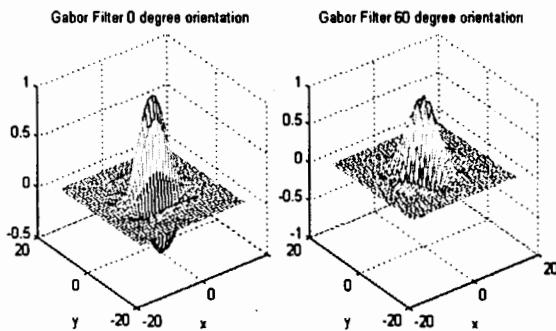


Fig. 13: 0 and 60 degree oriented Gabor Filter.

Equation (3) is the definition of Traditional Gabor filter [8].

$$G(x, y, f, \theta) = \exp \left\{ -\frac{1}{2} \left[\frac{x'^2}{\delta_x^2} + \frac{y'^2}{\delta_y^2} \right] \right\} \cos(2\pi f x') \quad (3)$$

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

$$y' = x \cos \theta - y \sin \theta$$

The parameters, δ_x and δ_y are Gaussian constants whose values are empirically determined and set to 4.0. Very small value will make the filter ineffective in removing noise while too large a value will destroy ridge and furrow details. The filter frequency f is set equal to average inter ridge frequency ($1/k$), where k is the average inter-ridge distance which is determined to be on average 10 pixels in 500 dpi image.

Feature Vector

After obtaining 6 filtered images, the variance of pixel values is calculated in each sector. Higher variance value in a sector means that the ridges in that sector are parallel in direction to that of applied Gabor filter. Whereas the low variance value indicates that the ridges are orthogonal to the angle of applied Gabor filter. So the filtering will smooth out the orthogonal (non-parallel) ridges. Corresponding to tessellation, a total of 360-variance values (6 angles x 60 sectors) are extracted known as feature vectors of the fingerprint.

Equation (4) is used for calculating variance values.

$$V_{i\theta} = \sqrt{\sum_{K_i} (F_{i\theta}(x, y) - P_{i\theta})^2} \quad (4)$$

Where $F_{i\theta}$ represents the individual pixel value in the i th sector. $P_{i\theta}$ is the mean of the pixel values. K_i is the number of pixels in the i th sector.

Challenges Associated with Filterbank-based System

Filterbank based representation [8] is a newly emerging technique. Though it is just in its nurturing stage but still it has achieved remarkable success. Various factors that cause setback to its efficiency are:

First step in this fingerprint representation is the reference point determination (core point location) which itself, in certain cases is very much unreliable. Even though this multi-resolution reference point location algorithm is accurate but it fails to detect correct reference point in very low quality images leading to a reject or even worse, a false rejection/matching in the identification system. The reference point location algorithm also has higher error rate in consistently locating the reference point in arch type fingerprints due to the absence of singular points (maximum curvature ridge). This factor can be addressed by introducing multiple reference points (core and delta point location).

The current implementation is not rotation invariant. For this reason the rotation is handled in matching stage,

which adds to the time for matching thereby hindering system's efficiency.

The current implementation of Filterbank representation extraction takes longer than a typical minutia-extraction algorithm. This is due to a large bank of Gabor filters (depending on the number of angles applied i.e. 6 or more angles) convolved with the original image. Gabor filtering is an expensive process in terms of computation. More than sixty percent of systems operational time is attributed to Gabor filtering stage.

Replacement can be thought of a single rotation invariant Gabor filter that not only accentuates the ridges in all directions in a single convolution but will also reduce the size of Finger-Code (feature vectors) used for fingerprint matching. For the time being, convolution can be made significantly faster by dedicated DSP processors or performing the filtering in the frequency domain. Furthermore the presently employed matching algorithms are simple. An improved matching algorithm would improve the overall systems performance.

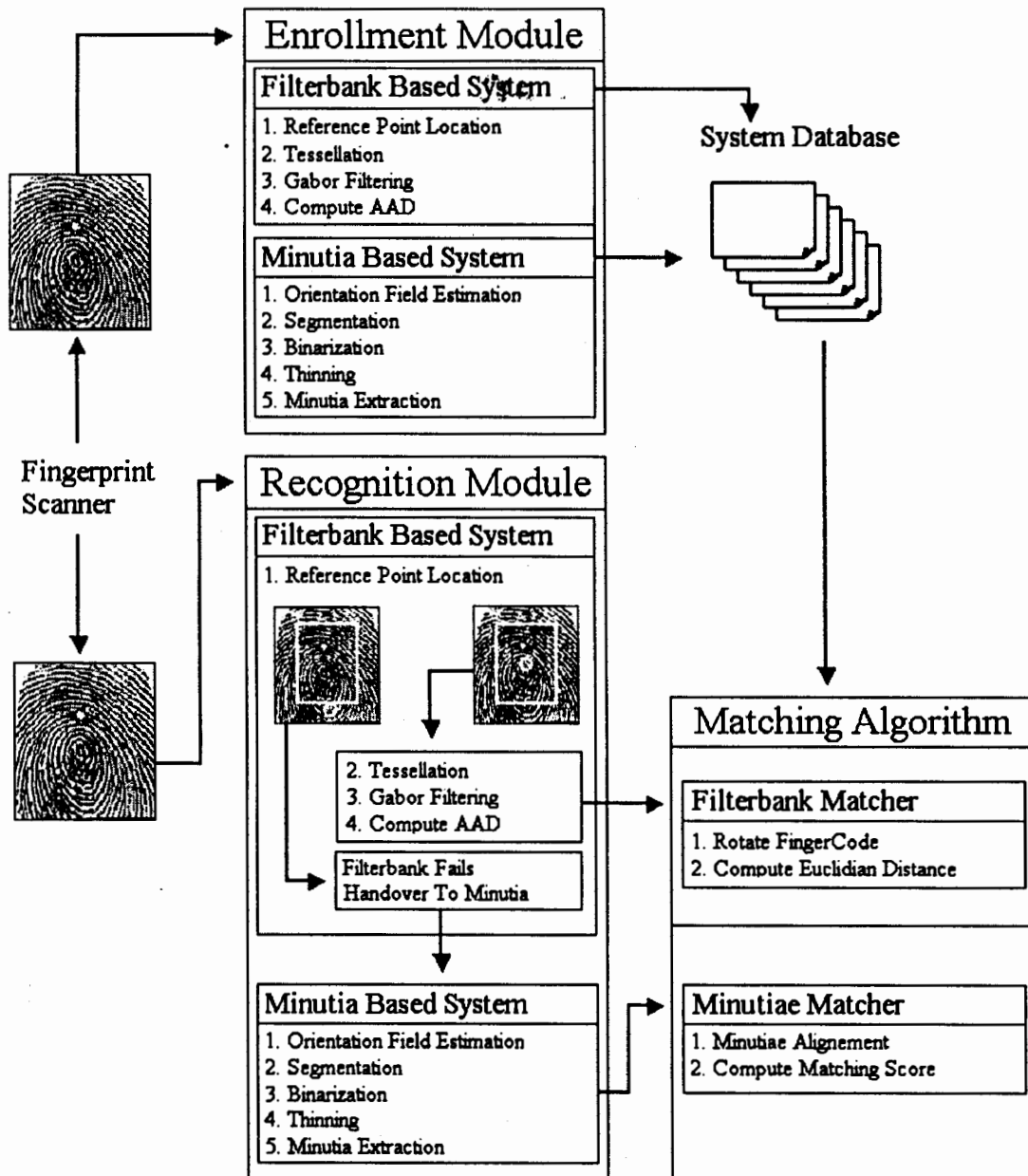


Fig. 14: Architecture of proposed Multi-Matcher based Fingerprint Identification System

Proposed Multi-Matcher based Fingerprint Identification System

The architecture of the proposed system, for multiple-matcher based fingerprint identification system, is depicted by Fig. 14. Fingerprint system based on multi matcher can be of many variations. The architecture highlights only one of many combinational or decision level integration point. For example, the proposed system relies first on one matcher but when that representation scheme seems to fail or degrade, the task is handed over to the other representation scheme. Another possibility could be that both matchers could be run in parallel and at the end the final decision could be reached upon by evaluating the result of the individual matcher.

The system put forth gives more importance to filterbank matcher than minutia. Because minutia matcher, being older and mature in terms of fingerprint identification technology, haven't got any definite failure or weak point. Its performance doesn't deteriorate due to a single factor, as is the case in filterbank-based system (i.e. reference point calculation algorithm though extremely efficient but at times it is also extremely unreliable factor). Also minutia-based system had been the focus point for fingerprint systems for a very long time now and it has evolved to its full strength and further developments in this particular direction have ceased to some extent or are very slow. Therefore alternate contemporary techniques should be given a chance for further nourishment.

This multi-matcher based system in its enrollment module extracts and stores fingerprint signature using both Filterbank and Minutia based systems accordingly. Next time when the person comes for authentication then his fingerprint signature is first extracted using Filterbank technique. Here we have made a check for reference point that if it is marked at a point (near the edge of fingerprint image), corresponding to which the system is unable to select the area for tessellation, then the system doesn't entertain the task of authentication through Filterbank technique. The authentication task is handed over to minutia system whose outcome will definitely be not as deteriorating as would have been the case, if Filterbank technique alone had been employed.

There may be certain constraints associated with these multi-matcher based systems like efficiency in terms of speed, decision level fusion when both the systems comes up with their own authentication outcomes, separate database management, etc. But these issues aren't hard to overcome and will definitely be addressed once an upper bound for the performance of multi-matcher based fingerprint identification system is established.

Conclusion

Multi- matcher based fingerprint identification systems though have been talked about a lot but haven't yet received any practical consideration until now. If appropriate efforts been utilized in this direction it may prove to be the cure for deficiencies that are present in the present day fingerprint identification systems.

Presented in this study is just one of many possible variations of multi-matcher based fingerprint systems. Further investigations will also focus on the precedence of fingerprint matchers for decision making in a multi-matcher based fingerprint identification system relative to their individual performance results obtained when operated individually.

References

1. A. K. Jain, L. Hong, S. Pankanti and R. Bolle, 1997. An Identity- Authentication System Using Fingerprints. Proc. of the IEEE, 85; 1365-1388.
2. M. Kawagoe and A. Tojo, 1984. Fingerprint Pattern Classification. Pattern Recognition, 17; 295-303.
3. L. Hong and Anil K. Jain, 1999. Classification of Fingerprint Images, 11th Scandinavian Conference on Image Analysis; Kangerlussuaq, Greenland.
4. Asker M. Bazen and Sabih H. Gerez. Segmentation of Fingerprint Images, Department of Electrical Engineering, University of Twente. Laboratory of Signals and Systems.
5. R Klette and P. Zamperoni, 1995. Handbook of Image Processing Operators. John Wiley & Sons, New York.
6. Anthony Lum, 1999. Fingerprint Recognition. Department Of Electrical Engineering, University Of Queensland, St Lucia, 4067.
7. S. Kasaei, M. Deriche, B. Boashash, 1997. Fingerprint Feature Enhancement Using Block-Direction on Reconstructed Images. International Conference on Information, Communications and Signal Processing; 721-725.
8. Anil Jain, S. Prabhakar, Lin Hong. Fingercod: A Filterbank for Fingerprint Representation and Matching. CPS Technical Report Library, MSU.
9. Anil Jain and F. Farrokhnia, 1991. Unsupervised Texture Segmentation Using Gabor Filters. Pattern Recognition, 24; 1167-1186.
10. J. Daugman, 1993. High Confidence Recognition of Persons by a Test of Statistical Independence. IEEE Trans. Pattern Anal. and Machine Intell., 15; 1148-1161.