NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION)

(AFFILIATED TO VTU, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA)

PROJECT REPORT

ON

"Fingerprint Indexing scheme using Ridge Orientation Fields"

Submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Engineering

in

Computer Science and Engineering

Submitted by:

1. BHARATH K A	1NT10CS401
2. SHRUTHI R PATIL	1NT09CS091
3. VINODA H K	1NT10CS413
4. VAISHNAVI S	1NT09CS099

Under the Guidance of:

Ms. VIDYADEVI G BIRADAR,
Asst. Professor, CSE Dept.

Department of Computer Science and Engineering

Nitte Meenakshi Institute of Technology,
Yelahanka, Bangalore– 560064

Academic Year 2012-13

NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

(AN AUTONOMOUS INSTITUTION)

(AFFILIATED TO VTU, BELGAUM, APPROVED BY AICTE & GOVT.OF KARNATAKA)

CERTIFICATE

Certified that the project work entitled "**Fingerprint Indexing scheme using Ridge Orientation Fields"** carried out by **BHARATH.K.A(1NT10CS401), SHRUTHI R PATIL (1NT09CS091), VINODA.H.K.(1NT10CS413) and VAISHNAVI.S (1NT09CS099),** bonafide students of NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY in partial fulfillment for the award of Bachelor of Engineering of VISVESVARAYA TECHNOLOGICAL UNIVERSITY, Belgaum during the year 2012-2013. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the department library. The project report has been approved as it satisfies the academic requirement in respect of the project work prescribed as per the autonomous scheme of NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY for the said Degree.

Name & signature of the Guide	Name & signature of the HOD	Name & Signature of the Principal
Ms. Vidyadevi G Biradar	Dr. Nalini N	Dr. H C Nagaraj

External Viva

Name of Examiners	Signature with date
1. _____	
2. _____	

ACKNOWLEDGEMENT

The immense pleasure and joy one drives on completion of Project is beyond description. It's the duty of the concerned person to pay their respect and acknowledge the advice, guidance and assistance received from all quarters of such an accomplishment.

First and foremost we wish to thank our ***Principal Dr. H C Nagaraj****, for encouraging us to complete this project phase.*

With deep sense of regards and gratefulness, we express our sincere thanks to our ***H.O.D Dr. Nalini N****, for her suggestion at every stage of completion of this project.*

We are extremely grateful to our project guide ***Ms. Vidyadevi G Biradar*** *for the support and guidance that she has provided throughout the course of this project.*

We would like to express our heartfelt gratitude to all ***Teaching*** *and* ***Non-Teaching*** *Staff who were a great support during our troubled times.*

Finally we thank our ***parents*** *and* ***friends*** *who took such keen interest in our work and helped us to overcome our difficulties during the completion of this Project.*

ABSTRACT

Fingerprints are one of many forms of biometrics used to identify individuals and verify their identity. Because of their uniqueness and consistency over time, they have been used for identification over centuries .Fingerprint indexing is an efficient technique that greatly improves the performance of Automated Fingerprint Identification Systems.

Orientation fields can be used to describe interleaved ridge and valley patterns of fingerprint image, providing features useful for fingerprint recognition. We employ moments for extraction of fingerprint representation. Moments are capable of describing fingerprint ridge flow structures. This feature vector gives a compact representation that is important for robust fingerprint indexing. A fingerprint indexing scheme is employed to facilitate efficient and effective classification of the fingerprints. Given a query fingerprint, the purpose of fingerprint indexing is to reduce the number of candidates for fine-matching with a desired accuracy and facilitate the complex search of fingerprints in Automatic Fingerprint Identification System (AFIS).

Table of Contents

CHAPTER 1

INTRODUCTION

CHAPTER 1

1. INTRODUCTION

1.1. Problem Statement

"To design a Fingerprint Indexing System using pose invariant representation of ridge orientation fields that can be used in Automated Fingerprint Identification System."

In this era of technologies authentication of a person is an important task for any organization. Biometrics has always been an effective solution for this problem due to its uniqueness and ease of use. This is the reason we prefer Biometrics systems like fingerprint recognition system though it is ancient method. With the growth of the human population, there is a tremendous increase in the number of users using such system. Therefore, the database of such system grows exponentially in size, in this condition the performance of such systems detoriate. In this project, we aim to improve the performance of such system through database indexing by clustering methods. To solve this problem, we propose to implement following steps.

- Take a sample fingerprint database. As a first step enhance the quality of the fingerprint images using noise removal and histogram equalization techniques.
- Pre-process all the fingerprint images.
- Extract the features from the fingerprint by calculating the polynomials using Legendre moments.
- Cluster the fingerprints according to the moments calculated by referring the Reference images stored in Reference Database using K-means Clustering.

1.2. Introduction

In a fingerprint identification system, once a user provides fingerprint image as an input, the system has to search a database to identify the corresponding one without a claimed identity. However, the size of very large databases seriously compromises the efficiency of the system; this can be alleviated by the techniques of classification and indexing which reduce the search space. Classification techniques attempt to classify fingerprints into exclusive classes. In exclusive classification, fingerprints are assigned to one of the human-predefined classes and the query fingerprint is matched with those belonging to the same class. The fingerprint indexing is challenging due to ambiguous fingerprints whose class membership cannot be exclusively stated even by human experts. These problems limit the capability of fingerprint classification to narrow down the search of large database. Hence there is a need of an efficient indexing method. We propose fingerprint indexing method that uses ridge orientation fields of the fingerprint.

In this project, we proposed a Fingerprint Indexing System using orientation fields which are represented through Legendre moment, these are pose invariant. The local ridge orientation field is a rich information resource for fingerprint image processing and feature extraction. Moments are capable of describing fingerprint orientation patterns including singular regions and restoring spurious orientations in noisy fingerprints. Unlike most indexing schemes using the raw orientation data, a set of rotation moment invariants are derived from moments to form a compact feature vector. Clustering of fingerprint images is carried out using the feature vector.

Any image processing application involves preprocessing, feature extraction and classification that hold for the application under consideration. This project is implemented in Matlab environment.

1.3. Biometrics

Biometrics, which refers to identifying an individual based on his or her physiological or behavioral characteristics, and has the capability to reliably distinguish between an authorized person and an imposter. Since biometric characteristics are distinctive, cannot be forgotten or lost, and the person to be authenticated needs to be physically present at the point of identification. Biometrics is inherently more reliable and more capable than traditional knowledge-based and token-based techniques. Biometrics also has a number of disadvantages. For example, if a password or an ID card is compromised, it can be easily replaced, but biometrics cannot be compromised and hence it cannot be replaced. Similarly, users can have a different password for each account, thus if the password for one account is compromised, the others accounts are still safe. However, if biometrics is compromised, all biometrics-based account can be broken-in. Among all biometrics (e.g., face, fingerprints, hand geometry, iris, retina, signature, voice print, facial thermo gram, hand vein, gait, ear, odor, keystroke dynamics, etc.), fingerprint-based identification is one of the most mature and proven technique.

1.4. Comparison of various Biometrics

Table 1.1 comparison of different biometrics based on their properties

Biometric characteristic	Universality	Unicity	Persistence	Colectability	Performance	Acceptability	Circumvention
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
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	medium	low	low	medium	low	high	low
Thermogram	high	high	low	high	medium	high	high

Advantages of fingerprints over other form of biometrics

- Universality, which means that each person has a fingerprint.
- Distinctiveness, which indicates that any two persons should be sufficiently different in terms of their fingerprints.
- Permanence, which means that the fingerprint should be sufficiently invariant (with respect to the matching criterion) over a period of time.
- Collectability, which indicates that the fingerprint can be measured quantitatively.

However, in a practical fingerprint recognition system, there are a number of other issues that should be considered, including:

- Performance, which refers to the achievable recognition accuracy, speed, robustness, the resource requirements to achieve the desired recognition accuracy and speed, as well as operational or environmental factors that affects that recognition accuracy and speed.
- Circumvention, which reflects how easy it is to fool the system by fraudulent methods.

A practical fingerprint recognition system should have acceptable recognition accuracy and speed with reasonable resource requirements, harmless to the users, accepted by the intended population, and sufficiently robust to various fraudulent methods.

A number of biometric identifiers are in use in various applications. Each biometric has its strength and weaknesses and the choice typically depends on the application. No single biometric is expected to effectively meet the requirement of all the applications. The match between a biometric and an application is determined depending upon the characteristics of the application and the properties of the biometrics.

1.5. Biometrics Applications

Biometrics has been widely used in forensics applications such as criminal identification and prison security. The biometric technology is rapidly evolving and has a very strong potential to be widely adopted in civilian applications such as electronic banking, e-commerce, and access control. Due to the rapid increase in the number and use of electronic transactions, electronic banking and electronic commerce are becoming one of the most important emerging applications of biometrics.

These applications include credit card and smart card security, ATM security, cheque cashing and fund transfers, online transactions and web access. The physical access control application have traditionally used token-based authentication. With the progress in biometric technology, these applications will be increasingly used in biometrics for authentication.

Remote login and data access applications have traditionally used knowledge-based authentication. These applications have already started using biometrics for personnel authentication. The term biometrics will become more widespread with the growth in technology and it becomes more trust worthy. Other biometrics applications include welfare disbursement, immigration checkpoints, national ID, voter and driver registration, and time and attendance.

A recognition/identification system compares the person's fingerprint with the records that are earlier enrolled and stored in the database. Features extracted from fingerprint images are stored in the database and image retrieval is done by specifying these features in the retrieval query.

A fingerprint recognition system involves several steps: fingerprint image acquisition, Preprocessing, Feature extraction, indexing and classification. Fingerprint recognition system uses patterns across fingerprints, which are called ridges and the spaces between ridges are valleys. Fingerprints are often compared through their ridge features such as minutiae, orientation field or sweat pores.

Block diagram of fingerprint recognition/identification system

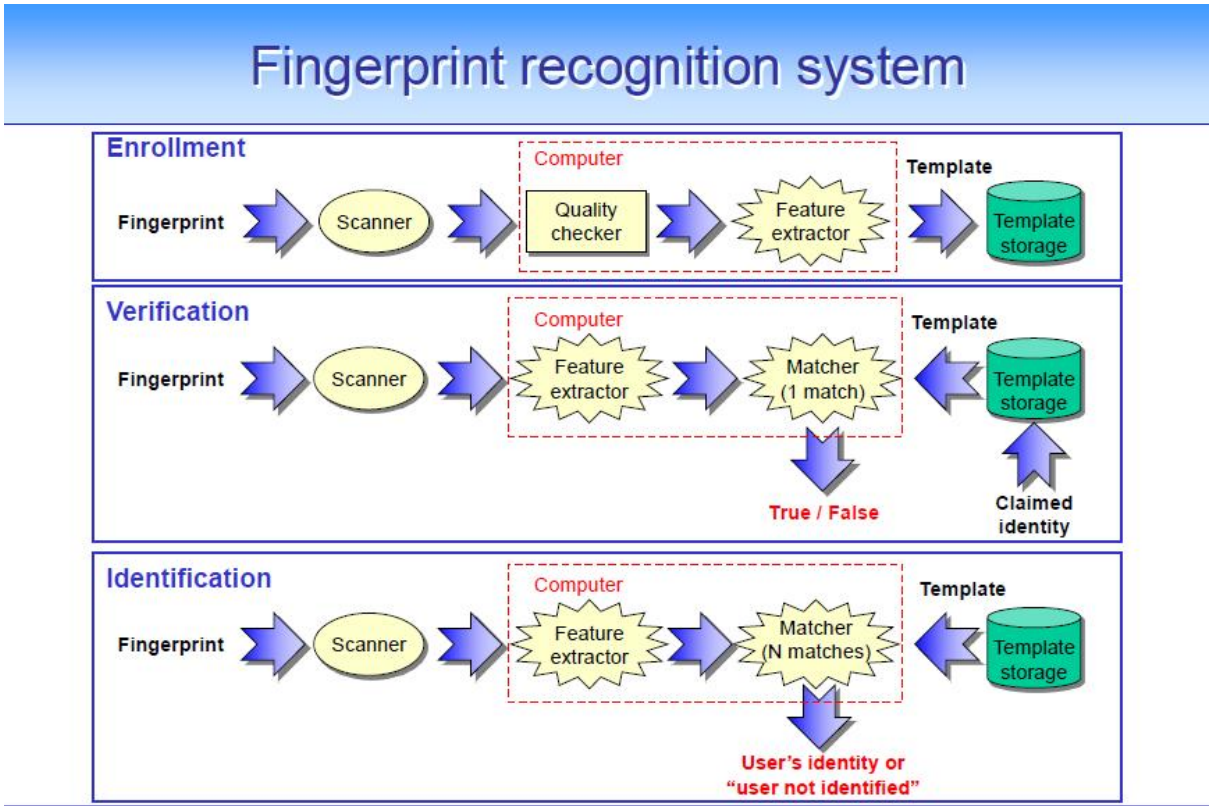

Figure 1.1 Block diagram

1.6. Fingerprints

Fingerprints as a kind of human biometrics have been widely used for personnel identification in the commercial and forensic areas because of its uniqueness and immutability. Fingerprints are composed of interleaved parallel ridge and valley flows. Efforts have been made to extract representative features from friction ridge details to determine the uniqueness of finger- prints for developing automatic fingerprint identification system (AFIS).

Figure 1.2 Showing the ridges and valley flows in a fingerprint

The following are the most important features of the fingerprints

- Every human individual has unique and never repeated fingerprints (even identical twins have different fingerprints).
- They are persistent throughout life except for permanents scarring; they vary within limits which allow for classification.
- They have some local discontinuities in their patterns; there are more than eight discontinuities but the two most prominent structures used are ridges and valleys.

A fingerprint is made of a number of ridges and valleys on the surface of the finger. Ridges are the upper skin layer segments of the finger and valleys are the lower segments. The ridges form so-called minutia points: ridge endings (where a ridge end) and ridge bifurcations (where a ridge splits in two). Many types of minutiae exist, including dots (very small ridges), islands (ridges slightly longer than dots, occupying a middle space between two temporarily divergent ridges), ponds or lakes (empty spaces between two temporarily divergent ridges), spurs (a notch protruding from a ridge), bridges (small ridges joining two longer adjacent ridges), and crossovers (two ridges which cross each other).

The uniqueness of a fingerprint can be determined by the pattern of ridges and furrows as well as the minutiae points. There are five basic fingerprint patterns: arch, tented

arch, left loop, right loop and whorl. Loops make up 60% of all fingerprints, whorls account for 30%, and arches for 10%.

Fingerprints are usually considered to be unique, with no two fingers having the exact same dermal ridge characteristics.

Figure 1.3 Basic and composite characteristics of Ridges

Three classes of fingerprints are loops, whorls, and arches that are described as follows:

1. **Loops:** 60-65% of the population has loops.

- It has one or more ridges entering from one side of the print, curving and exiting from the same side.
- Loop opening toward little finger is called as ulnar loop; Loop opening toward thumb is called as radial loop
- Type lines are ridges that diverge (separate).
- Deltas are located at the point of divergence. All loops must have one delta

2. Whorls:

- 30-35% of the human population has whorls
- All whorl patterns must have type lines and **two** deltas
- Four major types: plain, central pocket, double loop, accidental
- Plain whorls must have at least one ridge that makes a complete circuit, and an imaginary line from one delta to the other must touch a whorl ridge
- Central pocket whorls must have at least one ridge that makes a complete circuit, and an imaginary line from one delta to the other cannot touch a whorl ridge
- Double loop is two loops combined to make one whorl
- Any other types not in the three categories are called accidentals (generally, they have a whorl type pattern, which is why they are in the whorl grouping)

3. Arches:

- Only 5 percent of the human population has arches
- Arch ridges tend to enter from one side of the print and leave out the other side
- Two distinct types-- plain arches and tented arches, Plain arches tend to show a wave like pattern
- Tented arches show a sharp spike at the center of the arch, Arches do not have type line, deltas or cores.

Using the information above, here are some examples of each of the major fingerprint types:

Figure 1.4 The types of fingerprints

Applications of fingerprint biometrics:

Fingerprint sensors are best for devices such as cell phones, USB flash drives, notebook computers and other applications where price, size, cost and low power are key requirements. Fingerprint biometric systems are also used for law enforcement, background searches to screen job applicants, healthcare and welfare.

1.7. Fingerprint Indexing

Given a query fingerprint, the goal of indexing is to identify and retrieve a set of candidate fingerprints from a large database in order to determine a possible match. This significantly improves the response time of fingerprint recognition systems operating in the identification mode. Fingerprint indexing is a very important phase in fingerprint matching. Since fingerprints are highly reliable biometrics and widely used, it would be very desirable

to use them in identification applications. Indexing will allow fast matching of the query fingerprint against a large set of enrolled fingerprints. So far this is an unsolved problem, and matching query fingerprint against a database of N enrolled fingerprints requires N separate matching's of two fingerprints. Existing approaches can only reduce the matching set to some predefined fraction of the whole fingerprint set.

In a fingerprint identification system, a person is identified only by his fingerprint. To accomplish this, a database is searched by matching all entries to the given fingerprint. However, the maximum size of the database is limited, since each match takes some amount of time and has a small probability of error. A solution to this problem is to reduce the number of fingerprints that have to be matched. This is achieved by extracting features from the fingerprints and first matching the fingerprints that have the smallest feature distance to the query fingerprint. Using this indexing method, modern systems are able to search databases up to a few hundred fingerprints.

Several problems can be defined in the context of fingerprint recognition, in consideration with verification, identification and classification. The term recognition is used as a general term that encompasses all three kinds of tasks. Verification (or authentication) systems use a fingerprint to verify that a person really is the person who he or she claims to be. So, these systems receive two inputs: the claimed identity of the person requesting authentication and a fingerprint. The identity is used to retrieve a reference fingerprint stored in a database, which is matched (compared) to the currently o□ered fingerprint (the test fingerprint). This results in a measure of similarity on which the decision is based.

Identification systems on the other hand, only receive one input, a fingerprint. This might for instance be used to check whether new users are already known by the system with another name, or to identify non-cooperative users. In this case, the system has to search a database for a matching fingerprint. This task is also referred to as one-to-many matching. If a matching fingerprint is found, this identifies the person.

1.8. Purpose

A Fingerprint Recognition scheme is a computer application for automatically identifying or verifying a person from a digital image. One way to do this is by comparing selected fingerprint image features from the query fingerprint image and the fingerprint database. In a large scale identification system, the goal is to reduce the number of candidates to be considered by the identification algorithm.

1.9. Motivation

Large volumes of fingerprints are collected and stored every day in forensics and government applications. Automatic fingerprint identification requires matching the query fingerprint against a large database of templates (features extracted from the fingerprints and used for comparison). This operation can be very time consuming; therefore, it is desirable to reduce the number of database fingerprints to be considered, by using pre-filtering techniques that quickly select the most similar candidates. Designing fingerprint pre-filtering techniques poses challenging problems in terms of both accuracy and efficiency: assessing the current state of the art in fingerprint indexing is important for understanding its limitations and addressing future research.

To provide an efficient indexing scheme that reduces the search time involved in the Automated Fingerprint Recognition System by representing the fingerprints using local ridge orientation and extracting the features from the fingerprint using the Legendre Moments.

1.10. Literature Survey

The practice of using fingerprints as method of identifying individuals has been in use since the late nineteenth century when Sir Francis Galton defined some of the points or characteristics from which fingerprint can be identified. Fingerprint identification began its transition to automation in the late 1960s along with the emergence of computing

technologies. In 1969 there was a major push from the FBI to develop a system to automate its fingerprint identification system.

The determination and commitment of the fingerprint industry, government evaluation and needs, and organized standards bodies led to the next generation of fingerprint recognition, which promises faster and higher quality acquisition devices to produce higher accuracy and more reliability. The fingerprint Indexing methods are employed to reduce the search space involved in the Fingerprint Recognition Systems.

In general, fingerprint features can be categorized into three different levels. Level1 features capture the macro details of fingerprint such as the ridge flow structures, pattern types, orientation field and singular points (i.e., core and delta points). These coarse level features are not adequate to determine the uniqueness of a fingerprint. Level2 features capture the minute details of ridge flow patterns such as the ridge bifurcations and endings (i.e., minutiae points). Theoretical studies over a large number of fingerprint samples show that these feature scan provide a huge amount of discriminatory information to deter mine the individuality of a fingerprint. Level 3 features include all dimensional attributes of ridge details such as ridge pores, edge contours, incipient ridges, creases, scars and etc. Extraction of these fine and detailed features usually requires fingerprint images withmorethan500-ppi resolution. Currently, most commercial AFIS work on 500-ppi image so their recognition algorithms primarily use level1 and 2 features. In general ,high level (levels2 and 3) features are often used to provide more discriminatory information to estimate the individuality of fingerprint .But reliable extraction of these features requires higher computation cost and is easily affected by image quality and resolution. Although level 1 features such as those extracted from orientation field do not contain enough in formation for uniquely characterizing a fingerprint, they are more robust to image noise and play important role for extraction and description of high level features such as minutiae points .All these features are believed to be distinctive and useful, and can be incorporated into AFIS to determine the individuality of finger print. Level 1 features are often used for classification (categorizing fingerprints into major pattern types such as arch, loop and whorl) and indexing of fingerprint database .The outcome of classification and indexing provides a list of the most likely fingerprint candidates which would be further considered in the fine level matching using

higher level features. This multi-level matching scheme can improve the efficiency and accuracy of large scale fingerprint identification.

The first fingerprint classification rules were proposed in 1823 by Purkinje, who classified fingerprints into nine categories according to global ridge configurations. The first in depth study on fingerprint classification was made by Francis Galton, who divided fingerprints into three major classes and further divide each category into subcategories. Few years later, Edward Henry refined Galton's classification by increasing the number of classes which was widely used. The five most common classes were arch, tented arch, left loop, right loop and whorl.

The fingerprint orientation image was first introduced by Grasseli, is a matrix D whose elements encode the local orientation of the fingerprint ridges. Each element θ_{ij} ,corresponding to the node [i, j] of a square meshed grid located over the pixel $[x_i, y_j]$ denotes the average orientation of the fingerprint ridges in the neighborhood of the pixel $[x_i , y_j]$.in this approach The simplest and most natural approach for extracting local ridge orientation is based on computation of gradients in the fingerprint image this method although simple and efficient ,has some drawbacks .the ridge orientation estimate is generally very sensitive to noise and Prewitt or sobel convolution masks presents problems due to the non-linearity and discontinuity .

Orthogonal Legendre moments were introduced by Teague. These moments are used in a wide range of applications. Teh and Chain reported that, orthogonal Legendre moments can be used to represent an image with minimum amount of information redundancy.

Computer vision and pattern recognition applications usually required the computation of invariant features. These features can be obtained through Legendre moment invariants.

The invariance of Legendre moments could be achieved mainly through two methods. These methods are direct and indirect. In the direct method, Legendre moment invariants are computed directly by using Legendre polynomials. The first drawback of this method is the high computational demands especially with big size images and the higher order moments. A refined method is proposed to overcome this problem. The other drawback is the

impossibility of deriving rotational invariance property. On the other side, the indirect method easily achieves all kinds of invariance including the rotation invariance and is highly effective when the computational complexity is tremendously reduced.

CHAPTER 2

SOFTWARE AND HARDWARE REQUIREMENTS

CHAPTER 2

2. SOFTWARE AND HARDWARE REQUIREMENTS

2.1. Software Requirement Specification

The Software requirement specification phase is an important phase in the software development cycle and it is a complete description of the behavior of the system to be developed. The purpose of the Software Requirement Specification (SRS) document is to specify the user goals and tasks that need to be achieved. It must also include detailed description of the context and requirements that include both functional and non-functional which are vital to the successful completion of the project. Apart from these, the SRS document incorporates itself in the constraints and assumptions made during the course of the project. Requirements must be measurable, testable, related to the identified needs or opportunities, and defined to a level of detailed sufficient for system design.

This section presents the following topics:

- Software and hardware requirements
- Functional Requirements
- Non Functional Requirements

2.2. Overall Description

In this section, we describe representation of fingerprint by ridge orientation field which is represented using the Legendre Moments. The moments obtained by this process is used for achieving the main objective that is fingerprint indexing to reduce the number of candidates for fine-matching with a desired accuracy.

The next section will explain about the product perspective, product functions, assumptions and dependencies, User characteristics.

2.2.1. Product Perspective

The software must run on a Windows based Operating System. It should be responsible for the indexing fingerprint database into clusters. A query fingerprint to be searched is then compared only to the fingerprints in a single bin (few cluster) of the fingerprint database, thereby reducing the search space. The product should be made flexible enough to provide good GUI interfaces and different modes of operations which help to analyze the product development step by step.

2.2.2. Product Functions

The product takes the query image and searches that in fingerprint database it reports the search results, if there is an exact match it says found otherwise it gives nearest matching fingerprints.

2.2.3. Assumptions and dependencies

- The tool should use windows platform and the IDE used is Matlab.
- Grey scale fingerprint images are considered as input.
- Image can have moderate noise but not latent.

2.2.4. User characteristics

The end user

- Be familiar with the type of images to be used.
- Have basic knowledge of biometrics.
- Be aware with standard terminologies.
- Needs to know limitations of the system.

- Should know how to manually match fingerprints.
- Should have training on how to use the system.

2.3. Specific requirements

The hardware and software requirements that are feasible for the project are provided in this section.

2.3.1. Hardware Requirements

- Intel Core2Duo Processor
- 2GB RAM
- Minimum 10GB hard disk space

2.3.2. Software Requirements

- IDE : Matlab R2012a
- Host OS : Windows Platform [XP/7/8]

2.3.3. Functional Requirements

The functional requirements of the proposed project is

1. Read the image and perform preprocessing so that image is ready for feature extraction.
2. Extract the desired features used for fingerprint indexing; they are ridge orientation estimation and method to represent ridge orientation.
3. Estimate Legendre moments to represent ridge orientation.
4. Perform clustering with the feature vector
5. Take the test image and identify which cluster it belongs to and also show the Corresponding match.

2.3.4. Non-Functional Requirements

The non-functional requirements are as follows:

- Ease of use: The interface should be user-friendly. The usage of fields in GUI should be easy, simple, and intuitive.
- Performance: The performance is determined by the time consumed by the product to search a query fingerprint and report the result.
- Reusability: The modules developed should be reusable wherever applicable.
- The code developed should work with varieties of fingerprint databases.

CHAPTER 3

HIGH LEVEL DESIGN

CHAPTER 3

3. HIGH LEVEL DESIGN

3.1. Purpose

The purpose of the High Level Design (HLD) Document is to add necessary details to the current project description to represent a suitable model for coding. This document also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

3.2. Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD used non-technical to mildly-technical terms which should be understandable to the administrators of the system.

3.3. Design considerations

There are many aspects to consider in the design of a piece of software. The importance of each should reflect the goals the software is trying to achieve. Some of these aspects are:

- **Compatibility** - The software is able to operate with other products that are designed for interoperability with another product. For example, a piece of software may be backward-compatible with an older version of itself.

- **Extensibility** - New capabilities can be added to the software without major changes to the underlying architecture.
- **Fault-tolerance** - The software is resistant to and able to recover from component failure.
- **Maintainability** - A measure of how easily bug fixes or functional modifications can be accomplished. High maintainability can be the product of modularity and extensibility.
- **Modularity** - the resulting software comprises well defined, independent components. That leads to better maintainability. The components could be then implemented and tested in isolation before being integrated to form a desired software system. This allows division of work in a software development project.
- **Reliability** - The software is able to perform a required function under stated conditions for a specified period of time.
- **Reusability** - the software is able to add further features and modification with slight or no modification.
- **Robustness** - The software is able to operate under stress or tolerate unpredictable or invalid input. For example, it can be designed with resilience to low memory conditions.
- **Security** - The software is able to withstand hostile acts and influences.
- **Usability** - The software user interface must be usable for its target user/audience. Default values for the parameters must be chosen so that they are a good choice for the majority of the users.

3.3.1. Assumptions and dependencies

- The tool shall use windows platform.
- Input images are Grey scale images.
- Code works in matlab environment.
- Fingerprint images are of moderate quality.
- For performance factors database is assumed to be of moderate size.

3.4. Development Strategies

Iterative development is an approach to software development which centers on the idea of moving development cyclically, rather than trying to do everything all at once. This method is not suitable for all types of software but it can be highly beneficial and very useful in some settings. It is important to note that a common misconception about iterative development is that it is unplanned or spontaneous. This is not, in fact, the case. It is actually highly structured.

3.5. General Constraints

The Development of a tool assisting the fingerprint indexing scheme must be user friendly and as automated as possible. Administrators should not be required to do anything besides the initial setup and image selection in automated mode, and the users needn't have to understand its internal working.

3.6. Architectural Constraints

The architecture of the tool is based on several implications that are grouped into different categories and are presented in the following sections:

3.6.1. Functionality

The tool should be capable of clustering the fingerprints in the database and for testing purpose the tool should be able to search the query fingerprint in the clusters and display matched and nearby matched fingerprints.

3.6.2. Interfaces

This section describes the interfaces for the tool. They are as follows:

3.6.2.1. User Interfaces

The tool provides its user with the GUI interface. The GUI page is as follows in the figure 3.1.

3.6.2.2. Software Interfaces

GUI interfaces with Matlab programming constructs.

3.6.2.3. Hardware Interfaces

The tool shall interact with the kernel, which internally interfaces with Intel x64 processor and other essential Hardware used.

3.6.2.4. Usage Environment

This product works on windows based operating system with pre-installed R2012a or higher version of Matlab.

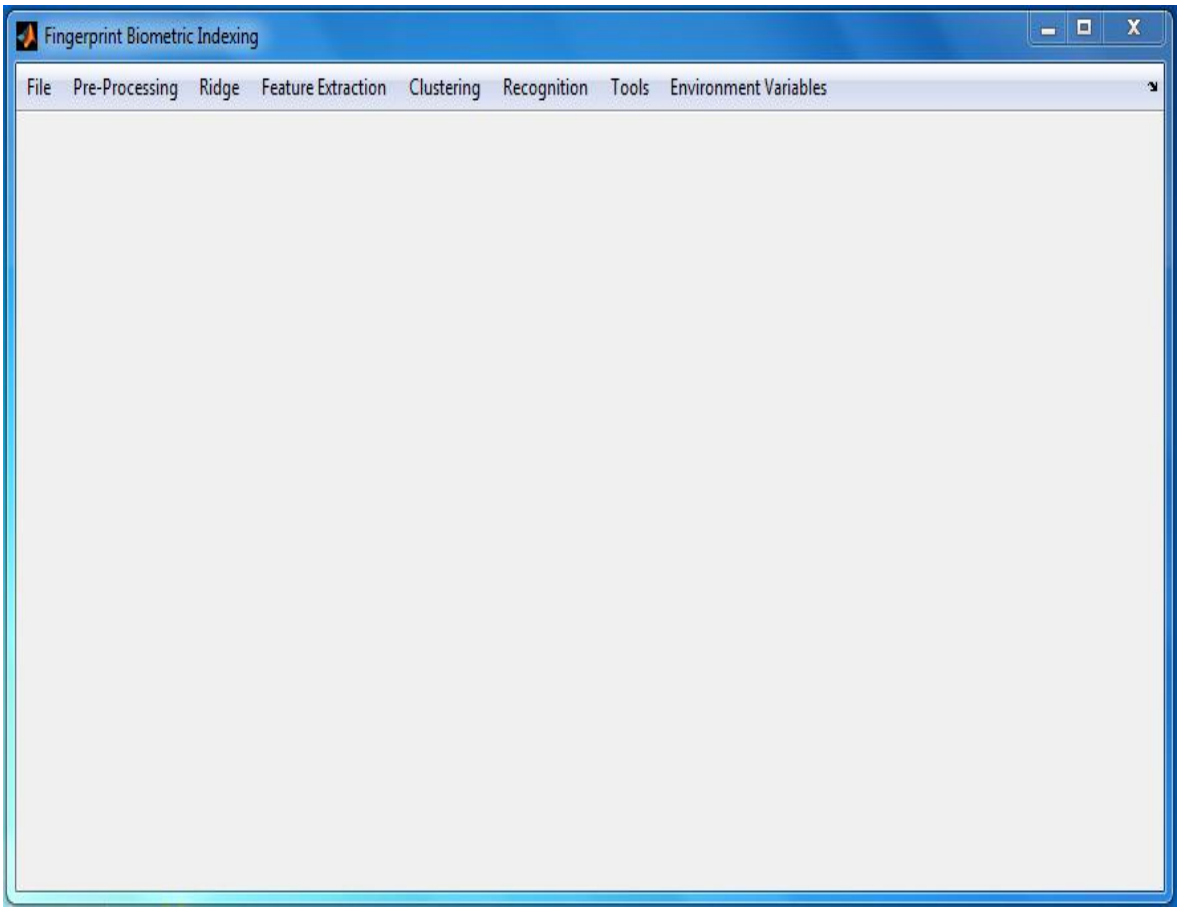

Figure 3.1 Main menu of Fingerprint Indexing

3.6.3. Memory management policies

Runtime Memory Management is automatically taken care by Matlab memory management system. Automatic management is one of the core feature provided by Matlab.

3.7. System Architecture

3.7.1. System Architecture Diagram

The following block diagram shows the detailed system architecture diagram of Fingerprint Indexing Scheme.

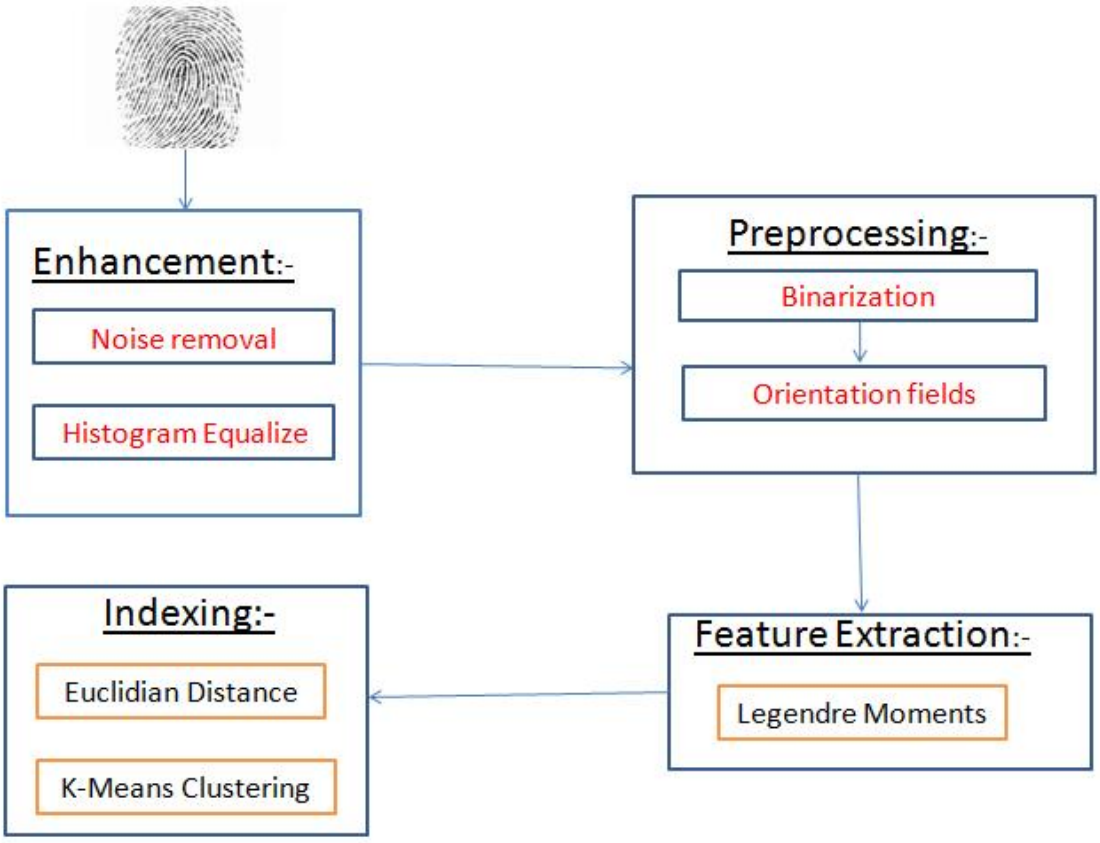

Figure 3.2 System Architecture Diagram

3.7.2. Dataflow Diagram

The dataflow diagram used in the Fingerprint Indexing Mechanism is as follows:

- The first step is to choose an input image and to enhance the quality of the image by equally distributing the intensities of the image thereby to increase the contrast of an image through Histogram Equalization.
- The local ridge orientation of the fingerprint is located. This is the pre-processing step of the fingerprint indexing scheme.
- Calculate the moments of the fingerprint by using Legendre moments.
- Cluster the fingerprint images using the moments of the reference images and the images in the fingerprint database using K-Means Clustering technique.
- Thereby all the fingerprints in the database all clustered in to 5 clusters which helps for the further processing steps in the Fingerprint Recognition System.

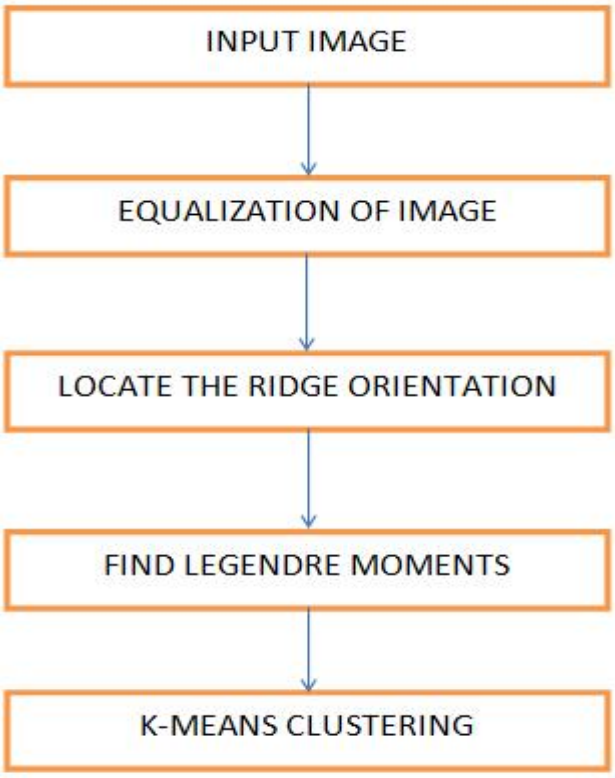

Figure 3.3 Dataflow Diagram

CHAPTER 4

DETAILED DESIGN

CHAPTER 4

4. DETAILED DESIGN

4.1. Purpose

The purpose of the Detailed Design Document (DDD) is to define the detailed design for all components of the system, which are specified in the Software Requirement Document (SRD) and Architectural Design Document (ADD). The low level components are designed, coded and tested.

4.2. Stage 1 – Fingerprint Pre-processing

The performance of a fingerprint processing algorithms depends heavily on the quality of the input fingerprint images. It is very important to acquiring good quality images but in the practice a significant percentage of required image are of poor quality due to some environmental factors or user's fingerprint condition. Hence it is essential for pre-processing of the fingerprint through image enhancement techniques.

4.2.1 Image Enhancement

Fingerprint image enhancement is used to make image clear for better use which is very easy to handle and can operate easily for further processing of the fingerprint image. The fingerprint image is usually affected by noise. Because our finger are often comes in contact with the most of the manual task we perform like fingertips becomes dirty, cut, scarred, creased, dry, wet, warm, etc. the image enhancement step is basically designed to reduce this noise and to enhance the definition of the ridges against valleys.

Histogram Equalization

Histogram Equalization has been used to do image enhancement. It is to expand the pixel value distribution of an image so as to increase the perceptional information for further processing of the fingerprint image. It is a method adopted to enhance the quality of the input fingerprint image. Histogram represents the relative frequencies of the various type of gray levels in an image. Histogram Equalization is a method that increases the contrast of an image by increasing the dynamic range of intensity given to pixels with the most probable intensity values and it is much less expensive when compared to the other method.

The figure shows the image after histogram equalization.

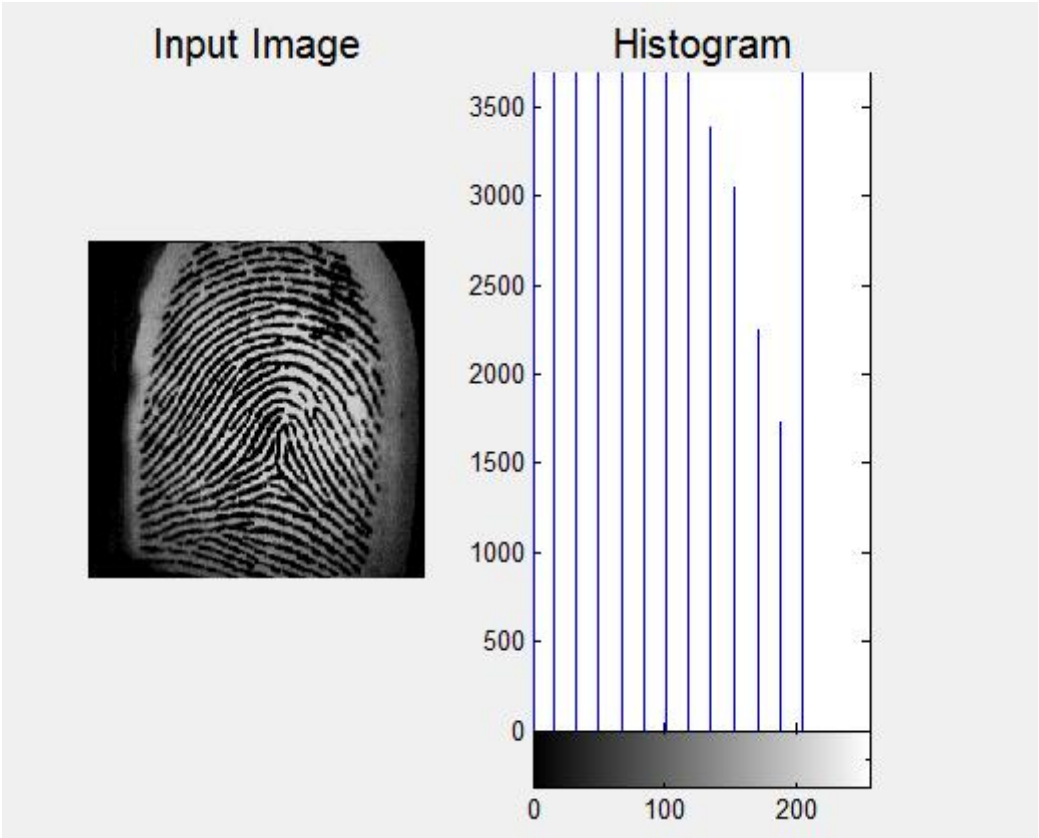

Figure 4.1 The plot of intensities of an image after Histogram Equalization

Figure 4.2 Histogram Equalization function applied to an input image

Noise Removal

Digital images are prone to different types of noise. Noise is the result of errors in the image acquisition process that result in pixel values that do not reflect the true intensities of the real scene. There are several ways that noise can be introduced into an image, depending on how the image is created. For example:

- If the image is scanned from a photograph made on film, the film grain is a source of noise. Noise can also be the result of damage to the film, or be introduced by the scanner itself.
- If the image is acquired directly in a digital format, the mechanism for gathering the data (such as a CCD detector) can introduce noise.
- Electronic transmission of image data can introduce noise.

Biometric Identification and Authentication Systems are in need of a quality image in order to aim at a reliable and accurate result. The quality of the fingerprint is obtained by the noise free images. To get a noise-free fingerprint image, they are subjected to pre-processing and filtering tasks. Especially the Fingerprint Recognition system demands accuracy factor. Hence the fingerprint images are processes to eliminate noise that helps for the accurate processing of the fingerprint images in the Fingerprint Recognition System.

Binarization

Fingerprint image Binarization is used to transform the 8-bit gray scale fingerprint image to a 1-bit binary image and here the values for ridges is 0 where as it is 1 for the furrows. After this operation, the ridges in the fingerprint will be highlighted with black color while furrows will be in white color. In this stage the gray-scale image converts into a binary image. A binary image can be processed faster than a gray-scale image. The following figure shows the image after applying Binarization technique.

Figure 4.3 Left – Input image, Right – Binarized image

4.3. Estimation of Local Ridge Orientation

Orientation fields can be used to describe interleaved ridge and valley patterns of fingerprint image, providing features useful for fingerprint recognition. Let [x, y] be a generic pixel in a fingerprint image. The local ridge orientation at [x, y] is the angle θ_{xy} that the fingerprint ridges, crossing through an arbitrary small neighborhood centered at [x, y], form with the horizontal axis. Because fingerprint ridges are not directed, θ_{xy} is an oriented direction lying in [0….180 degrees].

Thus Local ridge orientation is the angle Θ_{xy} at which the ridge crosses through an arbitrary neighborhood centered at one pixel, measured with respect to the horizontal axis. The orientation field is a discrete matrix composed of local ridge orientations estimated at discrete positions of fingerprint as shown in the figure 4.6, and thus it describes the structure of interleaved ridge and valley flows. In addition, orientation field is loosely co-related with the minutiae features that are often used in fine matching. This can reduce representation redundancy between indexing database and fine matching in an AFIS. Thus, orientation field is widely used as the main feature for fingerprint classification and indexing.

One of the most evident structural characteristic of a fingerprint is a pattern of ridges. When ridge lines takes distinctive shapes, a fingerprint pattern such as loop, delta or whorl is seen. The orientation of such ridges is taken as important feature in this project. We use the gradient based methods where the gradient vectors are compared by taking the partial derivatives of gray values at each pixel of the fingerprint and the orientation is estimated by averaging the squared gradients in local neighborhood. Although local ridge orientations have been widely employed for fingerprint representation, the best approach to extract a compact and representative feature from orientation field is still a challenging problem for fingerprint indexing. There are two important limitations in most existing orientation features. Firstly, the rotation of fingerprint image is an important factor that causes performance deterioration of fingerprint indexing. One popular method to achieve rotation invariance is to align orientation fields by a reference scheme. Secondly, the intrinsically periodic property of orientation data can affect the performance of the algorithms.

A fingerprint consists of two special direction-oriented parts: ridges and valleys, where valleys are the space between ridges and vice versa. These directional patterns contain various fingerprint features including a small number of singular points such as delta and core points and randomly distributed local discontinuities called minutiae. In this project we employ a ridge orientation estimation using gradient based algorithm that can generate an orientation of ridge flows of a fingerprint image. Thus we can use the ridge oriented image for the next step of fingerprint processing which is the Feature Extraction.

The following algorithm is used for the estimation of local ridge orientation of the image using gradient method.

1. Divide the input fingerprint image into blocks of size W x W
2. Computer the gradients G_x and G_y at each pixel in each block
3. Estimate the local orientation at each pixel (i ,j) using the following equations

$$V_x(i,j) = \sum_{u=i-\frac{W}{2}}^{i+\frac{W}{2}} \sum_{v=j-\frac{W}{2}}^{j+\frac{W}{2}} 2G_x(u,v)G_y(u,v), \tag{4.1}$$

$$V_y(i,j) = \sum_{u=i-\frac{W}{2}}^{i+\frac{W}{2}} \sum_{v=j-\frac{W}{2}}^{j+\frac{W}{2}} (G_x^2(u,v) - G_y^2(u,v)), \tag{4.2}$$

$$\theta(i,j) = \frac{1}{2}tan^{-1}(\frac{V_x(i,j)}{V_y(i,j)}), \tag{4.3}$$

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

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

$$C(i,j) = \frac{1}{N}\sqrt{\sum_{(i',j')\in D} |\theta(i',j') - \theta(i,j)|^2}, \tag{4.4}$$

$$|\theta' - \theta| = \begin{cases} d & if\,(d = (\theta' - \theta + 360)\ mod\ 360) < 180, \\ d - 180 & otherwise, \end{cases} \tag{4.5}$$

Where D represents the local neighborhood around the block (i, j); 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.

5. If the consistency level (eq (4.5)) 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.

After implementing this gradient based algorithm for the local ridge orientation estimation we get the plot of orientation fields that are shown in the following figures.

Figure 4.5 Binarized image which is given as input to the estimation of local ridge orientation

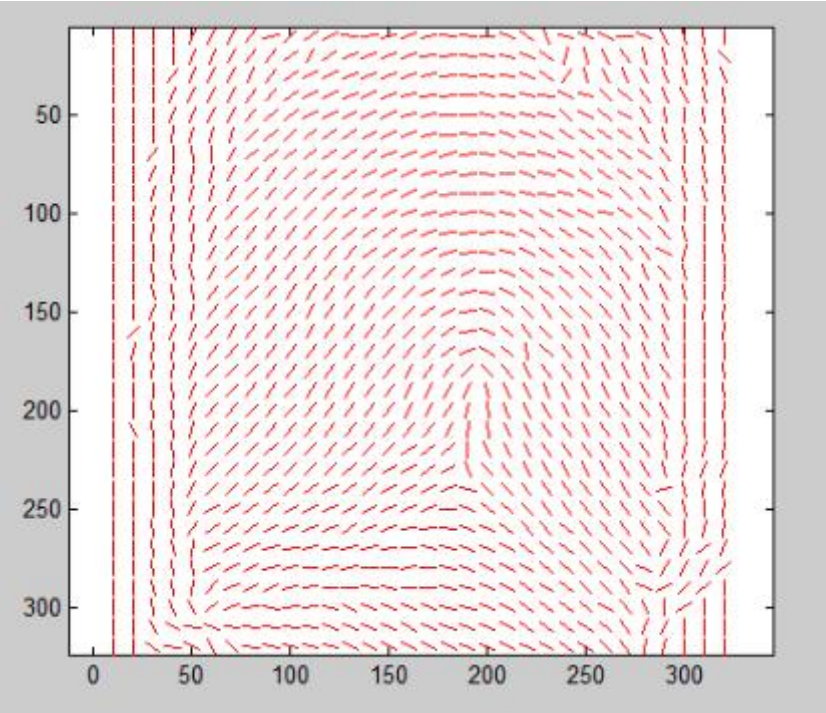

Figure 4.6: The plot of orientation

4.4. Moments

4.4.1. Introduction

Moments are powerful general tools in various fields, particularly in the field of image reconstruction and representation. It is well-known that an infinite set of mathematical moments can be derived to represent and reconstruct an image. In this project, we investigate employing a set of Legendre moments to extract the rotation-invariant representation of the orientation field. This moment-based representation is formulated to give a compact and numerical feature vector of fixed size, which is beneficial for clustering-based fingerprint indexing.

In many applications, different kinds of moments have been utilized to classify images and object shapes. Moments are important features used in recognition of different types of images. Feature extraction is an important phase in fingerprint recognition. Moment based features are very effective in fingerprint Recognition System.

Moments, which are projections of the input signal onto the polynomial function space, have found wide applications in image processing and computer vision. By the use of orthogonal Legendre polynomial, the calculation can be reduced, the error is easier to estimate, and the reconstruction can be simpler also. In this project, we characterize images by Legendre moments and the computational complexity to calculate the Legendre moments is much reduced, and it is independent of the window size.

In this project, we have implemented an invariant representation of orientation fields based on the set of moments for fingerprint indexing. Moments are capable of describing fingerprint orientation patterns including singular regions and restoring spurious orientations in noisy fingerprints. Unlike most indexing schemes using the raw orientation data, a set of rotation moment invariants are derived from moments to form a compact feature vector, which is beneficial for clustering-based fingerprint indexing.

Moment features can provide the properties of invariance to scale, position, and rotation. We use moments to extract invariant features from the fingerprint images for the processing and clustering of the fingerprint images. This is a very important step in the Fingerprint processing techniques.

4.4.2. Definition of Moments

Given an N ×N image, let f (x, y) denote the grey level of the pixel at location (x, y), then $(p+q)^{th}$ order moment is defined as follows:

$$M_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x,y)\, dx\, dy \tag{4.6}$$

For $p, q = 0, 1, 2,...$

Thus Moments of images provide efficient local descriptors and have been used extensively in image analysis applications. Their main advantage is their ability to provide invariant measures of shape.

4.4.3. Moments Invariants

The 2-D moment of order (p + q) of a digital image f(x, y) of size M x N is defined as

$$M_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x,y) dx dy \tag{4.7}$$

For *p, q* = 0, 1, 2, Adapting this to scalar (grey scale) image with pixel intensities *I*(*x*, *y*), raw image moments M_{ij} are calculated by

$$M_{ij} = \sum_x \sum_y x^i y^j I(x,y) \tag{4.8}$$

In some cases, this may be calculated by considering the image as a probability density function, *i.e.*, by dividing the above by

$$\sum_x \sum_y I(x,y) \tag{4.9}$$

A uniqueness theorem (Hu [1962]) states that if *f*(*x*, *y*) is piecewise continuous and has nonzero values only in a finite part of the *xy* plane, moments of all orders exist, and the moment sequence (M_{pq}) is uniquely determined by *f*(*x*, *y*). Conversely, (M_{pq}) uniquely determines *f*(*x*, *y*). In practice, the image is summarized with functions of a few lower order moments.

Central moments are defined as

$$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x-\bar{x})^p(y-\bar{y})^q f(x,y)\,dx\,dy \tag{4.10}$$

Where $\bar{x} = \frac{M_{10}}{M_{00}}$ and $\bar{y} = \frac{M_{01}}{M_{00}}$

are the components of the centroid.

If $f(x, y)$ is a digital image, then the previous equation becomes

$$\mu_{pq} = \sum_{x}\sum_{y}(x-\bar{x})^p(y-\bar{y})^q f(x,y) \tag{4.11}$$

The central moments of order up to 3 are:

$$\begin{aligned}
\mu_{00} &= M_{00},\\
\mu_{01} &= 0,\\
\mu_{10} &= 0,\\
\mu_{11} &= M_{11} - \bar{x}M_{01} = M_{11} - \bar{y}M_{10},\\
\mu_{20} &= M_{20} - \bar{x}M_{10},\\
\mu_{02} &= M_{02} - \bar{y}M_{01},\\
\mu_{21} &= M_{21} - 2\bar{x}M_{11} - \bar{y}M_{20} + 2\bar{x}^2M_{01},\\
\mu_{12} &= M_{12} - 2\bar{y}M_{11} - \bar{x}M_{02} + 2\bar{y}^2M_{10},\\
\mu_{30} &= M_{30} - 3\bar{x}M_{20} + 2\bar{x}^2M_{10},\\
\mu_{03} &= M_{03} - 3\bar{y}M_{02} + 2\bar{y}^2M_{01},
\end{aligned} \tag{4.12}$$

It can be shown that:

$$\mu_{pq} = \sum_{m}^{p}\sum_{n}^{q}\binom{p}{m}\binom{q}{n}(-\bar{x})^{(p-m)}(-\bar{y})^{(q-n)}M_{mn} \tag{4.13}$$

4.4.4. Estimation of Moments using Legendre polynomials

Moments with Legendre polynomials as kernel function, denoted as Legendre moments, were first introduced by Teague. Legendre moments belong to the class of orthogonal moments, and they were used in biometric recognition applications. They can be used to attain a near zero value of redundancy measure in a set of moment functions, so that the moments correspond to independent characteristics of the image. By convention, the translation and scale invariant functions of Legendre moments are achieved by using a combination of the corresponding invariants of geometric moments. They can also be accomplished by normalizing the translated and/or scaled images using complex or geometric moments. However, the derivation of these functions is not based on Legendre polynomials. This is mainly due to the fact that it is difficult to extract a common displacement or scale factor from Legendre polynomials.

The purpose of fingerprint indexing is to reduce the search space of automatic fingerprint identification. Extraction of reliable and discriminative features is crucial for fingerprint indexing. Orientations can effectively capture the ridge flow structure. Fingerprint images are typically divided into smaller blocks to capture more detailed orientations, but the resulting feature vector will have too many elements. In this project we propose to use Legendre Moments to generate compact and rotation invariant features for fingerprint indexing.

4.4.5. Calculation of the Legendre Moments

The Legendre moment of order p+q of an object with intensity function f(x, y) are defined as follows:

$$L_{pq} = \frac{(2p+1)(2q+1)}{4} \int_{-1}^{1} \int_{-1}^{1} P_p(x) P_q(y) f(x, y)\, dx\, dy, \tag{4.14}$$

Where the kernel function $P_p(x)$ denotes the p^{th} order Legendre polynomial and is given by,

$$P_p(x) = \sum_{k=0}^{p} C_{pk}[(1-x)^k + (-1)^p(1+x)^k],$$

(4.15)

With,

$$C_{pk} = \frac{(-1)^k}{2^{k+1}} \frac{(p+k)!}{(p-k)!(k!)^2}.$$

(4.16)

Since Legendre polynomials are orthogonal over the interval [¡1, 1] , a square image of N x N pixels with intensity function f(i , j), with 1≤I, j≤N, must be scaled to be within the region -1 ≤ x, y ≤ 1. When an analog original image is digitized to its discrete form, the 2D Legendre moments L_{pq} defined by Eq. (1) is usually approximated by the formula

$$L_{pq} = \frac{(2p+1)(2q+1)}{(N-1)^2} \sum_{i=1}^{N} \sum_{j=1}^{N} P_p(x_i) P_q(y_j) f(x_i, y_j).$$

(4.17)

Where x_i=(2i-N-1)/(N-1) and y_j=(2j-N-1)/(N-1), and for a binary image, f(x_i, y_j) is given as

$$f(x_i, y_j) = \begin{cases} 1, & \text{if } (i, j) \text{ is in the original object,} \\ 0, & \text{otherwise.} \end{cases}$$

(4.18)

To improve the accuracy, the approximated form proposed is as follows:

$$\tilde{L}_{pq} = \frac{(2p+1)(2q+1)}{4} \sum_{i=1}^{N} \sum_{j=1}^{N} h_{pq}(x_i, y_j) f(x_i, y_j).$$

(4.19)

Where

$$h_{pq}(x_i, y_j) = \int_{x_i-\Delta x/2}^{x_i+\Delta x/2} \int_{y_j-\Delta y/2}^{y_j+\Delta y/2} P_p(x) P_p(y)\, dx\, dy \tag{4.20}$$

with $\Delta x = x_i - x_{i-1} = 2/(N-1)$ and $\Delta y = y_j - y_{j-1} = 2/(N-1)$.

Using these equations we can estimate the Legendre Moments of the fingerprints.

4.5. Fingerprint Indexing

4.5.1. Introduction

Given a query fingerprint, the purpose of fingerprint indexing is to reduce the number of candidates for fine-matching with a desired accuracy and facilitate the complex search of database in AFIS. Exclusive classification is a fast indexing scheme, but it cannot sufficiently reduce the search space due to its intrinsic problems, i.e., the small number of classes and skewed fingerprint distributions. Continuous classification can avoid these problems and achieve better retrieval performance. But it only performs the sequential search of database while neglecting the similarities among database templates, which limits the efficiency of fingerprint indexing for large database. A clustering based indexing approach was employed to take advantages of both database clustering and continuous classification to speed up the query process without compromising indexing accuracy.

Fingerprint based identification systems search through a large database of fingerprint entries for possible matches based on the given query fingerprint. Each entry has an associated identity and a matching function is used in order to determine the similarity between two fingerprints. However, due to the large number of entries in the database, one to one matching of the query fingerprint with each fingerprint in the database would be computationally infeasible. Therefore, a filtering process is usually invoked in order to reduce the number of candidate hypotheses for matching operation.

Filtering can be achieved by two different approaches: classification and indexing.

Classification involves partitioning the database into multiple classes and comparing the query fingerprint with prints belonging to the class assigned to the query print. In contrast, indexing assigns an index value to each fingerprint and, therefore the query print is compared with those fingerprints in the database which have comparable index values.

Fingerprint classification schemes based on human defined categories (such as Left Loop, Right Loop, Arch, Tented Arch and Whorl) have an inherent problem due to the small number of classes and the uneven distribution of fingerprints across these classes. Furthermore, most classification schemes are based on the configuration of singular points (i.e., core and delta points) which may not be available in prints acquired using small-sized sensors. In this work, we focus on indexing techniques for fingerprint Clustering.

The purpose of fingerprint indexing is to reduce the search space of automatic fingerprint identification. Extraction of reliable and discriminative features is crucial for fingerprint indexing. Orientations can effectively capture the ridge flow structure. Fingerprint images are typically divided into smaller blocks to capture more detailed orientations, but the resulting feature vector will have too many elements. Alternately, we employ Legendre Moments to generate compact and rotation invariant features for fingerprint indexing.

Fingerprint classification is the problem of assigning a fingerprint to a class, generally based on global features (e.g., ridge structure and singularities), in an efficient and reliable way. However fingerprint indexing methods are used to improve the performance of the Fingerprint Recognition by clustering the fingerprint database into several clusters that contain similar fingerprints and thereby reduces the search space in Fingerprint Recognition system.

The following figure shows main fingerprint classifications:

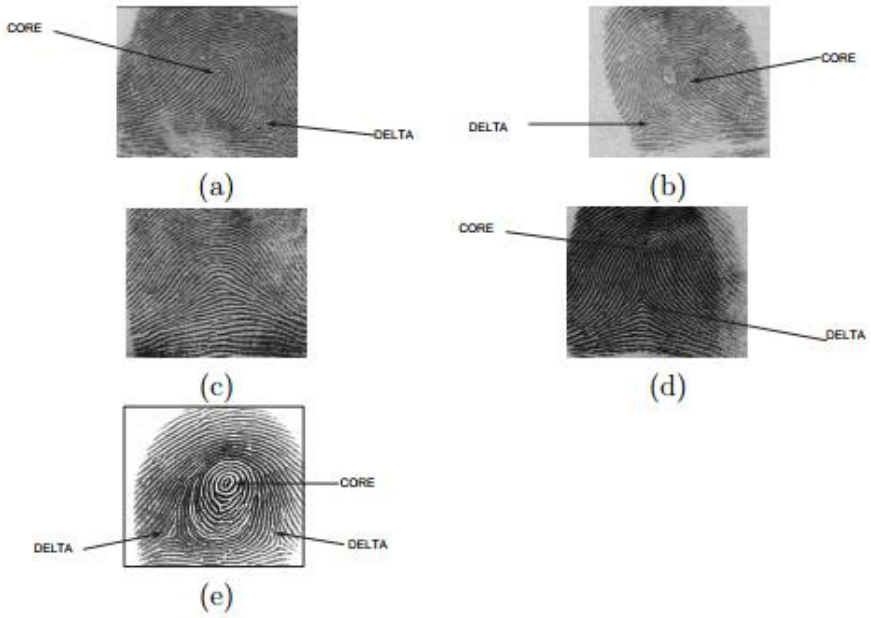

Figure 4.7 Fingerprint classification based on singularity points. (a) Left Loop. (b) Right Loop. (c) Arch. (d) Tented Arch. (e) Whorl.

4.5.2. Clustering-based indexing

In this project, we have implemented the K-means clustering for indexing the fingerprint database due to its high computational efficiency and low memory space requirements. The main purpose of clustering for fingerprint indexing is to facilitate a fast and effective query process. Fingerprint Indexing by clustering the fingerprint database into several clusters that contain similar fingerprints and thereby narrows down the search space in Fingerprint Recognition system. The next section describes the K-means Clustering approach which is used in our project.

4.5.3. K-means Clustering

K-means clustering is a widely used in partition clustering algorithm due to its high computational efficiency and low memory space requirements. This is employed to partition the moment-based feature space into a number of non-overlapping clusters. Each cluster is represented with a prototype (i.e., its mean vector) and each pattern is assigned to the cluster with the closest prototype iteratively until the cluster labels do not change. In the next step, the query fingerprint is compared with all cluster prototypes and the clusters close to the query are selected as the candidates first. The number of clusters is much smaller than the number of database templates so that the query process is speeded up by using clustering.

The number of clusters needs to be specified initially in K-means clustering. Instead of understanding the inherent data structure correctly, the main purpose of clustering for fingerprint indexing is to facilitate a fast and effective query process. Thus, the initial number of clusters is set to balance the time efficiency and accuracy of fingerprint indexing. After grouping N_F finger-prints into K_f clusters, the average number of comparisons in the query process is approximately computed as $K_f+(N_f/Kf)$ for retrieving the nearest cluster followed by sequential fingerprint search of the retrieved cluster. It is minimized when $K_f= \sqrt{N_f}$. However, multiple clusters close to the query fingerprint are often retrieved to avoid the problem of ambiguous fingerprint retrieval and improve the indexing accuracy.

The performance of fingerprint indexing is evaluated by the penetration rate vs. indexing accuracy (or error). The penetration rate is the average portion of database selected as the most likely candidates by fingerprint indexing over all query fingerprints. It indicates how much the fingerprint indexing can narrow down the search of database and is controlled by adjusting the size of the retrieval neighborhood in our employed indexing algorithm. Fingerprint indexing is considered successful if one of the selected candidates is from the same finger of query fingerprint. It is more likely to get the correct one if more candidate fingerprints are retrieved from the database. The indexing accuracy is thus computed as the portion of query fingerprints with correct retrieval at a given penetration rate. The tradeoff

between the penetration rate and indexing accuracy can be adjusted by changing the size of retrieval neighborhood.

4.5.4. K-Means Clustering Algorithm

This algorithm aims at minimizing an *objective function*, in this case a squared error function. The objective function

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - c_j \right\|^2 , \quad (4.21)$$

Where $\left\| x_i^{(j)} - c_j \right\|^2$ is a chosen distance measure between a data point $x_i^{(j)}$ and the cluster centre c_j, is an indicator of the distance of the *n* data points from their respective cluster centres.

The algorithm is composed of the following steps:

1. Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the K centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

Although it can be proved that the procedure will always terminate, the k-means algorithm does not necessarily find the most optimal configuration, corresponding to the global objective function minimum. The algorithm is also significantly sensitive to the initial randomly selected cluster centers. The k-means algorithm can be run multiple times to reduce this effect.

CHAPTER 5

IMPLEMENTATION

CHAPTER 5

5. IMPLEMENTATION

5.1. Implementation

Implementation is a process of realizing the design of the system into working model for this we need to decide which particular programming language to use so that the minimum error is put in programming the implementation of the system. Choosing a good programming language is a very critical step because one needs to think of the various aspects of the system and has to select a language that is more suitable for the system.

5.2. Implementation Requirements

The implementation decisions are mainly concerned with the care of future enhancement of the system. Those include the following:

- Operating system is windows
- The language chosen is matlab
- User interface is designed using Guide interface in mat lab.

5.2.1. Platform Selection

Windows is platform chosen to implement this project. The reason is being the efficiency of the platform.

5.2.2. Language Selection

Mat Lab was chosen to implement this project. For both front end and backend coding, Mat Lab is used.

5.2.3. GUI Selection

GUI for the project is implemented using Mat Lab is user friendly and easily learnt and modifiable.

Advantages of using mat lab GUI are as follows:

- No special configurations or changes are needed on the user's PC.
- Low cost.
- Efficient verification and validation.

5.3. Coding Standards

Any software is going to last for many years. If we produce low quality code that others find it hard to understand and maintain, then it might as well be thrown away and rewritten from scratch. Unfortunately this happens all too often. Making code readable and maintainable is as important, or is more important than making it work correctly, if it does not work, it can be fixed. If it is not maintainable it's a scrap.

- Matlab help programmers to go into any code and figure out what is going on.
- New people can get up to speed very quickly.
- People new to Mat lab are spared of making same mistakes over and over again.

So we follow the following naming conventions, style of comments, readable and maintainable code.

5.3.1. Naming conventions

- **Regular Variables**: Use mixed case, starting with lower case. Avoid names that are too short and cryptic, but also avoid extremely long names.

- **Global Variables**: Global variables must be clearly declared after function signature.

- **Function Names**: Function names are in mixed case, and start with upper case. They should normally be a verb, or a verb followed by noun.

- **Constants**: For global constants, or constants declared inside the functions use all uppercase with underscores to separate the words.

- **Keep Names Consistent**: Avoid using different names for the same concept.

5.3.2. Comments

Put always a header comment at every function definition. Use brief comments throughout the code to increase the code understandability. In general, comments should be brief and clear. Don't restate code with comments by giving information in your comments that is obvious from the code. Comment should precede the block of code with which they are associated and should be intended to the same level. This makes it easier to see the code.

- **Comments should explain WHY instead of HOW:** By definition high quality code is clear and readable enough anyway, so that the average programmer can easily understand how it works just by reading it. However, the reason why a particular algorithm is chosen, or why a certain action must be taken, usually cannot be derived just be reading the code. This is the information that should be documented in comments throughout the code.

- **Function Headers**: Function header comments must be used in every function that we write. A function header should contain a brief description of what the function does. Implementation details in the header must be avoided on WHAT the function does, not HOW it does. The header should be the "user's manual" for whoever uses this function.

5.3.3. Readable and maintainable code

Some of the thumb rules that should be followed to promote a readable and maintainable code are given.

- **Function length**: Try to write small functions, no longer than 2 pages of the code. This makes it easier to read and understand them.
- **Functional declaration**: Dummy parameters names in the function declaration must not be omitted unless the meaning is clear without them.
- **Avoid having too many parameters in the function**: Functions that take too many parameters often indicate poor code design and poor usage of data structures. Avoid using deeply nested code and avoid having too many levels of nesting in the code.
- **Don't duplicate the code**: If we find having the code in two or more functions, then pull the code out of those functions and create a new function with it and call this new function.
- High level language for technical computing.
- Development environment for managing code, files, and data.
- Interactive tools for iterative exploration, design and problem solving.
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization and numerical integration.
- 2-D and 3-D graphics functions for visualizing data.
- Tools for building custom graphical user interfaces.
- Functions for integrating MATLAB based algorithms with external applications and language such as C, C++, FORTRAN, Java, COM, and Microsoft Excel.

CHAPTER 6

TESTING

CHAPTER 6

6. TESTING

6.1. Software Testing

Software testing is the activity aimed at evaluating an attribute or capabilities of a program or system and determining that it meets its required result. Testing is an integral part in software development. It is broadly deployed in every phase in the software development cycle. Typically, more than 50% of the development time is spent in testing.
Testing objectives include:

- Testing is a process of executing a program with intent of finding an error.
- A good test case is one that has a high probability of finding a yet undiscovered error.
- A successful test is one that uncovers a yet undiscovered error.

Testing should systematically uncover different classes of errors in a minimum amount of time and with a minimum amount of effort. A secondary benefit of testing is that it demonstrates that the software appears to be working as stated in the specifications. The data collected through testing can also provide an indication of the software's reliability and quality. But, testing cannot show the location of defect, it can only show that software defects are present.

6.2. Test Plan

Testing is carried out in multiple levels. Activities at each level must be planned well in advance and it has to be formally documented. Based on the individual plan only, the individual test levels are carried out .Testing process start with a test plan that identifies all

the testing related activities that must be performed, specifies the schedule, allocates the resources and specifies guidelines for testing.

The testing for this project is done in three phases, unit testing, integration testing and system testing. For the failure test cases regression testing is also done.

6.3. Unit Testing

A unit test is a method of testing the correctness of a particular module of source code. The idea is to write test cases for every non-trivial function or method in the module so that each test case is separate from the others if possible. The goal of unit testing is to isolate each part of the program and show that the individual parts are correct.

The unit test plan must clearly specify the scope of unit testing. In this project, the unit test plan includes testing the basic input/output of the units along with their basic functionality. It also tests the boundary conditions. The input/output in the GUI mode pops up an error message in a separate window and resets the value to the defaults.

6.4. Integration Testing

Integration testing is the phase of software in which individual software modules are combined and tested as a group. This integration process involves building the system and testing the resultant system for problems that arises from interactions. Integration tests should be developed from the system specification and integration testing should begin as soon as usable versions of some of the system components are available. The integration test plan includes the test plans for testing the interfaces which will be used between two or more modules, dependencies between the modules and the functionality of the modules when grouped with other modules.

6.5. System Testing

Testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require to knowledge of the inner design of the code or logic.

6.6. Testing Strategy

The test strategies that are used for software testing in this project have the following details.

- **Feature to be tested** - Feature to be tested mainly contain the important features of the single or individual unit of project. It also contains the selection of proper input to test the functionality or feature of the unit.
- **Items to be tested** - mainly contains the individual unit or module or function which collectively forms the system.
- **Purpose of testing** - is used to verify whether the test cases are able to identify the errors or bugs present in the module, unit or any function.
- **Pass/Fail criteria** - to give the result of the test case the Actual output is taken and compared with the expected output. If both are matching then the criteria is given as pass else fail. In case of failure of the test cases the reason for that failure and the way in which the bug is fixed is given. The test cases are re-executed from the beginning.
- **Assumptions and Constraints** – while executing some of the test cases some parameters required for execution are assumed as some of the modules or functions will depend on the output of the other module or function.

6.7. Unit test cases

- Test cases for Input/output check
- Test cases for GUI
- Test cases for selection of mode of operation

6.8. Integration test cases

- Test case for GUI integration

CHAPTER 7

EXPERIMENTAL RESULTS

CHAPTER 7

7. EXPERIMENTAL RESULTS

OVERVIEW OF GUI

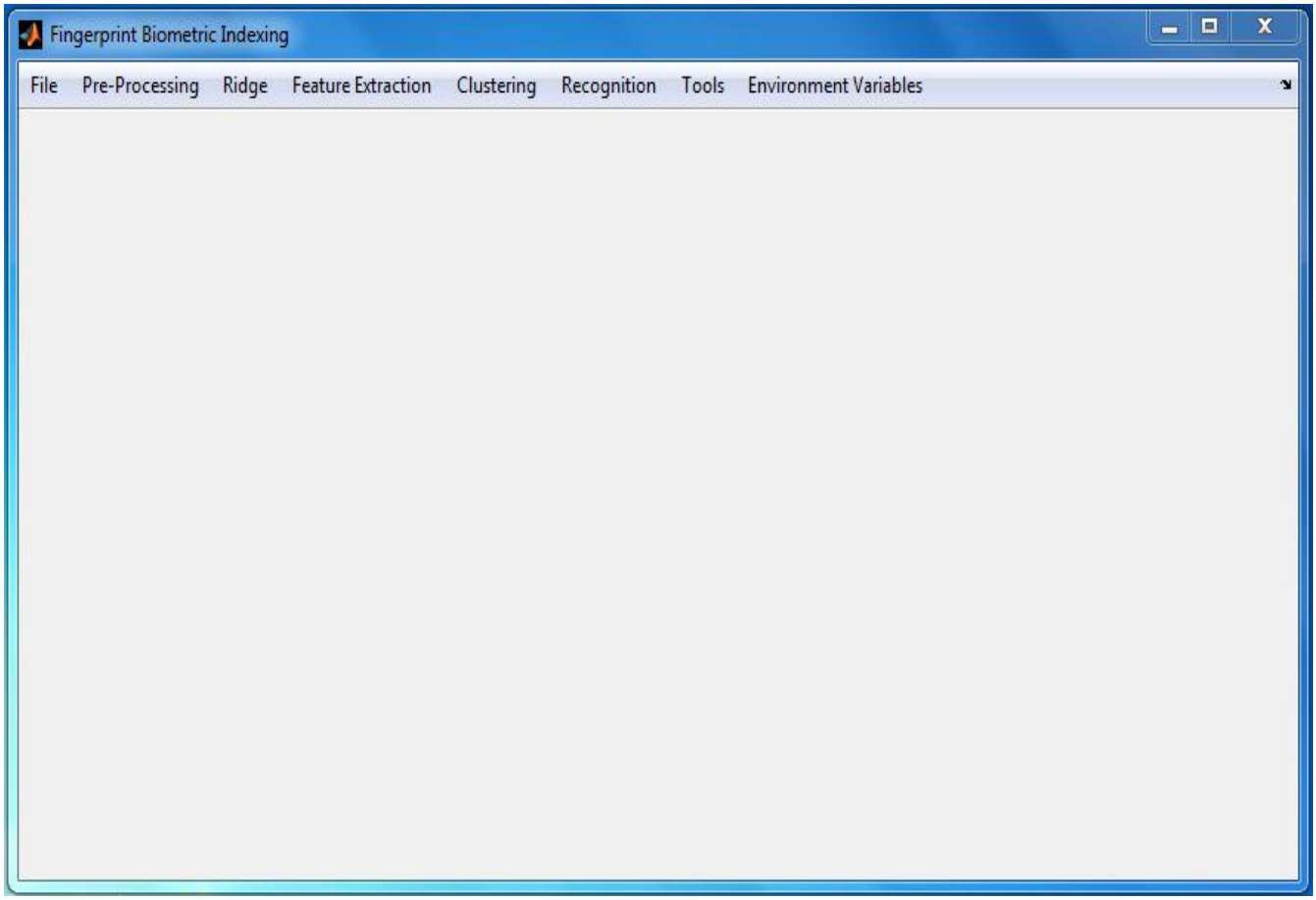

Figure 7.1 Overview of GUI

Input Image

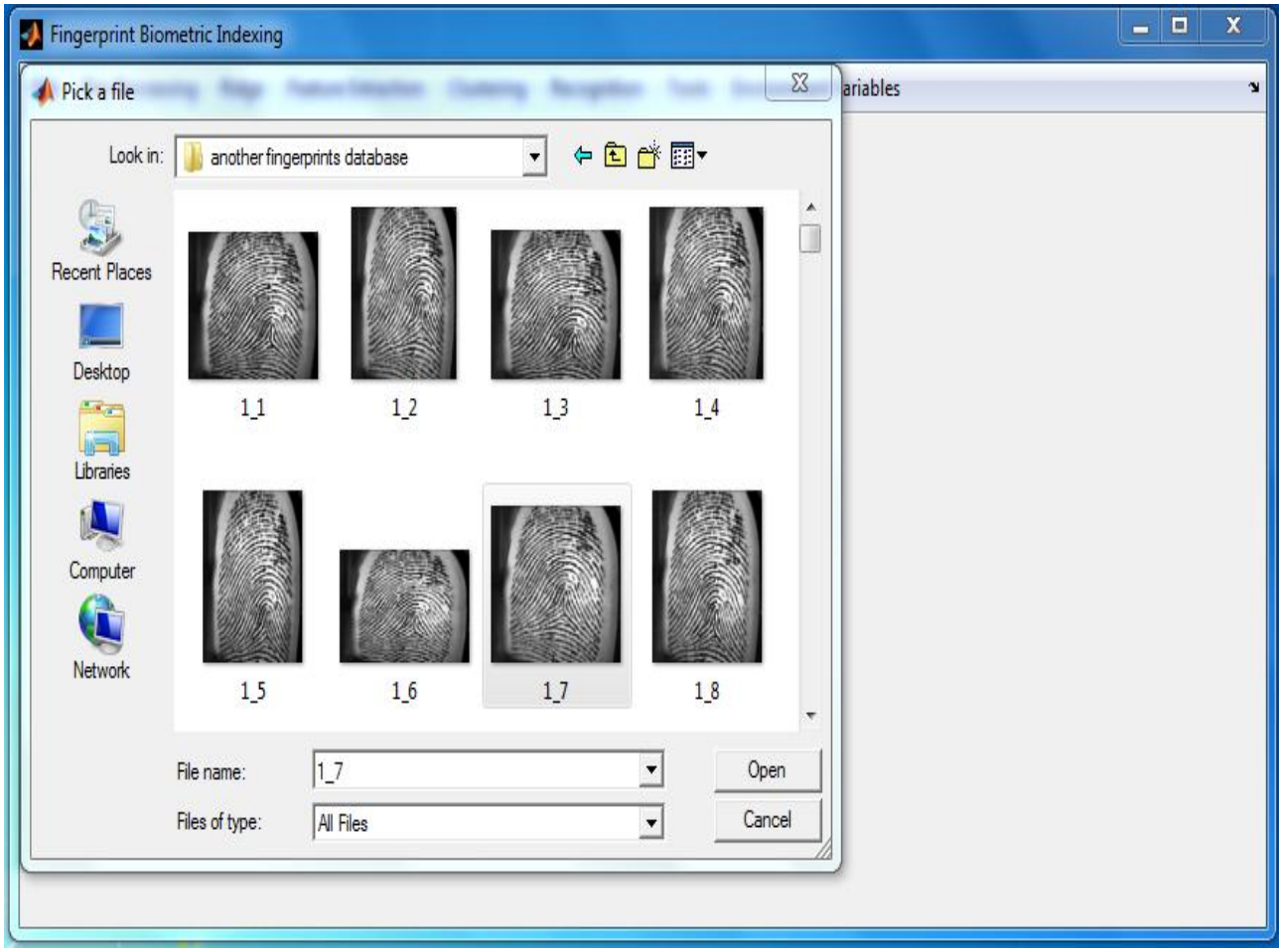

Figure 7.2 Opening an image

Window to open the fingerprint image from the source location and it is given as a input to the further processing.

Input and Histogram Image

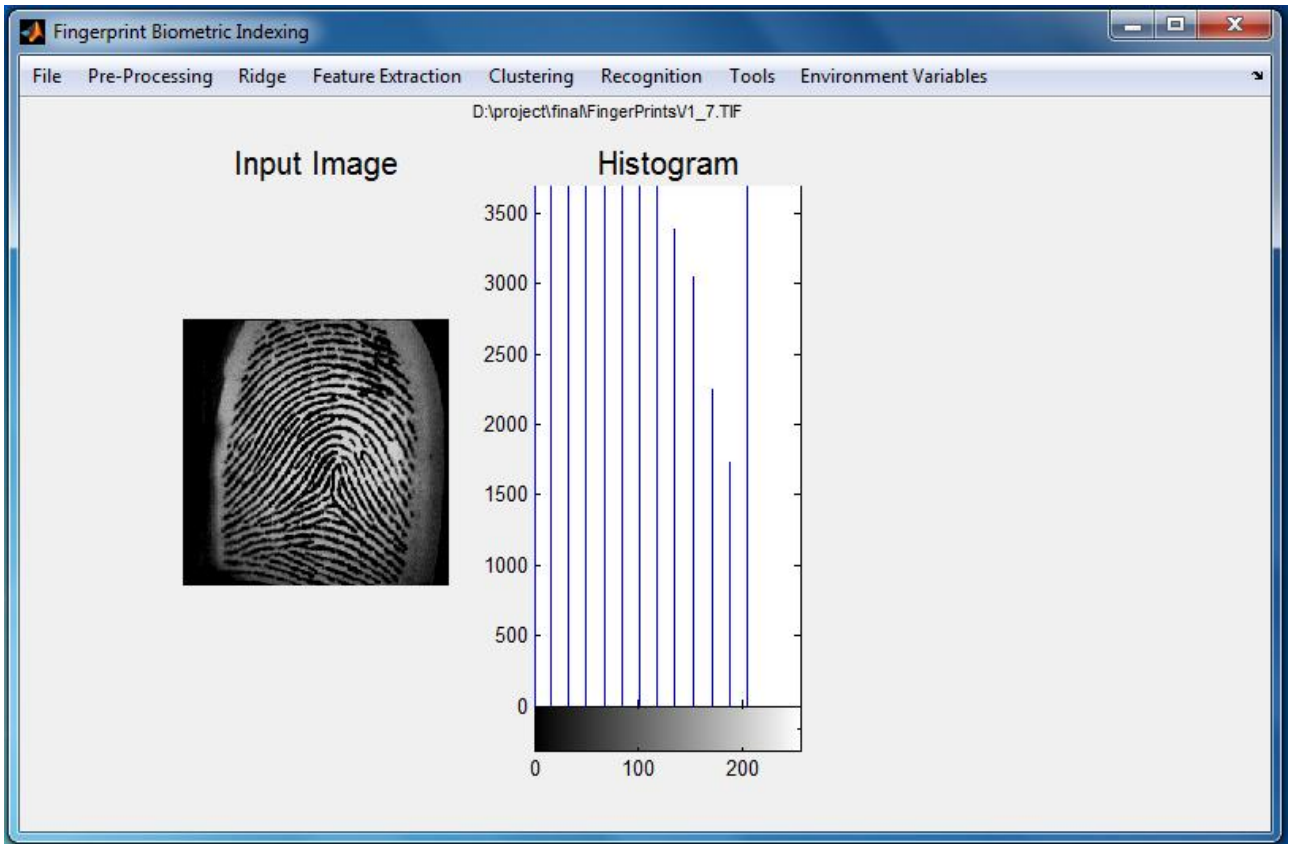

Figure 7.3 Image is displayed with Histogram

Histogram Equalize Image

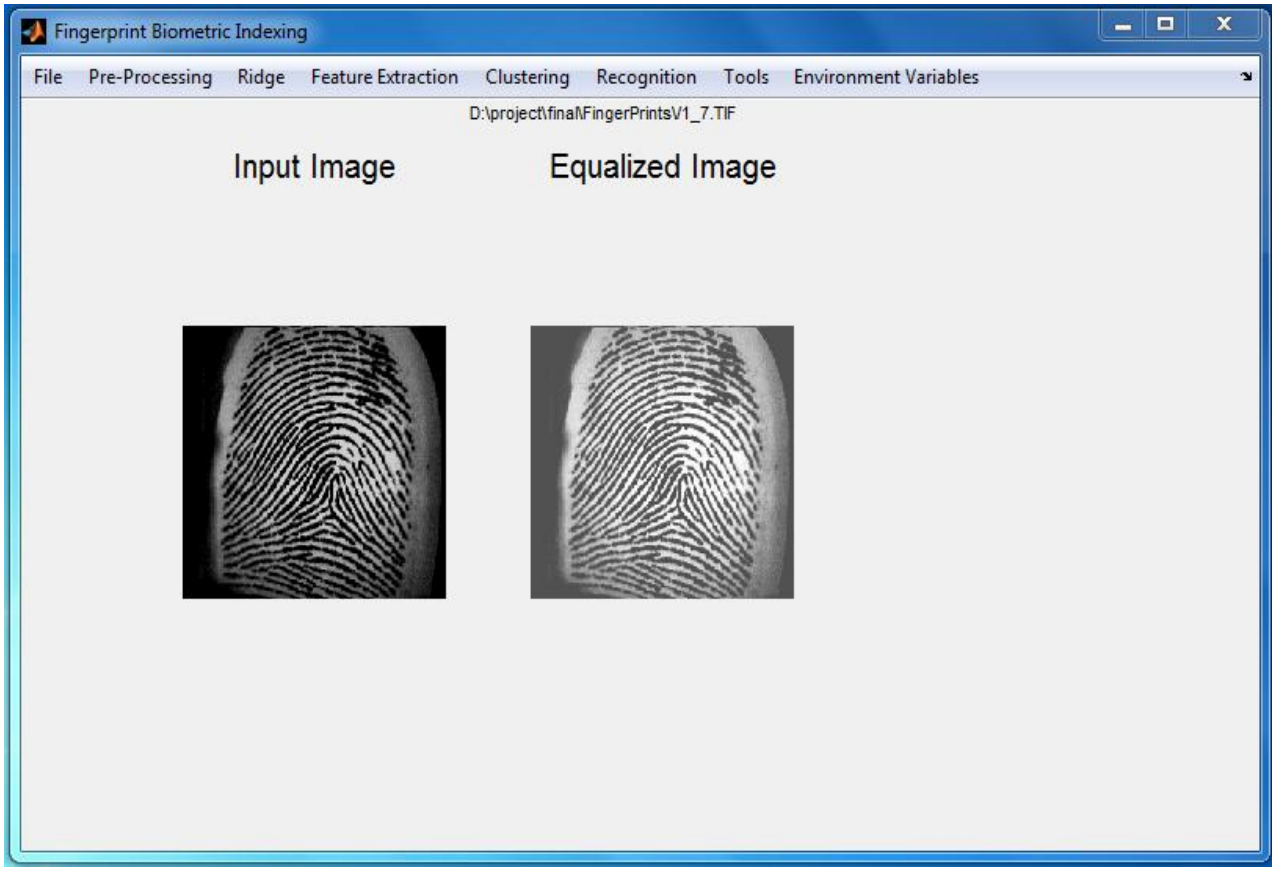

Figure 7.4 Histogram Equalize

The above figure shows the histogram equalization operation where image at the left is input image and image at the right is histogram equalized image.

Binarized Image

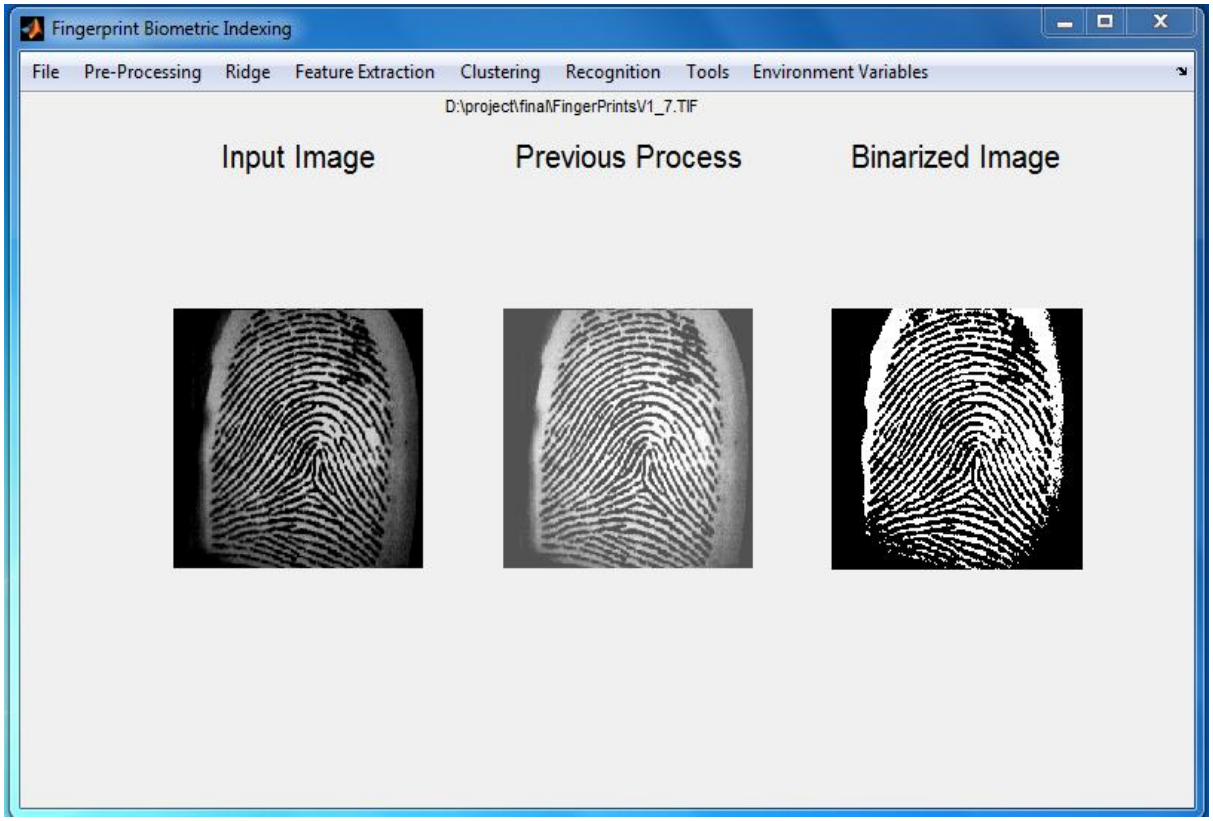

Figure 7.5 Binarized image

The image at the extreme right is the result of Binarization operation which applied on histogram equalized image.

Orientation with Original Image

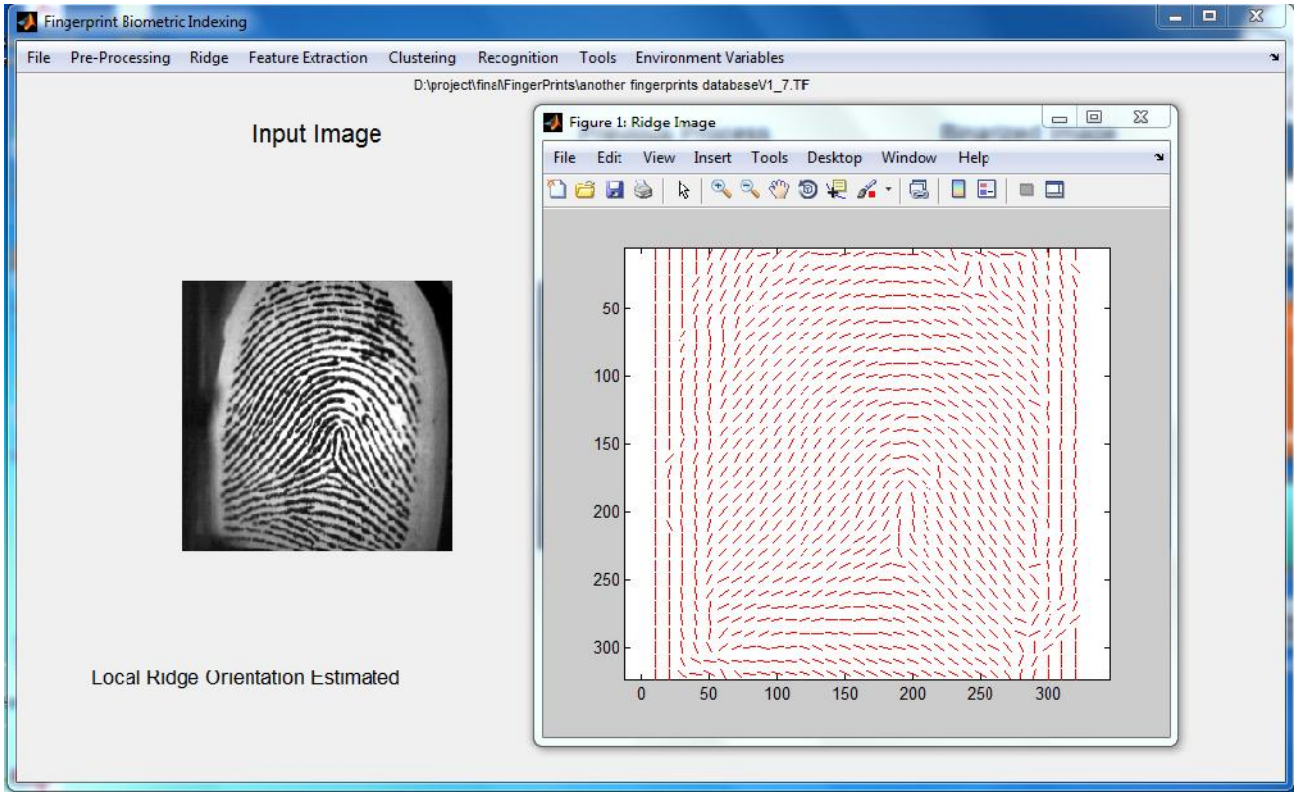

Figure 7.6 Orientation

The above figure shows the plot of the ridge orientation of the given input fingerprint image.

Legendre Moments calculation

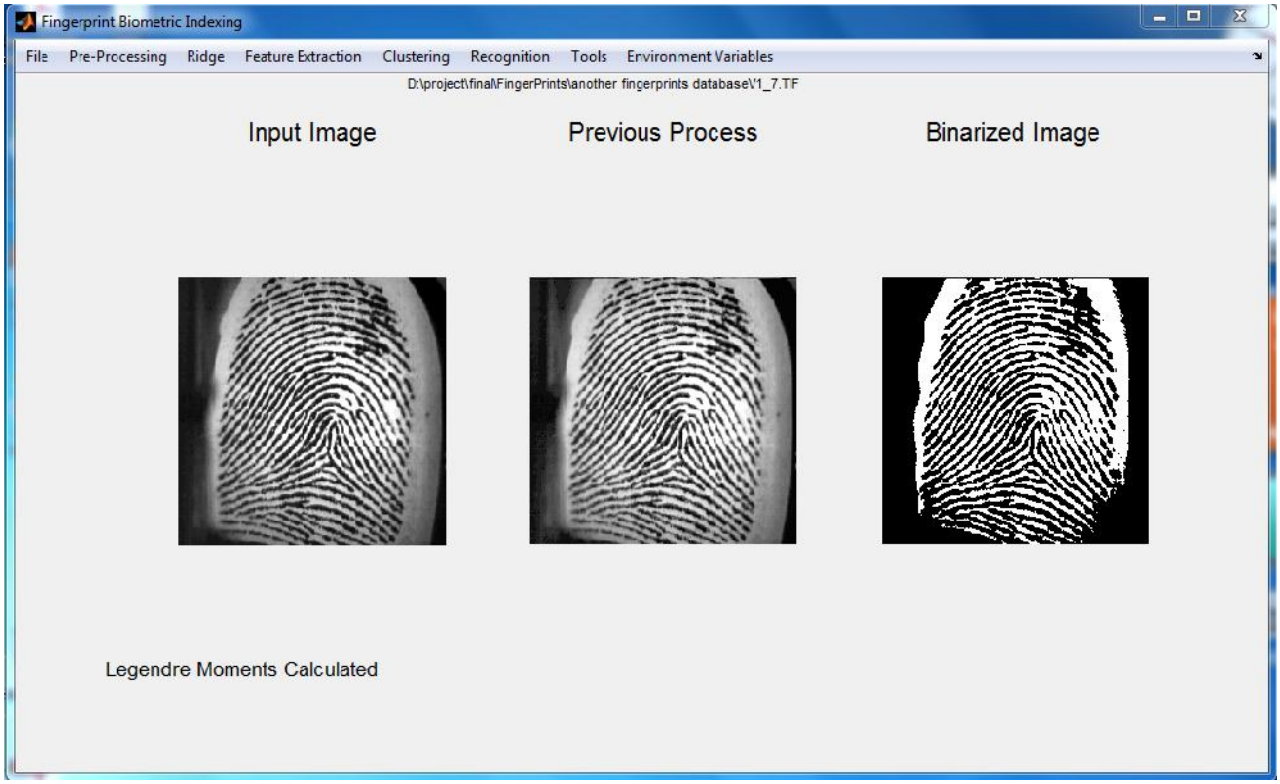

Figure 7.7 Legendre Moments

In this step the Legendre moments are calculated using the ridge orientation result.

Sorting Image using Cluster values as reference

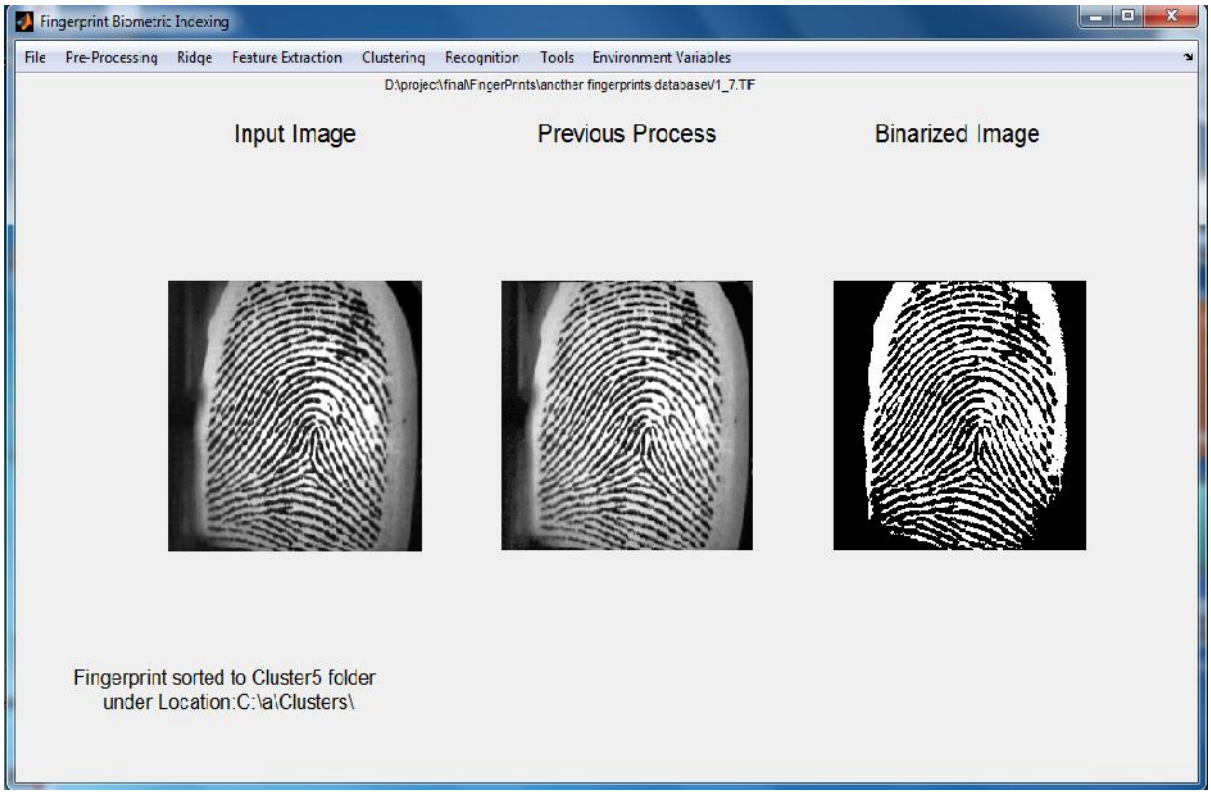

Figure 7.8 Adding Image to Cluster

After processing all the steps, now the fingerprint image is compared with the reference cluster value and it is sorted according to the reference cluster value.

Searching for matched Image

Figure 7.9 finding the matched image from the existing database

Finding the matched fingerprint image from the existing database, if the fingerprint image is not present in the clusters then it will display a nearly matched fingerprint image.

Clusters

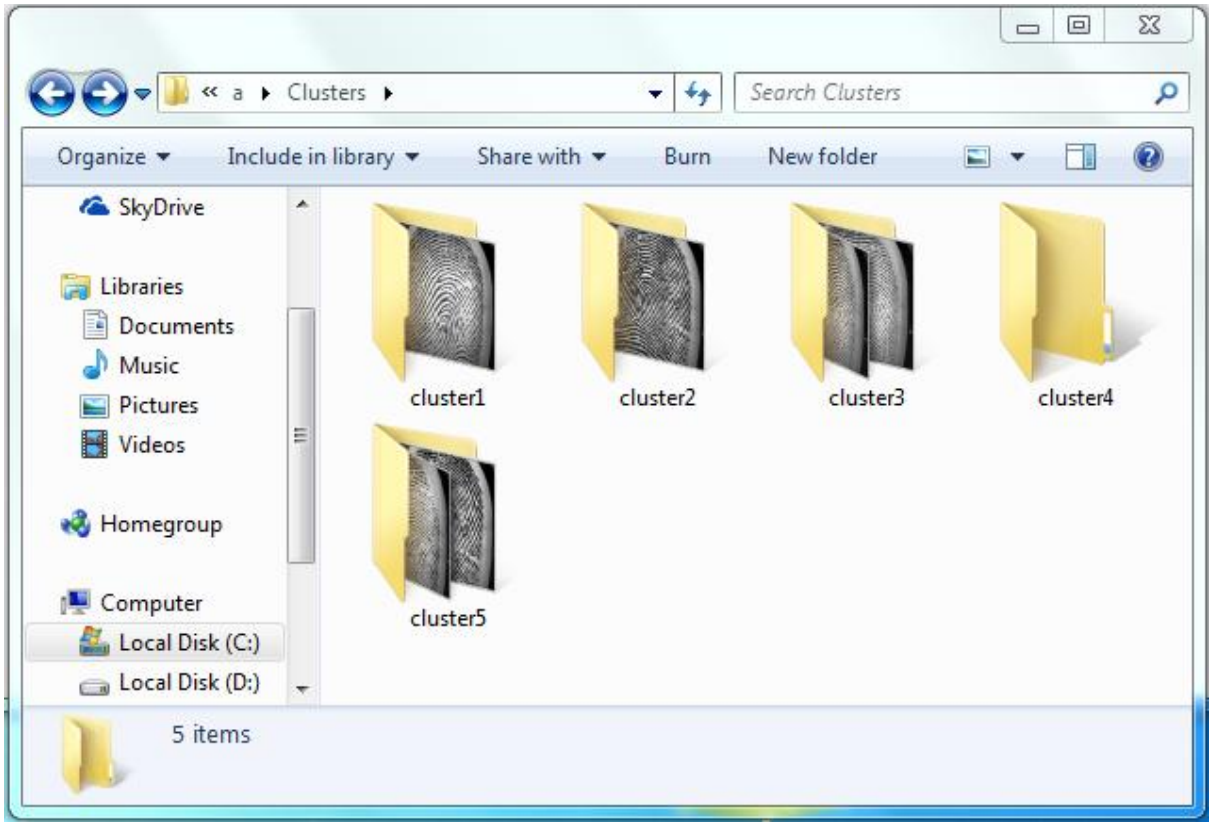

Figure 7.10 Database of the Image where they are ordered according to the Cluster Value

The above figure shows the clustering operation in process.

CHAPTER 8

CONCLUSION

CHAPTER 8

8. CONCLUSION

8.1. Overall Summary

Fingerprint recognition systems are used majorly in personal identification. The performance of these systems depends on the number of fingerprint in the database. As this increases, system takes longer time for recognition. Therefore indexing is required in order to reduce the search time during the matching process. Fingerprint indexing can be done by classifying the fingerprints into different groups.

In our project, ridge orientation field is used as a feature vector to represent individual fingerprint. We have worked on different edge detectors to estimate ridge orientation. In this stage of feature extraction, we could obtain the desired result using canny edge detector for ridge orientation. We then proceeded towards numerical representation of these features through calculation Legendre moments. The Euclidean distance between these moments where calculated on basis of which K-means clustering was performed. This clustering process indexes the database into several clusters and therefore narrows down the search space which is the objective of the project.

To achieve accuracy in this process, the input database is primarily subjected to several image enhancement operations, since the quality of the image plays major role in any image processing operations. The input images must be moderate in quality; the performance of the system reduces with poor quality images. The system doesn't take into account occlusion and obliteration of a part of the fingerprint. Work carried out so far does not include any performance analysis, therefore the future enhancement could be designed and development new algorithms using other types of moment features and carryout performance.

9. BIBLIOGRAPHY

[1] Manhua Liu, Xudong Jiang, Alex Chichung Kot, Efficient fingerprint search based on database clustering.

[2] D. Maltoni, D. Maio, A.K. Jain, A. Prabhakar, Handbook of Fingerprint Recognition, Springer, New York, 2003.

[3] K. Karu, A.K. Jain, Fingerprint classification, Pattern Recognition .

[4] Rafeal C Gonzalez, Richard E Woods, Digital Image Processing, 3rd edition

[5] Alasdair McAndrew . Introduction to Digital Image Processing with Matlab.

[6] A.K. Jain, J. Feng, K. Nandarkumar, Fingerprint matching, IEEE Computer 43 (2) (2010) 36–44.

[7] N.K. Ratha, S. Chen, K. Karu, A. Jain, A real-time matching system for large fingerprint databases, IEEE Transactions on Pattern Analysis and Machine Intelligence 18 (8) (1996) 799–813.

[8] R. Cappelli, A. Lumini, D. Maio, D. Maltoni, Fingerprint classification by directional image partitioning, IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (5) (1999) 402–421.