PERSONAL AUTHENTICATION USING 3-D FINGER GEOMETRY

P. 2177

A PROJECT REPORT

Submitted By

Senthalir.P **71204205048**
MeenaKaruna.J.S **71204205021**

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600025

APRIL 2008

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "PERSONAL AUTHENTICATION USING 3-D FINGER GEOMETRY" is a bonafide work of **"SENTHALIR.P and MEENAKARUNA.J.S"** who carried out the project work under my supervision.

SIGNATURE

Dr.S.Thangasamy

DEAN

Computer Science and Engineering

Kumaraguru College of Technology

Coimbatore-641006

SIGNATURE

Mr.E.A.Vimal

SUPERVISOR

Lecturer,

Information Technology

Kumaraguru College of Technology

Coimbatore-64106

The candidates with University Register Nos **71204205048** and **71204205021** were examined by us in the project viva-voice examination held on 24/04/08

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

The exhilaration achieved upon the successful completion of any task should be definitely shared with all the people behind the venture. This project is an amalgam of study and experience of many people without whose help this project would not have taken shape.

At the onset, we take this opportunity to thank the management of our college for providing us excellent facilities to work with. We express our **Dr.JOSEPH V.THANIKAL, B.E, M.E, Ph.D, PDF. CEPIT,** Principal, Kumaraguru College of Technology for ushering us to the path of triumph.

With great veneration and sincere gratitude we express our profound thanks to our beloved Dean of Computer Science and Engineering, **Dr.S.THANGASAMY**, for his immense encouragement and support for being a source of inspiration all through course of study.

We would like to place on record our heartfelt gratitude to our revered course co-coordinator Mr.**K.R.BASKARAN**, Department of Information Technology for his critical remarks and suggestions

We are very much obliged to express our sincere thanks and gratitude to our guide **Mr.E.A.VIMAL**, Lecturer, Department of Information Technology for his invaluable guidance, discussions, ideas, suggestions and encouragements in all phases of this project work.

We also feel elated in manifesting our deep sense of gratitude to all the staff and lab technicians in the department.

We express our heart-felt thanks to all our family and friends who were sharing their contribution in many subtle ways and indeed instrumental for lending us tips, support and co-operation throughout the project work.

ABSTRACT

Abstract

A biometric authentication system based on the measurements of user's three-dimensional (3-D) hand geometry is proposed. By exploiting the 3-D information, we provide a unique and secure way of access to each and every user within a working environment. Our finger geometry system relies on a novel real-time database that holds a collection of hand images. The system takes the image and performs various image processing proceedings to extract relevant information and stores them as a template of image information. After training the images, these templates are matched with the input for desired responses. When there is a match, the system will be able to identify the user with minimal time and errors.

Efficient, close to real-time algorithms for hand segmentation, localization and 3-D feature measurement are described and tested on an image database that simulates a variety of working conditions, and thus greatly contributing to the unobtrusiveness of the system. The performance of the system is shown to be similar to state-of-the-art hand geometry authentication techniques.

CONTENTS

TABLE OF CONTENTS

HAPTER No	TITLE	P.No.
	ABSTRACT	iv
	LIST OF TABLES	v
	LIST OF FIGURES	vi
1.	**INTRODUCTION**	1
	1.1 GENERAL	3
	1.2 PROBLEM DEFINITION	9
	1.3 OBJECTIVE	10
2.	**LITERATURE REVIEW**	11
	2.1 FEASIBILITY STUDY	11
	2.2 CURRENT STATUS OF THE PROBLEM	16
	2.3 PROPOSED SYSTEM AND ITS ADVANTAGES	17
	2.4 HARDWARE REQUIREMENTS	19
	2.5 SOFTWARE REQUIREMENTS	19
	2.7 SOFTWARE OVERVIEW	19
3.	**DETAILS OF METHODOLOGY EMPLOYED**	21
4.	**PERFORMANCE EVALUATION**	27
	4.1 UNIT TESTING	29
	4.2 SYSTEM TESTING	29
	4.3 INTEGRATION TESTING	30
	4.4 IMPLEMENTATION DETAILS	30

5.	**CONCLUSION**	32
6.	**FUTURE ENHANCEMENTS**	33
7.	**APPENDICES**	34
	7.1. SOURCE CODE	34
	7.2 SCREEN SHOTS	64
	7.3 SAMPLE IMAGES	68
8.	**REFERENCES**	69

LIST OF TABLES

S.No.	TABLE NAME	Table No.	P.No
1.	Drawbacks of existing biometric technologies	1.1	5
2.	Comparing various biometric technologies	1.2	6
3.	Typical Software quality factors	4.1	28

LIST OF FIGURES

S.No.	FIGURE NAME	FIG.No.	P. No.
1.	Different biometric Technologies	1.1	4
2.	Existing geometry system with pegs	2.1	16
3.	Existing system against a uniform background	2.2	16

INTRODUCTION

1. INTRODUCTION

As the level of security breaches and transaction fraud increases, the need for highly secure identification and personal verification technologies are becoming apparent. Biometric technologies are becoming the foundation of an extensive array of highly secure identification and personal verification solutions. Biometrics is an emerging activity that is likely to become an integral part of our daily lives, creating an enormous on-going market for products based on many state-of-the-art technologies such as optical, thermal, capacitance, ultrasonic sensors etc. Etymologically, Biometrics means measure of living body.

Biometrics is expected to be incorporated in solutions to provide for Homeland Security including applications for improving airport security, strengthening our national borders, in travel documents, visas and in preventing ID theft. Now, more than ever, there is a need for biometrics in federal, state, and local governments, military and commercial applications. Congressional offices and a large number of organizations involved in many markets are addressing the important role that biometrics will play in identifying and verifying the identity of individuals and protecting national assets.

The biometrics is most commonly defined as measurable psychological or behavioral characteristic of the individual that can be used in personal identification and verification. The driving force of the progress in this field is, above all, the growing role of the Internet and electronic transfers in modern society. Therefore, considerable number of applications is concentrated in the area of electronic commerce and electronic banking systems.

Biometric-based solutions are able to provide for confidential financial transactions and personal data privacy. Enterprise-wide network security infrastructures, government IDs, secure electronic banking, investing and other financial transactions, retail sales, law enforcement, and health and social services are already benefiting from these technologies.

Voice, lip movements, hand geometry, face, fingerprint, iris, vein and signature are the most commonly used authentication methods based on different personal characteristics.

1.1 General

The physiological biometrics is based on measurements and data derived from direct measurement of a part of the human body. Fingerprint, iris-scan, retina-scan, hand geometry and facial recognition are leading physiological biometrics. Behavioral characteristics are based on an action taken by a person. Behavioral biometrics is based on measurements and data derived from an action, and indirectly measure characteristics of the human body. Voice recognition, keystroke-scan, and signature-scan are leading behavioral biometric technologies. One of the defining characteristics of a behavioral biometric is the incorporation of time as a metric – the measured behavior has a beginning, middle and end.

A biometric system is essentially a pattern recognition system which makes a personal identification by determining the authenticity of a specific physiological or behavioral characteristic possessed by the user. An important issue in designing a practical system is to determine how an individual is identified. Depending on the context, a biometric system can be either a verification (authentication) system or an identification system.

Biometric authentication, once used for granting access to high security infrastructures, is gradually finding place in a wider range of applications. However, until today the requirement for highly reliable authentication has led to compromises with respect to user acceptance. It is clear that reliability and user convenience should coexist in order to achieve a widespread acceptance of biometrics.

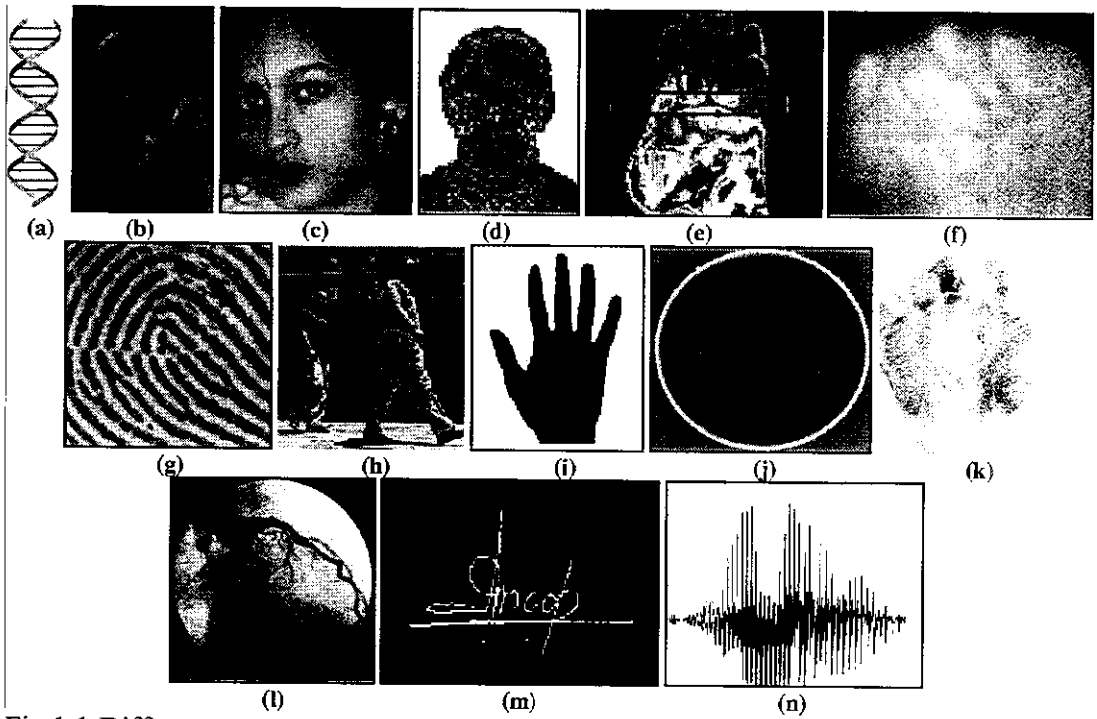


Fig.1.1 Different biometric Technologies: (a) DNA, (b) ear, (c) face, (d) facial thermo gram, (e) hand thermo gram, (f) hand vein, (g) fingerprint, (h) gait,(i) hand geometry, (j) iris, (k) palm print, (l) retina, (m) signature, and (n) voice.

1.1.1Comparison of various biometric technologies

It is possible to understand if a human characteristic can be used for biometrics in terms of the following parameters as shown in the table:

- **Universality** : each person should have the characteristic
- **Uniqueness** : is how well the biometric separates individually from another.
- **Permanence** : measures how well a biometric resists aging.
- **Collectability** : ease of acquisition for measurement.
- **Performance** : accuracy, speed, and robustness of technology used.
- **Acceptability** : degree of approval of a technology.
- **Circumvention** :ease of use of a substitute.

Table.1.1 COMPARISON OF VARIOUS BIOMETRIC TECHNOLOGIES

Biometrics	Universality	Uniqueness	Permanence	Collect ability	Performance	Acceptability	Circumvention*
Face	H	L	M	H	L	H	L
Fingerprint	M	H	H	M	H	M	H
Hand geometry	M	M	M	H	M	M	M
Keystrokes	L	L	L	M	L	M	M
Hand veins	M	M	M	M	M	M	H
Iris	H	H	H	M	H	L	H
Retinal scan	H	H	M	L	H	L	H
Signature	L	L	L	H	L	H	L
Voice	M	L	L	M	L	H	L
Facial thermograph	H	H	L	H	M	H	H
Odor	H	H	H	L	L	M	L
DNA	H	H	H	L	H	L	L
Gait	M	L	L	H	L	H	M
Ear Canal	M	M	H	M	M	H	M

- *- circumventability listed with reversed colors because low is desirable here instead of high*

(H=High, M=Medium, L=Low)

Apart from the parameters being compared, the various biometric technologies have the following backdrop when it comes to implementing in real time applications are listed in the table given below:

Biometric technologies	**Drawbacks**
Iris	Scanning closely causes a sensation of rejection, apparatus is bulky, high implementation cost
voice recognition	Not reliable since the human voice is subject to change during illness, hoarseness & other common throat problems or according to various noises from circumstances and static caused by inferior communication state.
Face	Recognized differently depending on triangle, expression and age, impossible to put into practical life
Vein	Complicated hardware structure and the high-cost entire system.
Signature	Easily reproduced and not reliable.

Table.1.2 Drawbacks of existing biometric technologies

1.1.2 Need for Hand Geometry:

The finger geometry recognition has been applied for the long time out of various biometric identification technologies. Nowadays, finger geometry recognition is considered the best choice for automated personalization of services and attendance tracking in working environments based on its *speed, reliability, non-intrusive interfaces, and cost – effectiveness* attaining 50% market shares in biometric markets.

1.1.3 Hand Geometry Technology

People's hands and fingers are unique but not as unique as other traits, like fingerprints or irises. That's why businesses and schools, rather than high-security facilities, typically use hand and finger geometry readers to authenticate users, not to identify them. One can use finger length, thickness, and curvature for the purposes of verification but not for identification.

In such situations it is desirable to have a biometric system that is sufficient for verification. Eg. Disney theme parks use finger geometry readers to grant ticket holders admittance to different parts of the park. Some businesses opt for hand geometry readers instead of timecards. Though many people's hands change over time due to injury or due to changes in weight or arthritis, systems update the data to reflect minor changes from day to day.

As hand geometry is not distinctive, it is the ideal choice. Furthermore, hand geometry data is easier to collect. With fingerprint collection good frictional skin is required by imaging systems, and with retina-based recognition systems, special lighting is necessary. Additionally, hand geometry can be easily combined with other biometrics, namely fingerprint. One can envision a system where fingerprints are used for (infrequent) identification and hand geometry is used for (frequent) verification.

Some of the areas where hand geometry finds its place include:

- Access to and from the Olympic Village was controlled.
- Colombian legislature and San Francisco International Airport
- Child day care centers to verify the identity of parents. Hospitals use hand geometry systems to monitor payroll accuracy and access control.
- The Fastgate (INSPASS) pilot program to track border crossings for frequent travelers.
- The University of Georgia uses for their student meal programs.
- All branches of the United States military & nuclear power plants
- Used to track prisoners and in International banks

1.2 Problem Definition

A problem of personal verification and identification is an actively growing area of research. In our global information society, there is an ever-growing need to authenticate individuals. The methods are numerous, and are based on different personal characteristics. Hand geometry recognition is one of the most popular biometrics used today for user verification. It works by comparing the 3-D geometry of the hand with a previously enrolled sample. A simple two-dimensional (2-D) camera is commonly used to capture an image of the user's palm. Features are extracted by analyzing the image contours of the hand views.

Various features such as width of the fingers, length of the fingers and width of the palm have been estimated Instead of using measurements of the hand for verification, uses points on a hand silhouette contour as features, while matching is based on the mean alignment error between two sets of silhouette points.

The major limitation of the above approaches is their obtrusiveness imposed by the use of pegs, which constrain the positioning and posture of the hand. Moreover, correct placement of the hand requires some training, and presents difficulties for specific user groups such as young children and elderly.

The problems are:

- Easily traceable by intruders
- Low reliability
- No unique identification

In these methods, an entire image is used for feature extraction.

1.3 Objective of the Project

With advent of Internet and technology development, there are many applications where identification and authentication of a person is required. The security policy determines the strength of authentication relating to the protection needs of information or resources. There are three methods to verify an identity presented to the system:

- something you know (PIN, Password, Pass phrase);
- something you have (smartcards, RF-ID, other token);
- Something you are (physiological or behavioral characteristics = biometrics).

Each of the methods as well as all possible combinations determines a specific security level. In this project, a medium level security authentication system that can overcome some of the limitations of the traditional automatic personal identification technologies is proposed.

The core objective is to provide the reliable authentication of individual with the help of finger geometry by imposing fewer constraints on the hand and on the environment incorporating the following modules to identify a user.

- Training Process
- Recognition Process
    - Thresholding
    - Fixing Boundary points
    - Enlargement

LITERATURE REVIEW

Inspired by the article Personal Recognition Using 3-D Finger Geometry, published in the IEEE Transactions on Information Forensics and Security, Volume.1, No.1, March 2006, we have taken up this project. Though this project is not the complete implementation of the paper, it can be treated as the base work. Based on this, we have done the following study.

2. Literature Review

2.1 Feasibility Study

The proposed biometric based authentication system is feasible like other systems that are in use. Hand geometry readers measure a user's hand along many dimensions and compare those measurements to a stored image file. Rather than a high range scanner, low cost camera makes it feasible. PGM image format is used for writing programs and processing image files quickly which are useful to identify the sudden and smooth changes in the image ideal for edge detection phase.

2.1.1 PGM Format

Image file formats provide a standardized method of organizing and storing photographic and other image information. Image files are made up of either pixel or vector (geometric) data, which is rasterized to pixels in the display process, with a few exceptions in vector graphic display. The pixels that make up an image are in the form of a grid of columns and rows. Each pixel in an image consists of numbers representing brightness and color.

Grayscale or greyscale digital image is an image in which the value of each pixel is a single sample i.e. it carries the full (and only) information about its intensity.

A PGM(Portable Gray Map) image represents a grayscale graphic image description that are represented as an array of arbitrary integers. It is the lowest common denominator grayscale file format i.e. one value per pixel instead of 3 (r,g,b). The only difference in the header section is the magic identifiers which are "P2" and "P5", these correspond to the ASCII and binary form of the data respectively with no compression.

There are many pseudo-PGM formats designed to be extremely easy to learn and write programs. One official variant of PGM is the transparency mask. A transparency mask in Netpbm is represented by a PGM image, except that in place of pixel intensities, there are opaqueness values. Libnetpbm C subroutine library can be used to accurately read and interpret the format.

A PGM file consists of a sequence of one or more PGM images. There are no data, delimiters, or padding before, after, or between images.

Each PGM image consists of the following:

1. pgm image's magic number is of two characters "P5"for identifying the file type.
2. White space (blanks, TABs, CRs, LFs).
3. A width, formatted as ASCII characters in decimal.
4. White space.
5. A height, again in ASCII decimal.
6. White space.
7. The maximum gray value (Maxval), again in ASCII decimal. Must be less than 65536, and more than zero.
8. A single white space character (usually a new line).

9. A raster of Height rows, in order from top to bottom. Each row consists of Width gray values, in order from left to right. Each gray value is a number from 0 through Maxval, with 0 being black and Maxval being white. Each gray value is represented in pure binary by either 1 or 2 bytes. If the Maxval is less than 256, it is 1 byte. Otherwise, it is 2 bytes. The most significant byte is first.

   A row of an image is horizontal. A column is vertical. The pixels in the image are square and contiguous.

10. Each gray value is a number proportional to the intensity of the pixel, adjusted by the ITU-R Recommendation BT.709 gamma transfer function.
11. A common variation on the PGM format is to have the gray value be "linear," i.e. takes a PGM variant as input and produces a true PGM as output.
12. In the transparency mask variation on PGM, the value represents opaqueness. It is proportional to the fraction of intensity of a pixel that would show in place of an underlying pixel. Generally white represents total opaqueness and black represents total transparency.
13. Strings starting with "#" may be comments

There is actually another version of the PGM format that is fairly rare: "plain" PGM format. The format specified above is generally considered the normal one known as the "raw" PGM format.

The difference in the plain format is:

- There is exactly one image in a file.
- Each pixel in the raster is represented as an ASCII decimal number (of arbitrary size).

- Each pixel in the raster has white space before and after it. There must be at least one character of white space between any two pixels, but there is no maximum.
- No line should be longer than 70 characters.

The files are smaller and many times faster to read and write. Note that this raw format can only be used for MAXVAL less than or equal to 255. If you use the PGM library and try to write a file with a larger MAXVAL, it will automatically fall back on the slower but more general plain format.

2.1.2 Image Noise

Image noise is a random, usually unwanted, fluctuation of pixel values in an image. Image noise during image acquisition (digitization) can originate in film grain, or as electronic noise in the input device (scanner or digital camera) or due to environmental conditions. For instance, in acquiring images CCD camera, light level and temperature are major factors affecting the amount of noise in the resulting image.

2.1.3 Image Segmentation

Image Segmentation refers to the process of partitioning a digital image into multiple regions (sets of pixels). The goal of segmentation is to simplify and change the representation of an image to analyze easily. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images.The result of image segmentation is a set of regions that collectively cover the entire image, or a set of contours extracted from the image.

Each of the pixels in a region are similar with respect to some characteristic or computed property, such as color, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s). There are many techniques availabile for image segmentation and they vary in complexity, power and area of application. Thresholding techniques are those that determine the threshold value based on certain criteria. All pixels with values less than t are given one color while those greater than or equal to t are given another.

2.1.4 Performance Parameters:

There are some basic methods most commonly used to measure the performance of biometric technologies:

- False Reject Rate (FRR)
- False Accept Rate (FAR)
- Crossover Error Rate (CER)

2.1.4.1 FRR (False Reject Rate)

The False Reject Rate, also known as a Type I error, measures the number of times an authorized user is wrongly refused access to the protected system. False rejects are often caused by insufficient data provided by the user. This can be caused by smudges on a finger scanner, incorrect positioning of the hand or finger, incorrect alignment of the retina or iris.

2.1.4.2 FAR (False Accept Rate)

The False Accept Rate, also known as a Type I1 error, measures the number of times an unauthorized user is accepted and therefore wrongly admitted to the protected system thus enabling a security breach. This type of error is usually of higher importance for biometric access control.

2.1.4.3 CER (Crossover Error Rate)/ EER (Equal Error Rate)

The Crossover Error Rate, sometimes known as the Equal Error Rate, is where the FRR and the FAR are equal. The CER measure is sometimes used to identify system accuracy.

2.2 Current Status of the Problem

Existing systems measure hand and finger geometry using a digital camera by placing your hand on a flat surface, aligning your fingers against several pegs to ensure an accurate reading. Constraints are enforced against positioning of the hand and on the environment. Then, a camera takes one or more pictures of the hand and uses this information to determine the length, width, thickness and curvature of your hand or fingers. The existing system needs a uniform background for a touch free technique. Some of the difficulties with the existing system are cluttered background, illumination variations, hand pose, finger bending and appearance of rings.

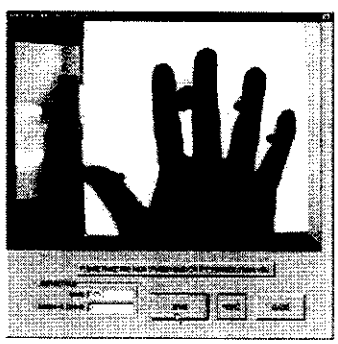


Figure 2.1 Existing geometry system pegs

Fig 2.2 Existing System against a Uniform background

To overcome these difficulties, systems have to be proposed to remove the requirement for pegs and a portable, easy camera for acquisition of hand images in real time.

2.3 Proposed System and its Advantages

In the proposed system, a passive method for verifying the identity of a person is given. A camera, no more than three feet, captures a very high-resolution photograph when the hand is kept if front of it. This process takes only one to two seconds and provides the details of the finger geometry that are mapped, recorded and stored for future matching/verification in a database. Here the dynamic method of verification is done with the help of Biometric concept.

2.3.1 Advantages

- Biometric characteristics of the individual are not easily transferable and are unique of every person, and cannot be lost, stolen or broken.
- Every individual's hand is shaped differently than another individual's hand and the shape of the person's hand does not significantly change over the course of time.
- Hand geometry technology posses one of the smallest reference templates in the biometric field, generally under ten bytes.
- These devices can take each point on the boundary line of the persons hand and fingers and process them in a short time taking into account no natural and environmental surface details, such as lines, scars, dirt, and fingernails
- Increasing speed of operation, reliability and accuracy, small template size, ease of integration into an existing system, and user-friendliness.

- The user is not obliged to place his/her hand on a surface and generally there are less constraints regarding the placement of the hand (e.g., using pegs) or the environment (e.g., uniform background).

Common applications include access control and time-and-attendance operations in healthcare solutions, parking lot, cash vault and dual custody applications and Interactive Kiosks, borrowing library books, cashless canteen systems, vending machines, class attendance and payments in schools. Hand geometry is very reliable when combined with other forms of identification, such as identification cards or personal identification numbers.

2.4 Hardware Requirement

Processor	:	Intel Processor IV
RAM	:	128 MB
Hard disk	:	20 GB
CD drive	:	40 x Samsung
Floppy drive	:	1.44 MB
Monitor	:	15' Samtron color
Keyboard	:	108 mercury keyboard
Mouse	:	Logitech mouse

2.5 Software Specification

Operating System – Windows XP/2000

Language used – J2sdk1.4.0(java)

2.6 Software Overview

The front end is designed and executed with the J2SDK1.4.0 handling the core java part with User interface Swing component. Java is robust, object oriented, multi-threaded, distributed, secure and platform independent language. It has wide variety of package to implement our requirement and number of classes and methods can be utilized for programming purpose. These features make the programmer's to implement to require concept and algorithm very easier way in Java.

The features of Java as follows:

- Core java contains the concepts like Exception handling, Multithreading; Streams can be well utilized in the project environment.
- The Exception handling can be done with predefined exception and has provision for writing custom exception for our application.
- Garbage collection is done automatically, so that it is very secure in memory management.
- The user interface can be done with the Abstract Window tool Kit and also Swing class. This has variety of classes for components and containers. We can make instance of these classes and this instances denotes particular object that can be utilized in our program.
- Event handling can be performed with Delegate Event model. The objects are assigned to the Listener that observe for event, when the event takes place the corresponding methods to handle that event will be called by Listener which is in the form of interfaces and executed.
- This application makes use of ActionListener interface and the event click event gets handled by this. The separate method actionPerformed() method contains details about the response of event.
- Java also contains concepts like Remote method invocation; Networking can be useful in distributed environment.
- This software is developed as a portable component hence it can be installed and used easily.

DETAILS OF METHODOLOGY

3. Details of the Methodology Employed

Each subject was asked to place his/her hand in front of his/her face with the inner of the palm facing the camera. We have found that this posture is the most convenient for the users, is interpreted unambiguously and provides the best resolution of the hand in the images.

The system was optimized for an access control application scenario. These are mainly areas that cannot be reached by the projected light (e.g., the sides of the fingers) and/or are highly refractive (e.g., painted finger nails and rings). Using the above setup, a hand image database was compiled and used for conducting the experiments. A group of 60 volunteer's hands were recorded for our purpose. The recording was supervised mainly because it was difficult for the users to place their hands inside the limited working volume of the sensor without any feedback. In the future we plan to provide such a feedback automatically.

3.1 Input Image:

The captured color image has more datavalues which makes manipulation on the 24 bit colour image very difficult. The gray leveling is done to reduce the processing data values into 256 color combinations between black and white. The way of applying the fractional velues for R,G,B should be in decreasing order. Hence it is converted to pgm image format where the image is represented as a array of rows and columns which is used for writing programs easily .

Here is an example of a small image in the plain PGM format.

```
P2
24 7
15
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 3 3 3 3 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 15 0
0 3 3 3 0 0 0 7 7 7 0 0 0 11 11 11 0 0 0 15 15 15 15 0
0 3 0 0 0 0 0 7 0 0 0 0 0 11 0 0 0 0 0 15 0 0 0 0
0 3 0 0 0 0 0 7 7 7 7 0 0 11 11 11 11 0 0 15 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

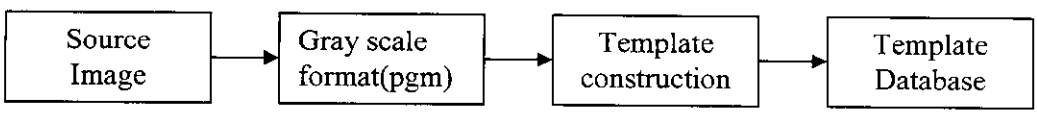
The gray scale pgm images are kept in the data folder in my application. The proposed algorithm is invariant of the camera calibration. The image is passed through four stages before extracting feature points.

There are two main phases according to user's point of view.

- Training phase
- Recognition phase

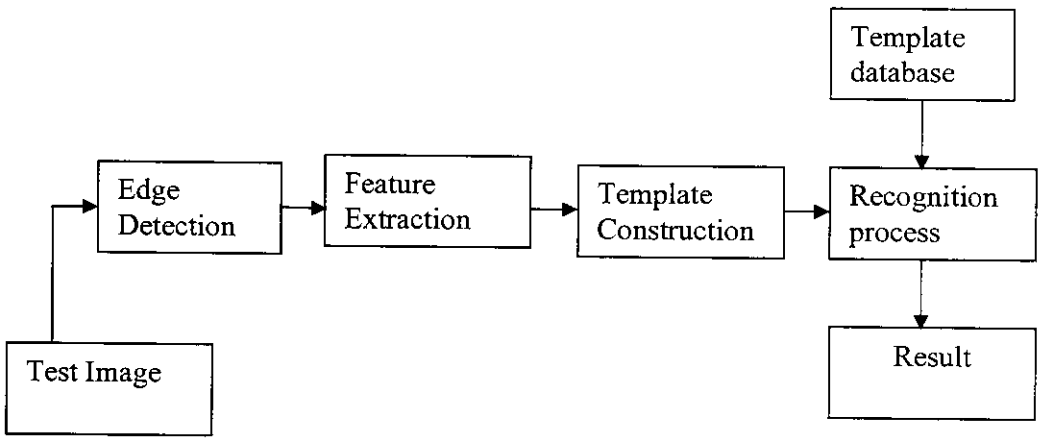
3.2. Training Phase

- Input a persons previously captured hand image
- All the features of the hand are extracted from the image
- Template is created after extraction and stored in a database.



3.3 Recognition Phase

- Input a image that is captured
- Hand image is segmented from the background using thresholding technique
- Geometry points of the hand are separated using edge detection method
- The template that is created with the extracted features is enlarged to fix boundary points
- This is compared with the templates stored in the database and if a match is found, the result is displayed.



The first step is the segmentation of the hand from the body, which is achieved using simple thresholding to separate the hand from the background. Detection and segmentation of the hand is based on a more elaborate scheme that relies on statistical modeling of the hand, arm and head plus torso points in 3-D space.

3.3.1 Thresholding Method of segmentation

The thresholding hand has two main purposes:

- To segment the image from the background
- To remove noise or unwanted information

Thresholding is the simplest method of image segmentation. Individual pixels in a grayscale image are marked as "object" pixels if their value is greater than some threshold value (assuming an object to be brighter than the background) and as "background" pixels otherwise. Typically, an object pixel is given a value of "1" while a background pixel is given a value of "0.".The key parameter in thresholding is obviously the choice of the threshold. Several different methods for choosing a threshold exist. The simplest method would be to choose the mean or median value as threshold, the rationale being that if the object pixels are brighter than the background, they should also be brighter than the average.

3.3.2 Edge Detection:

The image segmented is passed to second stage, where finger geometry points are extracted by linear edge detection. Edge detection aims at identifying the points in a digital image at which the image brightness changes sharply or more formally has discontinuites during image processing within the areas of feature detection and feature extraction. The purpose of detecting sharp changes in image brightness is to capture important events and changes in properties of image.

The result of applying an edge detector to an image may lead to a set of connected curves that indicate the boundaries of objects, the boundaries of surface markings as well curves that correspond to discontinuities in surface orientation. Thus, applying an edge detector to an image may significantly reduce the amount of data to be processed and may therefore filter out information that may be regarded as less relevant, while preserving the important structural properties of an image.

3.3.3 Boundary points enlargement

Then image with the extracted boundary points is passed to third stage, which is simple enlargement. That is, the finger boundary image is placed inside a large image in this stage. This is done in order to find/select the geometry points from the boundary points of the finger. In other words the image is enlarged so that the entire edge-image is fit in the center and geometry-lines can be projected from outer. in this proposed algorithm 3d geometry points of the segmented finger are extracted and feature points are extracted from these points.

Using this enlarged image as input, the geometry points of the finger are found. For this, a large circle is drawn around the finger in the enlarged image and from each circle point (360 degree) one line is drawn towards the center and the point (x,y) in that line where first edge pixel of the finger is found is stored.

3.3.4 Euclidean Distance:

During recognition, feature vector for test-image is found and compared with all feature vectors of training images. The Euclidean-distance between test vector and training vectors are found and the matched image is

the image with smallest distance. Accuracy is 100-r where r is the ratio between smallest and maximum distance. This process takes only one to two seconds and provides the details of the finger geometry that are mapped, recorded and stored for future matching/verification.

This software is developed to overcome the drawbacks in the existing identification methods. Here the dynamic method of verification is done with the help of Biometric concept. The real benefit is in the false-rejection rate, a measure of authenticated users who are rejected.

The main difference of our approach in comparison with the above techniques is that fewer constraints are posed on the placement of the hand and the on environment. Working on a combination of color and 3-D information, we present robust algorithms which are capable of withstanding constraints. There are certainly limitations on the working conditions under which the system may operate reliably, e.g., working outdoors or under large pose and finger bending conditions may be problematic. However, these constraints are far less than those imposed by existing systems.

PERFORMANCE EVALUATION

4. Performance Evaluation

Software testing is the process used to assess the quality of computer software. Software testing is an empirical technical investigation conducted to provide stakeholders with information about the quality of the product or serviceThey examine and change the software engineering process itself to reduce the amount of faults that end up in defect rate. Software testing can be done by software testers.

Software testing is used in association with verification and validation:

1.Verification: Have we built the software right (i.e., does it match the specification)?

2.Validation: Have we built the right software (i.e., is this what the customer wants)?

Testing can involve some or all of the following factors.

- Business requirements
- Functional design requirements
- Technical design requirements
- Regulatory requirements
- Programmer code
- Systems administration standards and restrictions
- Corporate standards
- Professional or trade association best practices
- Hardware configuration
- Cultural issues and language differences

Quality has three sets of factors - functionality, engineering, and adaptability. These three sets of factors can be taken as dimensions in the software quality space. Each dimension may be broken down into its component factors and their considerations in detail.

Functionality (exterior quality)	**Engineering (interior quality)**	**Adaptability (future quality)**
Correctness	Efficiency	Flexibility
Reliability	Testability	Reusability
Usability	Documentation	Maintainability
Integrity	Structure	

Table 4.Typical Software Quality Factors

Good testing provides measures for all relevant factors. Software testing answers questions that development testing and code reviews can't.

- Does it really work as expected?
- Does it meet the users' requirements?
- Is it what the users expect?
- Do the users like it?
- Is it compatible with our other systems?
- How does it perform?
- How does it scale when more users are added?
- Which areas need more work?
- Is it ready for release?

4.1 Unit Testing

A series of stand-alone tests are conducted during Unit Testing. Each test examines an individual component that is new or has been modified. A unit test is also called a module test because it tests the individual units of code that comprise the application. Each test validates a single module that, based on the technical design documents, was built to perform a certain task with the expectation that it will behave in a specific way or produce specific results.

Unit tests focus on functionality and reliability, Unit testing is done in a test environment prior to system integration. If a defect is discovered during a unit test, the severity of the defect will dictate whether or not it will be fixed before the module is approved.

4.2 System Testing

System Testing tests all components and modules that are new, changed, affected by a change, or needed to form the complete application. The system test may require involvement of other systems but this should be minimized as much as possible to reduce the risk of externally-induced problems.

Testing the interaction with other parts of the complete system comes in Integration Testing. The system testing is validating and verifying the functional design specification and seeing how all the modules work together.

4.3 Integration Testing

Integration testing examines all the components and modules that are new, changed, affected by a change, or needed to form a complete system. Where system testing tries to minimize outside factors, integration testing requires involvement of other systems and interfaces with other applications, including those owned by an outside vendor, external partners, or the customer.

4.4 Implementation Details

The main form is created with JFrame that contains other component like label, text box, picture box and buttons. This provision enables the user to browse through the test images from the folder called \data\test. Also the data folder contains temp, test and train. The system has already processed the images in the Train folder.

The application would have gained the knowledge on the training process as follows: *Unsupervised learning* - this is learning from observation and discovery. The data mining system is supplied with objects but no classes are defined so it has to observe the examples and recognize patterns (i.e. class description) by itself. This system results in a set of class descriptions, one for each class discovered in the environment. Again this is similar to cluster analysis as in statistics.

When the user clicks the Browse button, the JOpenDialog opens the file dialog and user can choose the image of finger which is available in the form of PGM. From this user select the image and clicks OK button.The image is displayed in the Input image frame. The provision is also given to

view the data base images by clicking the button View DB. Using the Prev and Next button the user can scan the entire data base.

To verify the image of the user, he has to click the Recognize button. While clicking the Recognize button the system calls the corresponding classes and executed. The algorithm is called for the already stored images and the name of the image to which gets matched. The matched image will be displayed in the Matched image frame along with the efficiency factor displayed in %.

Finger geometry scanners have a 1-2% false-rejection rate. FRRs are typically set to a figure between **2%** and 5%.FARs are usually set to a figure between 0.5% and 0.1%. CER of 2% is more accurate than a system with a CER of 5%.

CONCLUSION

5.Conclusion

The objective of this work was to investigate hand geometry features to achieve a better, reliable and cost benefit scheme. The results obtained demonstrate this context using a simple image acquisition setup and a small database. The method described is automated, inherently simple and tailored for its practical usage. The achieved results are significant since the two biometric traits were derived from the same image. Our results also show that the matching level achieves better performance. The hand geometry based authentication systems offer least user transaction time. The performance of the system with transformed features was compared with the raw features and it was experimentally shown that the performance of the authentication greatly improved

FUTURE ENHANCEMENTS

6. FUTURE ENHANCEMENT

A practical personal feature extraction method, combinable with other personal traits provides authentication. This point-based matching enables the sophisticated, time-consumptive and noise-sensitive finger geometry texture matching to be omissible and brings about a robust and real-time processing of less than one second. We expect that this noncontacting personal feature extraction method will have considerable utility in the practical authentication in combination with the hand geometry, palm vascular pattern, and/or facial processing.

Some other enhancements that have to be implemented are to capture a video and convert them into desirable picture format enables more security. Further examinations for more subjects and for the efficient acquisition device construction are necessary.

APPENDICES

//FingerAuthentication.java

```
import java.lang.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.filechooser.*;
import javax.imageio.*;

public class FingerAuthentication extends JFrame implements ActionListener
{
	//mainform declarations
	String strTitle="Finger Authentication";
	JFrame frmMain=new JFrame("Personal Authentication using 3D Finger
Geometry");
	JLabel lblPath=new JLabel("Path:");
	JTextField txtPath=new JTextField(Globals.TestPath+"\\test1.pgm");
	JButton btBrowse=new JButton("Browse");
	JButton btRecognize=new JButton("Recognize");
	JButton btRefresh=new JButton("Refresh");
	JButton btViewDB=new JButton("View DB");
	JButton btTrain=new JButton("Train");
	JButton btSeparator1=new JButton("");
	MyPictureBox pbInput=new MyPictureBox();
	MyPictureBox pbMatched=new MyPictureBox();
	JTextArea txtResult=new JTextArea("");
	JScrollPane spResult=new JScrollPane(txtResult);

	public FingerAuthentication()
	{
		//mainform definitions
		frmMain.setResizable(false);
		frmMain.setBounds(50,50,480,395);
		frmMain.getContentPane().setLayout(null);
		frmMain.setLocationRelativeTo(null);
		frmMain.setDefaultCloseOperation(EXIT_ON_CLOSE);

		lblPath.setBounds(17,15,100,20);
		frmMain.getContentPane().add(lblPath);
		txtPath.setBounds(50,15,frmMain.getWidth()-295,20);
```

```
        txtPath.setEditable(false);
        frmMain.getContentPane().add(txtPath);

        btBrowse.setBounds(frmMain.getWidth()-237,15,100,20);
        btBrowse.addActionListener(this);
        frmMain.getContentPane().add(btBrowse);
        btRecognize.setBounds(frmMain.getWidth()-127,15,100,20);
        btRecognize.addActionListener(this);
        frmMain.getContentPane().add(btRecognize);

        btRefresh.setBounds(frmMain.getWidth()-127,60,100,20);
        btRefresh.addActionListener(this);
        frmMain.getContentPane().add(btRefresh);

        btViewDB.setBounds(frmMain.getWidth()-127,85,100,20);
        btViewDB.addActionListener(this);
        frmMain.getContentPane().add(btViewDB);

        btTrain.setBounds(frmMain.getWidth()-127,120,100,20);
        btTrain.addActionListener(this);
        frmMain.getContentPane().add(btTrain);

        btSeparator1.setBounds(15,45,frmMain.getWidth()-40,3);
        btSeparator1.setEnabled(false);
        frmMain.getContentPane().add(btSeparator1);

        pbInput.setBounds(15,60,155,155);
        pbInput.setTitle("Input-Image");
        frmMain.getContentPane().add(pbInput);

        pbMatched.setBounds(180,60,155,155);
        pbMatched.setTitle("Matched-Image");
        frmMain.getContentPane().add(pbMatched);

        txtResult.setEditable(false);
        spResult.setBounds(15,230,320,120);
        spResult.setColumnHeaderView(new JLabel("Result"));
        frmMain.getContentPane().add(spResult);

        Globals.initialize();
        frmMain.setVisible(true);
}

//events
public void actionPerformed(ActionEvent evt)
{
```

```
            if(evt.getSource()==btBrowse) //browse for inputimage
            {
                JFileChooser tFileChooser=new JFileChooser(Globals.DataPath);
                tFileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
                int tResult=tFileChooser.showOpenDialog(this);

                if(tResult==JFileChooser.APPROVE_OPTION)
                {
                    txtPath.setText(tFileChooser.getSelectedFile().getPath());
                    PGM tpgm=new PGM(txtPath.getText());
                    pbInput.setImage(tpgm.getBufferedImage());
                }
            }
            else if(evt.getSource()==btRecognize) //process
            {
                if(new File(txtPath.getText()).exists()==false) return;
                PGM tpgm=new PGM(txtPath.getText());
                pbInput.setImage(tpgm.getBufferedImage());
                GeometryMatching geom=new GeometryMatching();
                RecognitionResult tresult=geom.recognize(txtPath.getText());
                pbMatched.setImage(tresult.matched.getBufferedImage());
                String strResult="Matched:"+new
File(tresult.matchedpath).getName()+"\r\n";
                strResult+="Match%: "+tresult.accuracy;
                txtResult.setText(strResult);
            }
            else if(evt.getSource()==btRefresh)
            {
                pbInput.revalidate();
                pbMatched.revalidate();
            }
            else if(evt.getSource()==btViewDB)
            {
                new ViewDatabase();
            }
            else if(evt.getSource()==btTrain)
            {
                Training t=new Training();
                t.train();
            }
    }

    public static void main(String args[])
    {
        new FingerAuthentication();
    }
```

```
}
```

//Circle.java

```
import java.lang.*;
import java.io.*;
import java.awt.*;
import java.util.*;

public class Circle
{
	private ArrayList points=new ArrayList();

	//constructors
	public Circle()
	{
		clear();
	}

	//get functions
	public ArrayList getPoints()
	{
		return(points);
	}

	public Point getPoint(int tindex)
	{
		Point pt=(Point)points.get(tindex);
		return(pt);
	}

	//internal methods
	private void addPoint(int x,int y)
	{
		boolean found=false;
		for(int t=0;t<size();t++)
		{
			Point pt=getPoint(t);
			if(x==pt.x&&y==pt.y)
			{
				found=true;
				break;
			}
		}
```

```
        if(found==false)
        {
                Point pt=new Point(x,y);
                points.add(pt);
        }
}

private void circlePoints(int cx,int cy,int x,int y)
{
        if(x==0)
        {
                addPoint(cx,cy+y);
                addPoint(cx,cy-y);
                addPoint(cx+y,cy);
                addPoint(cx-y,cy);
        }
        else if(x==y)
        {
                addPoint(cx+x,cy+y);
                addPoint(cx-x,cy+y);
                addPoint(cx+x,cy-y);
                addPoint(cx-x,cy-y);
        }
        else if(x<y)
        {
                addPoint(cx+x,cy+y);
                addPoint(cx-x,cy+y);
                addPoint(cx+x,cy-y);
                addPoint(cx-x,cy-y);
                addPoint(cx+y,cy+x);
                addPoint(cx-y,cy+x);
                addPoint(cx+y,cy-x);
                addPoint(cx-y,cy-x);
        }
}

//methods
public int size()
{
        return(points.size());
}

public void clear()
{
        points.clear();
```

```
    }

    public void findPoints(int xcenter,int ycenter,int radius)
    {
        int r2=radius*radius;
        int x,y,p;

        x=0;
        y=radius;
        p=(5-radius*4)/4;

        circlePoints(xcenter,ycenter,x,y);
        while(x<y)
        {
            x+=1;
            if(p<0)
            {
                p+=2*x+1;
            }
            else
            {
                y-=1;
                p+=2*(x-y)+1;
            }
            circlePoints(xcenter,ycenter,x,y);
        }
    }
}
```

//GeometryLine.java

```
import java.lang.*;
import java.io.*;
import java.awt.*;
import java.util.*;

public class GeometryLine
{
    private ArrayList points=new ArrayList();

    //constructors
    public GeometryLine()
    {
        clear();
```

```
}

//get functions
public ArrayList getPoints()
{
        return(points);
}

public Point getPoint(int tindex)
{
        Point pt=(Point)points.get(tindex);
        return(pt);
}

//internal methods
private void addPoint(int x,int y)
{
        boolean found=false;
        for(int t=0;t<size();t++)
        {
                Point pt=getPoint(t);
                if(x==pt.x&&y==pt.y)
                {
                        found=true;
                        break;
                }
        }

        if(found==false)
        {
                Point pt=new Point(x,y);
                points.add(pt);
        }
}

//methods
public int size()
{
        return(points.size());
}

public void clear()
{
        points.clear();
}
```

```
	public void findPoints(int x0,int y0,int x1,int y1)
	{
		int dx,dy;
		double m,b;

		dx=x1-x0;
		dy=y1-y0;
		addPoint(x0,y0);

		if(Math.abs(dx)>Math.abs(dy)) //slope<1
		{
			m=(double)dy/(double)dx; //compute slope
			b=y0-m*x0;
			dx=dx<0?-1:1;
			while(x0!=x1)
			{
				x0+=dx;
				addPoint(x0,(int)(m*x0+b));
			}
		}
		else
		{
			if(dy!=0) //slope>=1
			{
				m=(double)dx/(double)dy; //compute slope
				b=x0-m*y0;
				dy=dy<0?-1:1;
				while(y0!=y1)
				{
					y0+=dy;
					addPoint((int)(m*y0+b),y0);
				}
			}
		}
	}

	//static methods
	public static int findDistance(int x1,int y1,int x2,int y2)
	{
		double d=Math.sqrt(Math.pow((double)(x2-
x1),2.0)+Math.pow((double)(y2-y1),2.0));
		return((int)d);
	}
}
```

//GeometryMatching.java

```
import java.lang.*;
import java.io.*;
import java.awt.*;
import java.util.*;

public class GeometryMatching
{
	//private ArrayList features=new ArrayList();

	//constructors
	public GeometryMatching()
	{ }

	private int findSimilarity(ArrayList arr1,ArrayList arr2)
	{
		int tcount=arr1.size();
		if(arr2.size()<tcount) tcount=arr2.size();

		int tsum=0;
		for(int t=0;t<tcount;t++)
		{
			int val1=((Integer)arr1.get(t)).intValue();
			int val2=((Integer)arr2.get(t)).intValue();
			int diff=(val1-val2);
			tsum+=diff*diff;
		}
		int sim=(int)Math.sqrt(tsum);
		return(sim);
	}


	public RecognitionResult recognize(String testpath)
	{
		Process p=new Process();
		ArrayList testfeature=p.findFeature(testpath);

		//find matched image
		int smallest=-1,matchedindex=-1,max=-1;
		for(int t=0;t<Globals.features.size();t++)
		{
			ArrayList trainfeature=(ArrayList)Globals.features.get(t);
			int sim=findSimilarity(testfeature,trainfeature);
			//System.out.println(Globals.fnames[t]+": "+sim);

			if(t==0)
			{
```

```
                    smallest=sim;
                    matchedindex=0;
                    max=sim;
                }
                else
                {
                    if(smallest>sim)
                    {
                        smallest=sim;
                        matchedindex=t;
                    }
                    if(max<sim) max=sim;
                }
            }

            //find matched imagepath
            String matchedfname=Globals.fnames[matchedindex];
            System.out.println("matched: "+matchedfname);

            RecognitionResult tresult=new RecognitionResult();
            tresult.matchedpath=matchedfname;
            tresult.matched=new PGM(matchedfname);
            tresult.diff=smallest;
            tresult.accuracy=100.0-(((double)smallest/(double)max)*100.0);
            return(tresult);
        }
}
```

//Globals.java

```
import java.lang.*;
import java.io.*;
import javax.swing.filechooser.*;
import java.awt.*;
import java.util.*;

public class Globals
{
    public static String DataPath="data";
    public static String TrainPath="data\\train";
    public static String TestPath="data\\test";
    public static String TempPath="data\\temp";
    public static boolean TempFlag=true;
```

```
public static int TrainCount=68;
public static int SizeX=128,SizeY=128;
public static int xDiv=128,yDiv=128;

public static int BackGround=0,ForeGround=255;
public static int NW=0,N=1,NE=2,E=3,SE=4,S=5,SW=6,W=7;
public static int ENLARGEWIDTH=200,ENLARGEHEIGHT=200;
public static ArrayList features=new ArrayList();
public static String fnames[]=new String[Globals.TrainCount];

//system methods
public static void initialize()
{
        TrainCount=findImageCount();
}

//file methods
public static int findImageCount()
{
        FileSystemView fv=FileSystemView.getFileSystemView();
        File files[]=fv.getFiles(new File(TrainPath),true);

        int tcount=0;
        for(int t=0;t<files.length;t++)
        {
                String tfname=files[t].getName();
                String ext=getExtensionFromFileName(tfname);
                if(ext.compareToIgnoreCase("pgm")==0) tcount+=1;
        }

        return(tcount);
}

public static String getExtensionFromFileName(String tpath)
{
        int tpos=tpath.lastIndexOf(".");
        String ext=tpos==-1?"":tpath.substring(tpos+1);
        return(ext);
}

public static int findRandom(int max)
{
        double trandom=Math.random();
        int ans=(int)(max*trandom);
        return(ans);
}
```

```
	//pgm utils
	public static PGM resize(PGM imgin,int wout,int hout)
	{
		PGM imgout=new PGM(wout,hout);

		//resize algorithm (linear)
		int win=imgin.getColumns(),hin=imgin.getRows();
		for(int r=0;r<imgout.getRows();r++)
		{
			for(int c=0;c<imgout.getColumns();c++)
			{
				int xi=c,yi=r;
				int ci=(int)(xi*((double)win/(double)wout));
				int ri=(int)(yi*((double)hin/(double)hout));
				int inval=imgin.getPixel(ri,ci);
				int outval=inval;
				imgout.setPixel(yi,xi,outval);
			}
		}

		return(imgout);
	}
}
```

//HandDetection.java

```
import java.lang.*;
import java.io.*;
import java.awt.*;
import java.util.*;

public class HandDetection
{
	//segment finger from background (thresholding)
	public static PGM thresh(PGM imgin)
	{
		//find mean
		int tsum=0;
		for(int r=0;r<imgin.getRows();r++)
		{
			for(int c=0;c<imgin.getColumns();c++)
			{
```

```
                        tsum+=imgin.getPixel(r,c);
                    }
                }
                double
tavg=((double)tsum)/((double)imgin.getRows()*(double)imgin.getColumns());
                int tval=(int)(tavg*2);

                //remove background pixels
                PGM imgout=new PGM(imgin.getColumns(),imgin.getRows());
                for(int r=0;r<imgout.getRows();r++)
                {
                        for(int c=0;c<imgout.getColumns();c++)
                        {
                                int inval=imgin.getPixel(r,c);
                                int outval=inval>=140?255:0;
                                imgout.setPixel(r,c,outval);
                        }
                }

                return(imgout);
        }

        //find finger boundary pixels (edge detection)
        public static PGM findboundary(PGM imgin,int maxdiff)
        {
                PGM imgout=new PGM(imgin.getColumns(),imgin.getRows());

                //initialize
                for(int r=0;r<imgout.getRows();r++)
                {
                        for(int c=0;c<imgout.getColumns();c++)
                        {
                                imgout.setPixel(r,c,255);
                        }
                }

                for(int r=0;r<imgout.getRows();r++)
                {
                        for(int c=0;c<imgout.getColumns();c++)
                        {
                                int inval=imgin.getPixel(r,c); //get current pixel

                                int tinval[]=new int[2];
                                tinval[0]=imgin.getPixel(r,c+1); //get right pixel
                                tinval[1]=imgin.getPixel(r+1,c); //get down pixel
```

```
                        //determine if there is an edge
                        boolean flag1=java.lang.Math.abs(inval-
tinval[0])>maxdiff?true:false;
                        boolean flag2=java.lang.Math.abs(inval-
tinval[1])>maxdiff?true:false;
                        boolean fval=flag1||flag2;

                        int outval=fval==true?0:255;
                        if(c!=imgout.getColumns()-1&&r!=imgout.getRows()-1)
imgout.setPixel(r,c,outval);
                }
        }

        return(imgout);
    }

    //enlarge image to find finger geometry boundary points
    public static PGM enlarge(PGM imgin,int newwidth,int newheight)
    {
        PGM imgout=new PGM(newwidth,newheight);
        int startc=(newwidth-imgin.getColumns())/2;
        int startr=(newheight-imgin.getRows())/2;

        //initialize
        for(int r=0;r<imgout.getRows();r++)
        {
            for(int c=0;c<imgout.getColumns();c++)
            {
                imgout.setPixel(r,c,255);
            }
        }

        //fit to center
        for(int r=0;r<imgin.getRows();r++)
        {
            for(int c=0;c<imgin.getColumns();c++)
            {
                int inval=imgin.getPixel(r,c);
                imgout.setPixel(startr+r,startc+c,inval);
            }
        }

        return(imgout);
    }

    //plot geometry points to image
```

```
	public static PGM plot(PGM imgin,ArrayList points)
	{
		PGM imgout=new PGM(imgin.getColumns(),imgin.getRows());

		//initialize to background
		for(int r=0;r<imgout.getRows();r++)
		{
			for(int c=0;c<imgout.getColumns();c++)
			{
				imgout.setPixel(r,c,255);
			}
		}

		//plot the points
		for(int t=0;t<points.size();t++)
		{
			Point pt=(Point)points.get(t);
			imgout.setPixel(pt.y,pt.x,0);
		}

		return(imgout);
	}
}
```

//MyPictureBox.java

```
import java.lang.*;
import java.io.*;
import java.awt.*;
import java.awt.image.*;
import javax.swing.*;

public class MyPictureBox extends JPanel
{
	//members
	private String Title;
	private BufferedImage Picture;

	private JScrollPane tScrollPane=new JScrollPane();
	private JLabel tLabel=new JLabel();

	//internal members
	private int TitleHeight=22;
```

```
//constructor
public MyPictureBox()
{
	Title="";
	Picture=null;
	this.setLayout(null);
}

//get functions
public String getTitle()
{
	return(Title);
}

public BufferedImage getImage()
{
	return(Picture);
}

//set functions
public void setTitle(String tTitle)
{
	Title=tTitle;
	repaint();
}

public void setImage(BufferedImage tPicture)
{
	Picture=tPicture;
	repaint();
}

//default methods
protected void paintComponent(Graphics g)
{
	super.paintComponent(g);

	try
	{
		//draw title
		if(Title.equals("")==false)
		{
			g.setColor(new Color(128,0,0));
			g.setFont(new Font("Verdana",Font.BOLD,12));
			g.drawString(Title+":",4,16);
		}
```

```
                    //draw image
                    if(Picture!=null)
                    {
                            tLabel.setIcon(new ImageIcon(Picture));
                            tScrollPane.setBounds(1,TitleHeight+1,getWidth()-
2,getHeight()-TitleHeight-2);
                            tScrollPane.getViewport().add(tLabel);
                            this.add(tScrollPane);
                    }

                    //draw border
                    g.setColor(new Color(0,0,0));
                    g.drawRect(0,0,getWidth()-1,getHeight()-1);
                    g.drawLine(0,TitleHeight,getWidth()-1,TitleHeight);
            }
            catch(Exception err)
            {
                    System.out.println("Error: "+err);
                    System.exit(-1);
            }
    }
}
```

//PGM.java

```
import java.lang.*;
import java.io.*;
import java.awt.*;
import java.awt.image.*;

public class PGM
{
    private String FilePath;

    //pgm imageheader
    private String Type;
    private String Comment;
    private int Columns,Rows,MaxGray;

    //pgm imagedata
    private int[][] Pixels;

    //constructors
    public PGM()
```

```
{
        FilePath="";
        Type="";
        Comment="";
        Columns=0;
        Rows=0;
        MaxGray=0;
        Pixels=null;
}

public PGM(String tpath)
{
        FilePath=tpath;
        readImage();
}

public PGM(int tColumns,int tRows)
{
        FilePath="";
        Type="P5";
        Comment="";
        MaxGray=255;
        setDimension(tColumns,tRows);
}

//get functions
public String getFilePath()
{
        return(FilePath);
}

public String getType()
{
        return(Type);
}

public String getComment()
{
        return(Comment);
}

public int getColumns()
{
        return(Columns);
}
```

```
public int getRows()
{
        return(Rows);
}

public int getMaxGray()
{
        return(MaxGray);
}

public int getPixel(int tr,int tc)
{
        return(tr<0||tr>Rows-1||tc<0||tc>Columns-1?0:Pixels[tr][tc]);
}

public int[][] getNeighbor(int tr,int tc)
{
        int neighbor[][]=new int[3][3];

        neighbor[0][0]=getPixel(tr-1,tc-1); //NW
        neighbor[0][1]=getPixel(tr-1,tc  ); //N
        neighbor[0][2]=getPixel(tr-1,tc+1); //NE
        neighbor[1][0]=getPixel(tr  ,tc-1); //W
        neighbor[1][1]=getPixel(tr  ,tc  );
        neighbor[1][2]=getPixel(tr  ,tc+1); //E
        neighbor[2][0]=getPixel(tr+1,tc-1); //SW
        neighbor[2][1]=getPixel(tr+1,tc  ); //S
        neighbor[2][2]=getPixel(tr+1,tc+1); //SE

        return(neighbor);
}

public int getNeighbor(int tr,int tc,int direction)
{
        int pval=0;
        if    (direction==Globals.NW) pval=getPixel(tr-1,tc-1);
        else if(direction==Globals.W ) pval=getPixel(tr  ,tc-1);
        else if(direction==Globals.SW) pval=getPixel(tr+1,tc-1);
        else if(direction==Globals.S ) pval=getPixel(tr+1,tc  );
        else if(direction==Globals.SE) pval=getPixel(tr+1,tc+1);
        else if(direction==Globals.E ) pval=getPixel(tr  ,tc+1);
        else if(direction==Globals.NE) pval=getPixel(tr-1,tc+1);
        else if(direction==Globals.N ) pval=getPixel(tr-1,tc  );
        return(pval);
}
```

```
//set functions
public void setFilePath(String tFilePath)
{
        FilePath=tFilePath;
}

public void setType(String tType)
{
        Type=tType;
}

public void setComment(String tComment)
{
        Comment=tComment;
}

public void setDimension(int tColumns,int tRows)
{
        Rows=tRows;
        Columns=tColumns;
        Pixels=new int[Rows][Columns];
}

public void setMaxGray(int tMaxGray)
{
        MaxGray=tMaxGray;
}

public void setPixel(int tr,int tc,int tval)
{
        if(tr<0||tr>Rows-1||tc<0||tc>Columns-1) return;
        else Pixels[tr][tc]=tval;
}

//methods
public void readImage()
{
        try
        {
                FileInputStream fin=new FileInputStream(FilePath);

                int c;
                String tstr;

                //read first line of ImageHeader
                tstr="";
```

```
c=fin.read();
tstr+=(char)c;
c=fin.read();
tstr+=(char)c;
Type=tstr;

//read second line of ImageHeader
c=fin.read(); //read Lf (linefeed)
c=fin.read(); //read '#'
tstr="";
boolean iscomment=false;
while((char)c=='#') //read comment
{
        iscomment=true;
        tstr+=(char)c;
        while(c!=10&&c!=13)
        {
                c=fin.read();
                tstr+=(char)c;
        }
        c=fin.read(); //read next '#'
}
if(tstr.equals("")==false)
{
        Comment=tstr.substring(0,tstr.length()-1);
        fin.skip(-1);
}

//read third line of ImageHeader
//read columns
tstr="";
if(iscomment==true) c=fin.read();
tstr+=(char)c;
while(c!=32&&c!=10&&c!=13)
{
        c=fin.read();
        tstr+=(char)c;
}
tstr=tstr.substring(0,tstr.length()-1);
Columns=Integer.parseInt(tstr);

//read rows
c=fin.read();
tstr="";
tstr+=(char)c;
while(c!=32&&c!=10&&c!=13)
```

```
                {
                        c=fin.read();
                        tstr+=(char)c;
                }
                tstr=tstr.substring(0,tstr.length()-1);
                Rows=Integer.parseInt(tstr);

                //read maxgray
                c=fin.read();
                tstr="";
                tstr+=(char)c;
                while(c!=32&&c!=10&&c!=13)
                {
                        c=fin.read();
                        tstr+=(char)c;
                }
                tstr=tstr.substring(0,tstr.length()-1);
                MaxGray=Integer.parseInt(tstr);

                //read pixels from ImageData
                Pixels=new int[Rows][Columns];
                for(int tr=0;tr<Rows;tr++)
                {
                        for(int tc=0;tc<Columns;tc++)
                        {
                                c=(int)fin.read();
                                setPixel(tr,tc,c);
                        }
                }

                fin.close();
        }
        catch(Exception err)
        {
                System.out.println("Error: "+err);
                System.exit(-1);
        }
}

public void writeImage()
{
        try
        {
                FileOutputStream fout=new FileOutputStream(FilePath);

                //write image header
```

```
                //write PGM magic value 'P5'
                String tstr;
                tstr="P5"+"\n";
                fout.write(tstr.getBytes());

                //write comment
                Comment=Comment+"\n";
                //fout.write(comment.getBytes());

                //write columns
                tstr=Integer.toString(Columns);
                fout.write(tstr.getBytes());
                fout.write(32); //write blank space

                //write rows
                tstr=Integer.toString(Rows);
                fout.write(tstr.getBytes());
                fout.write(32); //write blank space

                //write maxgray
                tstr=Integer.toString(MaxGray);
                tstr=tstr+"\n";
                fout.write(tstr.getBytes());

                for(int r=0;r<Rows;r++)
                {
                        for(int c=0;c<Columns;c++)
                        {
                                fout.write(getPixel(r,c));
                        }
                }

                fout.close();
        }
        catch(Exception err)
        {
                System.out.println("Error: "+err);
                System.exit(-1);
        }
}

public void writeImageAs(String tFilePath)
{
        PGM imgout=new PGM(getColumns(),getRows());

        for(int r=0;r<getRows();r++)
```

```
            {
                for(int c=0;c<getColumns();c++)
                {
                    imgout.setPixel(r,c,getPixel(r,c));
                }
            }

            imgout.setFilePath(tFilePath);
            imgout.writeImage();
        }

        public BufferedImage getBufferedImage()
        {
            BufferedImage timg=new
BufferedImage(Columns,Rows,BufferedImage.TYPE_INT_RGB);

            for(int r=0;r<getRows();r++)
            {
                for(int c=0;c<getColumns();c++)
                {
                    int tgray=getPixel(r,c);
                    int trgb=getRGBValue(tgray,tgray,tgray);
                    timg.setRGB(c,r,trgb);
                }
            }

            return(timg);
        }

        //static methods
        public static int getRGBValue(int tred,int tgreen,int tblue)
        {
            return((tred<<16)+(tgreen<<8)+tblue);
        }
}
```

//Process.java

```
import java.lang.*;
import java.io.*;
import java.awt.*;
import java.util.*;

public class Process
{
	//private ArrayList features=new ArrayList();


	//constructors
	public Process()
	{
		//
	}

	//internal methods
	private ArrayList findGeometry(PGM imgin)
	{
		//find circlepoints
		Circle circle=new Circle();
		int
midx=Globals.ENLARGEWIDTH/2,midy=Globals.ENLARGEHEIGHT/2;
		circle.findPoints(midx,midy,90);
		ArrayList circlepoints=circle.getPoints();

		//find lines from center to each circlepoint
		ArrayList ans=new ArrayList();
		for(int t=0;t<circlepoints.size();t++)
		{
			Point circlept=(Point)circlepoints.get(t);
			GeometryLine line=new GeometryLine();
			line.findPoints(midx,midy,circlept.x,circlept.y);
			ArrayList linepoints=line.getPoints();

			//mark face boundary
			for(int j=linepoints.size()-1;j>=0;j--)
			{
				Point linept=(Point)linepoints.get(j);
				int inval=imgin.getPixel(linept.y,linept.x);
				if(inval==0)
				{
					ans.add(linept);
					break;
```

```
                        }
                    }
                }

                return(ans);
        }

        private ArrayList findDistances(ArrayList tpoints)
        {
                ArrayList tarr=new ArrayList();
                int
midx=Globals.ENLARGEWIDTH/2,midy=Globals.ENLARGEHEIGHT/2;
                for(int t=0;t<tpoints.size();t++)
                {
                        Point pt=(Point)tpoints.get(t);
                        int dist=GeometryLine.findDistance(pt.x,pt.y,midx,midy);
                        tarr.add(new Integer(dist));
                }
                return(tarr);
        }

        public ArrayList findFeature(String tpath)
        {
                PGM pgmin=new PGM(tpath);
                PGM pgm1=HandDetection.thresh(pgmin);
                PGM pgm2=HandDetection.findboundary(pgm1,4);
                PGM
pgm3=HandDetection.enlarge(pgm2,Globals.ENLARGEWIDTH,Globals.ENLARGEHE
IGHT);
                ArrayList geometrypoints=findGeometry(pgm3);
                PGM pgm4=HandDetection.plot(pgm3,geometrypoints);

                //write intermediate images
                if(Globals.TempFlag==true)
                {
                        //pgm1.writeImageAs(Globals.TempPath+"\\1_thres_["+new
File(tpath).getName()+"].pgm");
                        //pgm2.writeImageAs(Globals.TempPath+"\\2_boundary_["+new
File(tpath).getName()+"].pgm");
                        //pgm3.writeImageAs(Globals.TempPath+"\\3_enlarge_["+new
File(tpath).getName()+"].pgm");
                        pgm4.writeImageAs(Globals.TempPath+"\\4_geometry_["+new
File(tpath).getName()+"].pgm");
                }

                ArrayList tfeature=findDistances(geometrypoints);
```

```
                for(int t=0;t<2;t++) tfeature.add(new Integer(Globals.findRandom(10)));
                return(tfeature);
        }
}
```

//RecognitionResult.java

```
import java.lang.*;
import java.io.*;

public class RecognitionResult
{
        public String matchedpath;
        public PGM matched;
        public double diff;
        public double accuracy;
}
```

//GeometryMatching.java

```
import java.lang.*;
import java.io.*;
import java.awt.*;
import java.util.*;

public class Training
{
        //private ArrayList features=new ArrayList();
        //private String fnames[]=new String[Globals.TrainCount];

        //constructors
        public Training()
        {
                //
        }

        //methods
        public void train()
        {
                int tindex=0;
                System.out.println("Training...");
                for(int t=0;t<Globals.TrainCount;t++)
                {
                        String fname=Globals.TrainPath+"\\"+(t+1)+".pgm";
```

```
                    Globals.fnames[tindex]=fname;
                    System.out.println(fname);
                    Process p=new Process();
                    ArrayList tfeature=p.findFeature(fname);
                    Globals.features.add(tfeature);
                    tindex+=1;
                }
        }
}
```

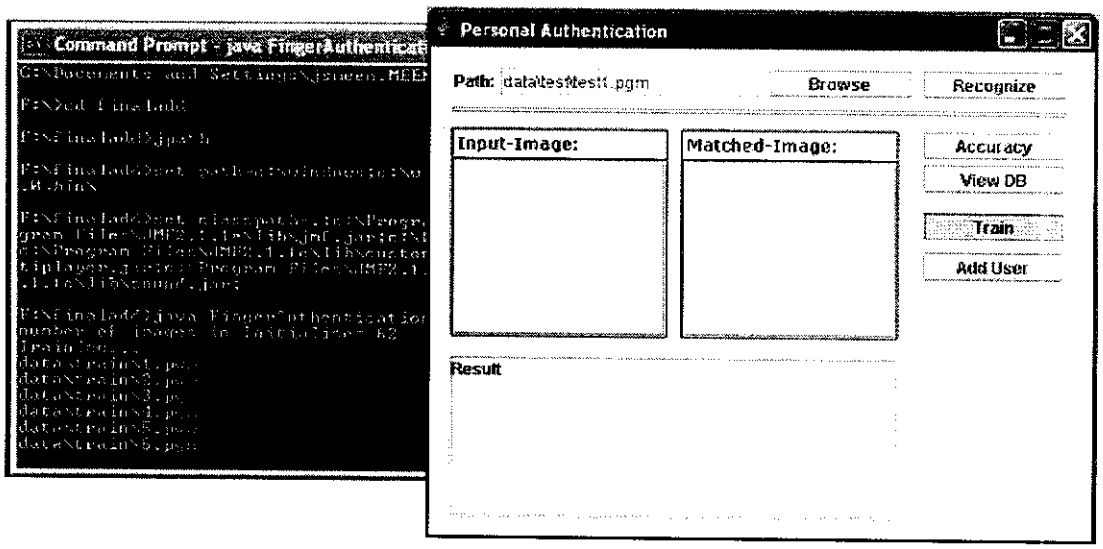
//ViewDatabase.java

```
import java.lang.*;
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import java.awt.image.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.filechooser.*;
import javax.imageio.*;

public class ViewDatabase implements ActionListener
{
        //mainform declarations
        String strTitle="View Database";
        JFrame frmMain=new JFrame("View Database");
        MyPictureBox pbMain=new MyPictureBox();
        JButton btRefresh=new JButton("Refresh");
        JButton btPrevious=new JButton("Prev");
        JButton btNext=new JButton("Next");

        //system declarations
        private int CurrentIndex=1;

        public ViewDatabase()
        {
                //mainform definitions
                frmMain.setResizable(false);
                frmMain.setBounds(50,50,280,285);
                frmMain.getContentPane().setLayout(null);
                frmMain.setLocationRelativeTo(null);

                pbMain.setBounds(15,15,frmMain.getWidth()-40,frmMain.getHeight()-
85);
```

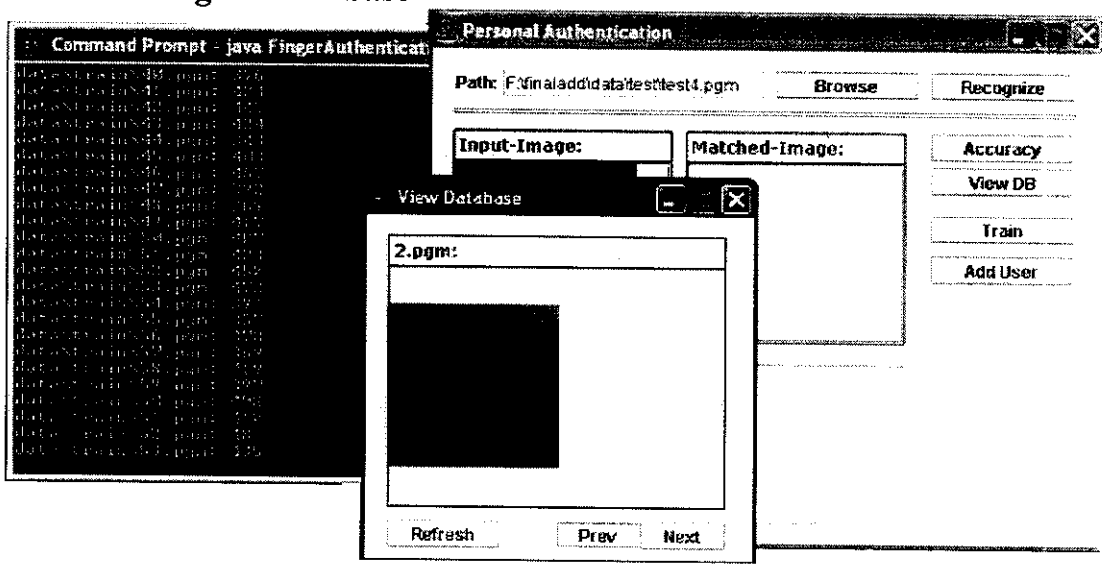
```
            frmMain.getContentPane().add(pbMain);

            btRefresh.setBounds(15,frmMain.getHeight()-60,80,20);
            btRefresh.addActionListener(this);
            frmMain.getContentPane().add(btRefresh);

            btPrevious.setBounds(frmMain.getWidth()-145,frmMain.getHeight()-
60,60,20);
            btPrevious.addActionListener(this);
            frmMain.getContentPane().add(btPrevious);

            btNext.setBounds(frmMain.getWidth()-85,frmMain.getHeight()-
60,60,20);
            btNext.addActionListener(this);
            frmMain.getContentPane().add(btNext);

            ShowCurrentImage();
            frmMain.setVisible(true);
        }

        //events
        public void actionPerformed(ActionEvent evt)
        {
            if(evt.getSource()==btNext)
            {
                CurrentIndex+=1;
                if(CurrentIndex>Globals.TrainCount)
                {
                    CurrentIndex=1;
                }
            }
            else if(evt.getSource()==btPrevious)
            {
                CurrentIndex-=1;
                if(CurrentIndex<1)
                {
                    CurrentIndex=Globals.TrainCount;
                }
            }

            ShowCurrentImage();
        }

        //private methods
        private void ShowCurrentImage()
        {
```

```
                String strPath=Globals.TrainPath+"\\"+CurrentIndex+".pgm";
                pbMain.setTitle(new File(strPath).getName());
                if(new File(strPath).exists()==true)
                {
                        PGM tpgm=new PGM(strPath);
                        PGM resized=Globals.resize(tpgm,120,120);
                        pbMain.setImage(resized.getBufferedImage());
                }
        }
}
```

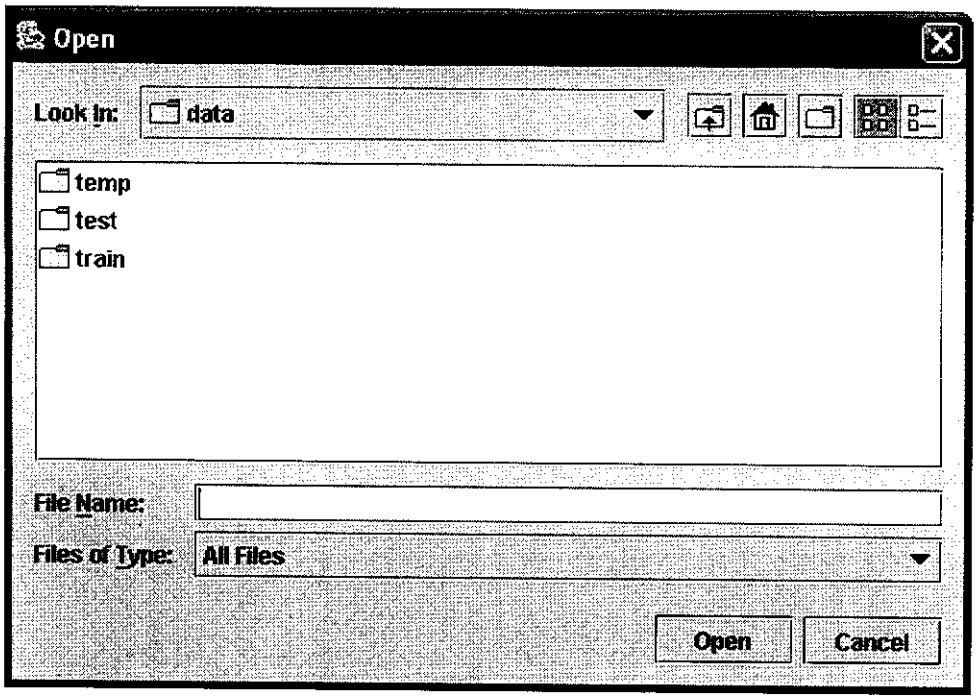
7.2.1 Training the images



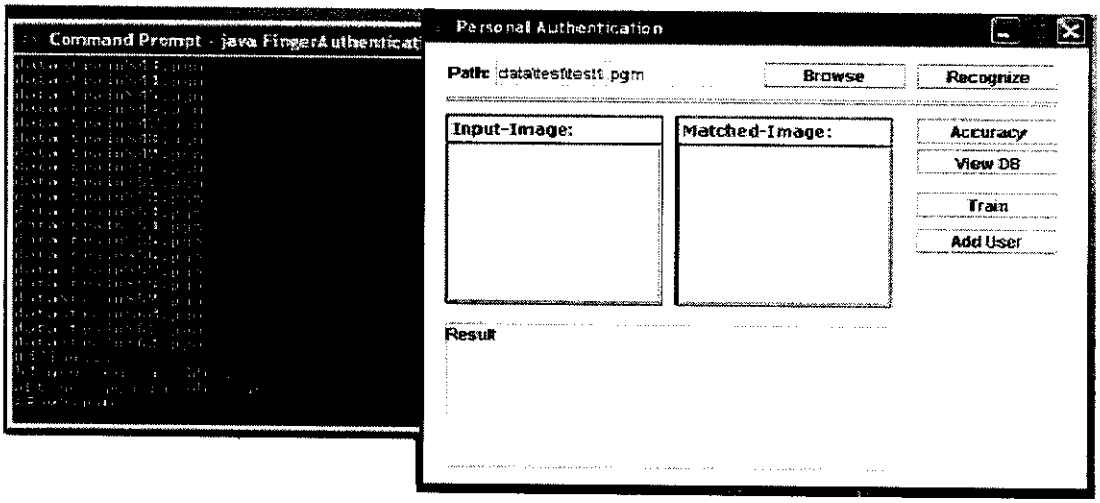
7.2.2 Viewing the database



7.2.3 Open Dialog Box:



7.2.4 Adding New user



REFERENCES

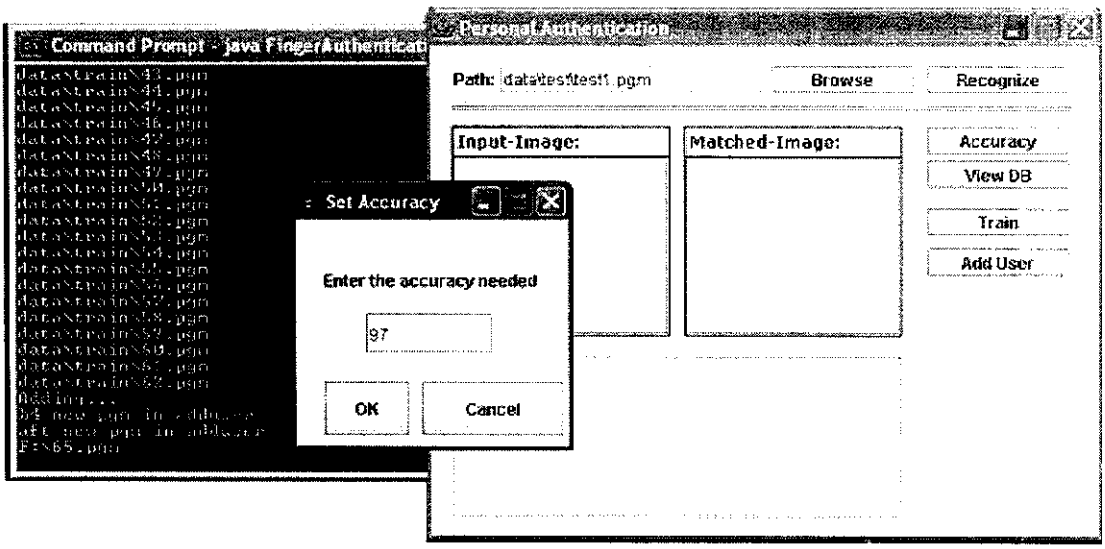
8. REFERENCES

1. Sotiris Malassiotis,Nike Aifanti and Michael G. Strintzis,"Personal Authentication Using 3-D Finger Geometry", IEEE Trans,Information Forensics and Security, Vol.1,No.1,Mar 2006.
2. Nongluk Covavisaruch, Pipat Prateepamornkul, Puripant Ruchikachorn, And Piyanaat Taksaphan, "Personal Verification And Identification Using HandGeometry", ECTI Trans,Computer and Information Technology vol.1,no.2,pp:134-140, November 2005
3. Rand Sanchez-Rclllo and Ana Gonznlez-Marcas,"Access Control System Hand Geometry Verification and Smart Cards", IEEE AES System, pp:45-48, Feb 2000
4. R. Sanchez-Reillo, C. Sanchez-Avila, and A. Gonzalez-Macros, "Biometric Identification through Hand Geometry Measurements", *IEEE Trans. PAMI*, vol. 22, no.10, pp. 1168-1171, Oct. 2000.
5. S. Prabhakar, S. Pankanti, and A. K. Jain, "Biometric recognition: Security and privacy concerns," *IEEE Security Privacy Mag.*, vol. 1, no. 2, pp. 33–42, 2003.
6. Anil K. Jain, Arun Ross and Salil Prabhakar," An Introduction to Biometric Recognition" IEEE Trans, Circuits And Systems For Video Technology, Vol. 14, No. 1, pp:4-20, Jan 2004
7. Junta Doi and Masaaki Yamanaka, "Discrete Finger and Palmar Feature Extraction for Personal Authentication" IEEE Trans, Instrumentation And Measurement, Vol. 54, No. 6, pp:2213-2219, December 2005

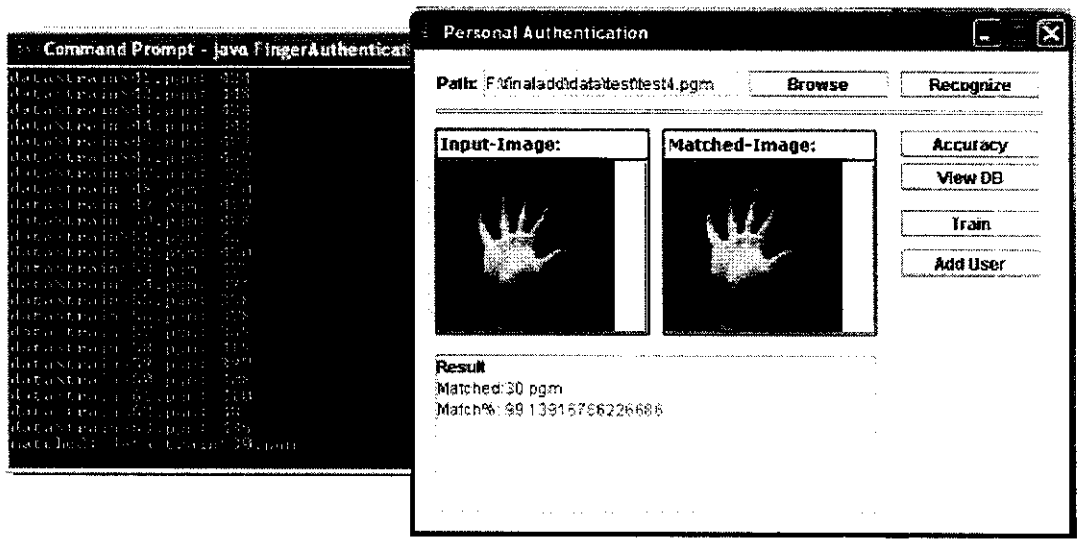
WEBSITES:

1. http://www.biometrics.cse.msu.edu
2. http://www.biometricwatch.com
3. http://en.wikipedia.org
4. http://netpbm.sourceforge.net

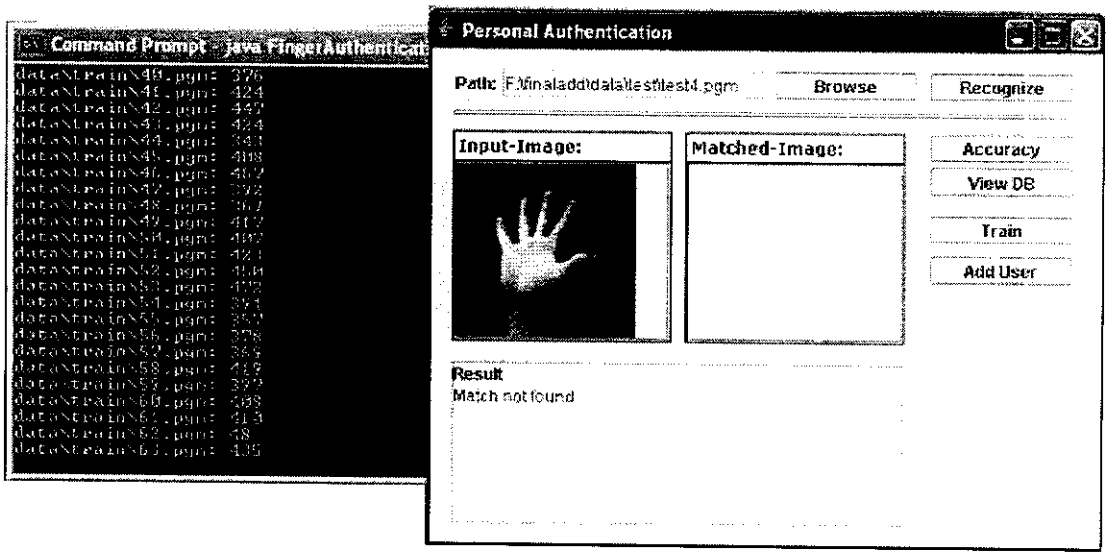
7.2.5 Setting the accuracy

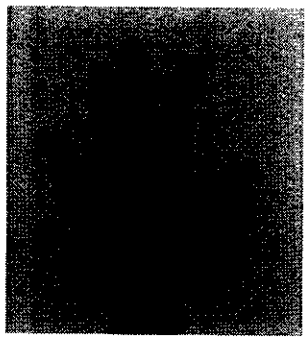


7.2.5 Matched Result



7.2.6 Match not found

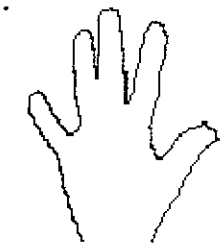




7.3.1 Input Image

7.3.2 After Thresholding

7.3.3. After Fixing Boundary points



7.3.4 Enlargement of segmented hand

7.3.5 After fixing geometry points