

FINGER PRINT RECOGNITION SYSTEM



PROJECT REPORT



P-1155

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

Bachelor of Engineering in Information Technology
Of
Bharathiar University, Coimbatore.

Submitted by

S.R.ASHOK RAJAN

0027S0071

T.JAGADEESWARAN

0027S0078

Under the guidance of

Mrs.V.VANITHA M.E

Lecturer.

DEPARTMENT OF INFORMATION TECHNOLOGY
KUMARAGURU COLLEGE OF TECHNOLOGY,
COIMBATORE – 641006.

MARCH 2004.

DEPARTMENT OF INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY

(Affiliated to Bharathiar University, Coimbatore)



CERTIFICATE

This is to certify that the project entitled
FINGER PRINT RECOGNITION SYSTEM
is done by

S.R.ASHOK RAJAN

0027S0071

T.JAGADEESWARAN

0027S0078

and submitted in partial fulfillment of the
requirement for the award of the degree of the

Bachelor of Engineering in Information Technology

Of

Bharathiar University, Coimbatore.

Professor & Head of the department

(Dr.S.THANGASAMY)

Guide

(Mrs.V.VANITHA)

Certified that the candidates were examined by us in the project work
Viva voce examination held on 26.03.04.

Internal Examiner

External Examiner

Declaration

We,

S.R.Ashok Rajan 0027S0071



T.Jagadeeswaran 0027S0078

declare that we do the project entitled “FINGERPRINT RECOGNITION SYSTEM”, and to the best of our knowledge, a similar work has not been submitted earlier to the Bharathiar University or any other institution, for fulfillment of the requirement of the course study.

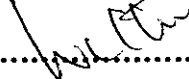
This project report is submitted on the partial fulfillment for the requirement for the awards of the degree of Bachelor of Engineering in Information Technology from Bharathiar University.

Place: Coimbatore.

Date: 25.3.04


[S. R. Ashok Rajan]

[T. Jagadeeswaran]

Project Guided by

.....


Mrs. V. Vanitha M.E.,

ACKNOWLEDGEMENT

The exhilaration achieved upon the successful completion of any task should be definitely shared with 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 having provided us excellent facilities to work with. We express our deep gratitude to our Principal Dr .K. K. Padmanabhan Ph.D., for ushering us in the path of triumph.

We are always thankful to our beloved Professor and the Head of the Department Dr.S. Thangasamy Ph.D., whose consistent support and enthusiastic involvement helped us a great deal.

We are greatly indebted to our beloved guide Mrs.V.Vanitha M.E., Lecturer, Department of Computer Science and Engineering for her excellent guidance and timely support during the course of this project. As a token of our esteem and gratitude, we honor her for her assistance towards this cause.

We also thank our project coordinator Mrs. S. Devaki M.S., and our beloved class advisor Ms. P. Sudha B.E., for their invaluable assistance.

We also feel elated in manifesting our deep sense of gratitude to all the staff and lab technicians in the Department of Computer Science and Engineering.

We feel proud to pay our respectful thanks to our Parents for their enthusiasm and encouragement and also we thank our friends who have associated themselves to bring out this project successfully.

SYNOPSIS

As the need for personal authentication increases, many people are turning to biometric authentication as an alternative to traditional security devices. An effective fingerprint recognition system is presented here for the purpose of E-attendance. The matching system consists of two main blocks: The first allows for the extraction of essential information from the reference image offline, the second performs the matching itself. The information is obtained from the reference image by minutiae extraction procedures and also the pore extraction process, which we have proposed for the further developments. The fingerprint identification using minutia based on three point matching to cope with the strong deformation of fingerprint due to static friction or finger rolling. Also proposes the further enhancement on the facility to match the pore-based recognition, which will not fail in any of the worst conditions. The verification algorithm is implemented using language 'C'. Low false reject and zero false accept error rates have been predicted based on the technique that possesses various advantages over systems employed in current world.

CONTENTS

1. Introduction	1
1.1. Existing System and Limitations	1
1.2. Proposed System and Advantages	1
2. System Requirement Analysis	3
2.1. Product Definition	3
3. Software Requirements Specification	4
3.1. Introduction	4
3.2. Specific Requirements	5
4. System Design Specification	8
4.1. Overall Design	8
4.2. Design Diagram	9
4.3. Biometrics	9
4.4. Finger Print Recognition	11
4.5. Minutiae	13
4.6. Methods for Extraction	13
4.7. Three Point Matching Algorithm	16
4.8. Functions Used	17
4.9. Algorithm	19
5. Product Testing	21
6. Future Enhancements	23
7. Conclusion	24
8. Bibliography	25
9. Appendix	26

1.Introduction

Our project is an effort to improve the existing finger print recognition systems. This system aims at extracting and recognizing fingerprint for the security purpose like bank automation, criminal identification. Here the input is taken in the form of an image file. The images are processed and the minutiae are extracted for the verification process. Output is displayed on the screen according to the threshold value.

1.1.Existing system and limitations :

The limitation of the existing systems is that those existing systems are in limited constraints in the way of obtaining the matching points and the way of calculating the co-ordinates of the reference points to verify the fingerprint and the verification process also complex because of the algorithm used.

1.2.Proposed system and advantages :

The goal of a verification system is to compare two fingerprint images: a fingerprint captured and processed in a particular moment with one fingerprint stored in the database in real time. In this work a verification system is proposed and it is shown in Fig 1.1. Two steps, an off-line and online step, compose this system. In the off-line step all fingerprints of a set of person are captured, processed and stored in a database for later use. In the on the line step a person gives its own fingerprint to verify the identity: the fingerprint is compared with ones stored in the database. In Fig1.1, it is possible to see that the phases of off-line and on-line steps are constituted by: image enhancement, minutiae extraction and

matching. The matching algorithm used here is a triangular algorithm that is based on line length and line angle between minutiae.

The applications here are,

- Financial Services
- Health Care
- Law Enforcement
- Criminal Identification
- Citizen Identification

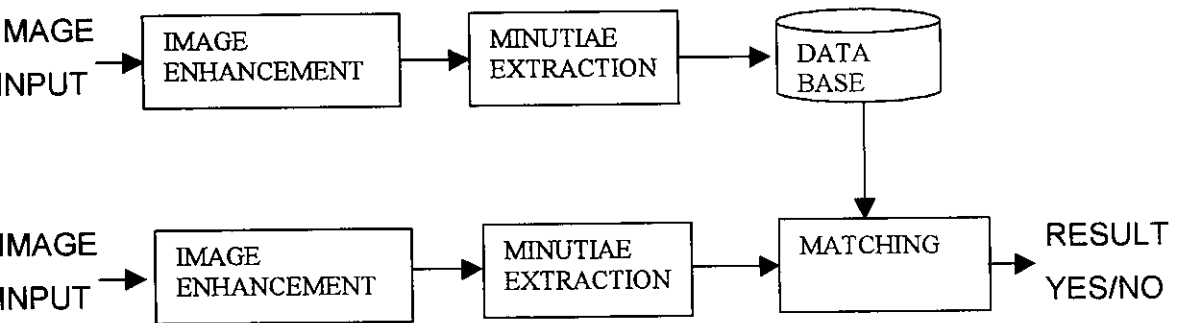


Fig 1.1. Proposed system

2. System Requirement Analysis

2.1. Product Definition

Fingerprint Recognition System extracts and recognizes fingerprint image and then compares the test image with the reference image for the security and identification purpose. Here the input is taken in the form of an image file that contains the Fingerprint. The minutiae's are extracted from the images and converted as the binary value. Three point matching algorithm is used on these image files to identify the fingerprints. The output is displayed on the screen according to the Threshold value.

3. Software Requirement Specification:

3.1. Introduction

3.1.1. Purpose

The purpose of this system is to produce an active demonstration of the use of fingerprints as neural networks where minutiae's can be stored on a database. And then a competitive analysis is performed on the fingerprint and compared to the database. The process here is to compare a fingerprint with the stored database.

3.1.2. Scope

The system will aim to produce the following results:

Demonstrate the fingerprint Verification can be used in conjunction with minutiae using Three point Matching Algorithm and Produce a demonstration on fingerprint Verification.

3.1.3. Definitions, Acronyms, Abbreviations

Definition

Biometrics:

Biometric technologies are defined as "automated methods of identifying or authenticating the identity of a living person based on a physiological or behavioral characteristic."

Three Point Matching:

This is the algorithm where the minutiae of the finger prints are extracted and compared with the testing image. The three minutiae points are the bifurcation, ridge ending, and the plain.

Acronyms and Abbreviations :

tif image	Format of fingerprint image.
GUI	Graphical User Interface.
SDD	Software Design Description.
SRS	Software Requirements Specification.
Minutiae	Points like bifurcation, ridge ending and plane for comparison.
Extraction	obtaining the minutiae points for verification and future enhancement.

3.1.4.Overview

Upon completion of the project the system will demonstrate among other things its feasibility for use in the real world. The significant components will consist of a PC, a fingerprint scanner. All these components will be integrated to produce a system with a high level of security using fingerprint verification as a means of identification.

3.2.Specific Requirements:

3.2.1.Functional Requirements

3.2.1.1.Introduction

The basic use of this is for easier and quicker retrieval of Fingerprint Verification hence decrease the amount of overhead in the other non secured security system.

3.2.1.2.List of Inputs

The Fingerprint of the user.

3.2.1.3.Information Processing Requirement

From the captured input the minutiae points on the fingerprints which are bifurcation points, ridge ending, and planes are extracted and converted as the binary information regarding with the co-ordinates of the axes x, y.

3.2.2.Performance Requirement:

3.2.2.1.Security

The system will authenticate users via their fingerprint characteristic, a template of which will be stored on the database. This template will be compared to a live scan taken at the time of user authentication. Other fields of the certificate will contain information such as the name of the user, name of the issuing authority and a digital signature of the issuing authority.

3.2.3.Design Constraints

3.2.3.1. Hardware limitations

Minimal Requirements are

- Pentium III processor and above.
- 64 MB RAM
- 10 GB Hard Disk
- 101 Keys Keyboard

3.2.3.2. User Interfaces, Screen Formats

The user interface to be designed will be user friendly so that no other professional training is required on the user part. The User interface for promoting the Fingerprint will be present and another form in which the details or various categories present in the corresponding history will be there.

3.2.3.3. Hardware Interfaces and Software Interfaces with other systems

The hardware interface to be designed provides communication between fingerprint Scanner and the PC. The software interface provides a database, which stores the required Fingerprint. The user-interface is to be designed in C.

3.2.4. Other Requirements

3.2.4.1. Operations required by the user

The Fingerprint should be unique for all the users so that the integrity is maintained.

3.2.4.2. System Reliability

The system will be designed, and tested in the laboratory with the objective of determining its feasibility for use in the real world. As a result a major factor that will contribute to the success of the project will be the reliability of the system developed. The level of security provided should be higher than that provided by conventional security systems in use today. This fact should be emphasised in the design of the security policy if the system is to be considered for commercial use.

4. System Design Specification

4.1. Overall Design:

Our system implements algorithms to extract the minutiae of the fingerprint from the image file, to recognize them and verify the output. The algorithm extracts the three points called bifurcation, plain, and the ridge end from the image file. Three point matching algorithm is used here for extracting those points. In the next module, which is the matching algorithm, the reference image minutiae points which are extracted from the previous module are compared with the test image module where the same three minutiae of the test images also extracted. The threshold values of the variances between those two images are calculated using this algorithm. The final module of the project is simply the output module where according to the threshold value the output whether matched or not matched is displayed.

4.2.Design Diagram:

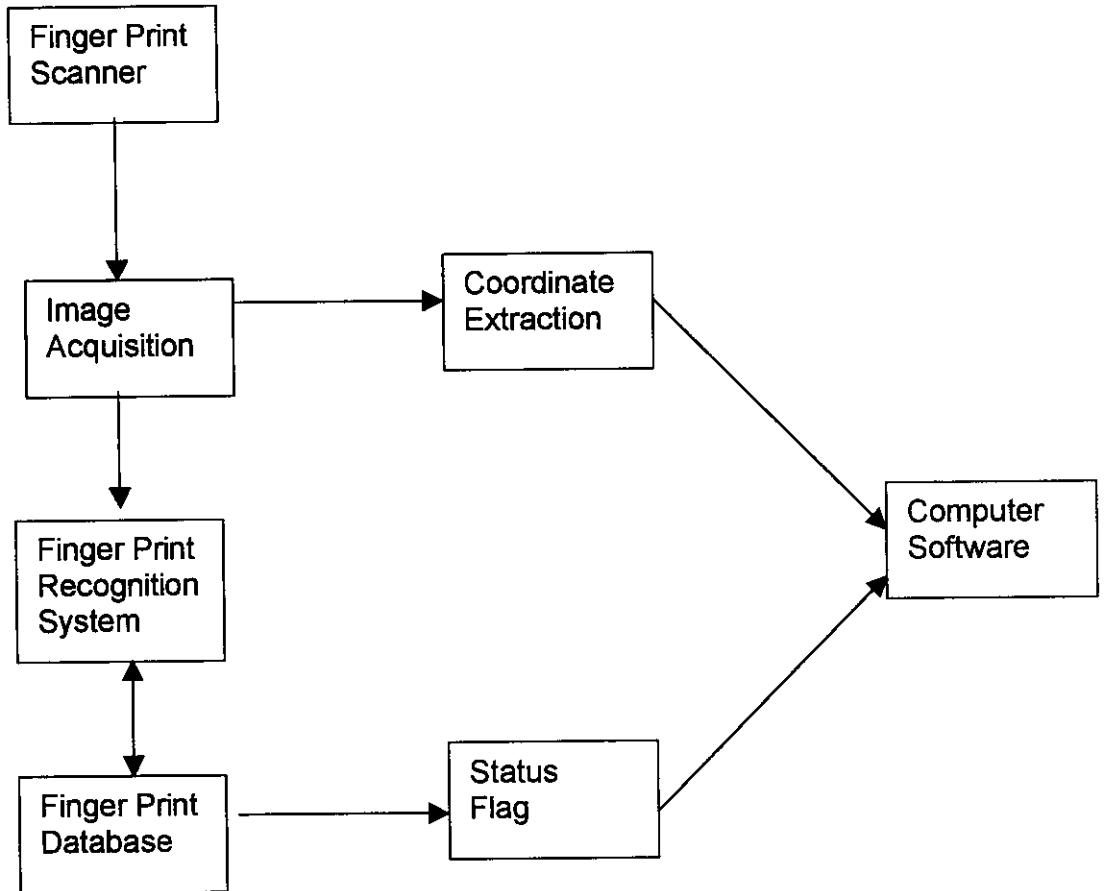


Fig.1.2.Design Diagram

4.3. BioMetrics:

Biometrics and bioinformatics are the fast developing scientific direction, studying the processes of creation, transmission, reception, storage, processing, displaying and interpretation of information in all the channels of functional and signal systems of living objects which are known to biological and medical science and practice. Modern natural sciences at present sharply need in the updating of scientific picture of the

world, and the essential contribution in this process can be made by the biometric and biomedical methods.

How is '**Biometrics**' Defined?

Biometric technologies are defined as ***"automated methods of identifying or authenticating the identity of a living person based on a physiological or behavioral characteristic."***

Because biometrics can be used in such a variety of applications, it is very difficult to establish an all-encompassing definition. The most suitable definition of biometrics is:

The automated use of physiological or behavioral characteristics to determine or verify identity.

To elaborate on this definition, physiological biometrics is based on measurements and data derived from direct measurement of a part of the human body. Finger-scan, iris-scan, retina-scan, hand-scan, and facial-scan are leading physiological biometrics. Behavioral characteristics are based on an action taken by a person. Behavioral biometrics, in turn, is based on measurements and data derived from an action, and indirectly measure characteristics of the human body. Voice-scan, 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. It is important to note that the behavioral/physiological distinction is slightly artificial. Behavioral biometrics is based in part on physiology, such as the shape of the vocal chords (voice-scan) or the dexterity of hands and fingers (signature-scan). Physiological biometric technologies are similarly informed by user behavior, such as the manner in which a user presents a finger or looks at a camera. However, the behavioral, physiological distinction is a helpful tool in understanding how biometrics work and how they can be applied in the real world.

Biometrics techniques may be classified in three categories:

- Those based on the analysis of biological traces (odor, saliva, urine, blood, DNA, etc.)
- Those based on behavioral analysis (movement of a signature line, typing on a computer keyboard, etc.)
- Those based on morphological analysis (fingerprints, shape of the hand, lines of the face, network of veins on the retina, the iris of the eye, etc.)

4.4.FINGERPRINT RECOGNITION

Due to steady increases in computing power and the advent of unobtrusive, easy-to-use fingerprint sensors, fingerprints (Fig3) are used more frequently as a biometric (identification based on a physiological or behavioral characteristic) for identification and recognition. Since fingerprints are unique, even between identical twins, they are perfect for various security uses. The primary technique for matching newly acquired prints is the extraction and matching of landmarks known as **minutiae**. Minutiae are areas where the ridges of the print either terminate to form a ridge ending, or split into two new ridges, forming a ridge bifurcation (see Fig 1.3 (a-d)).

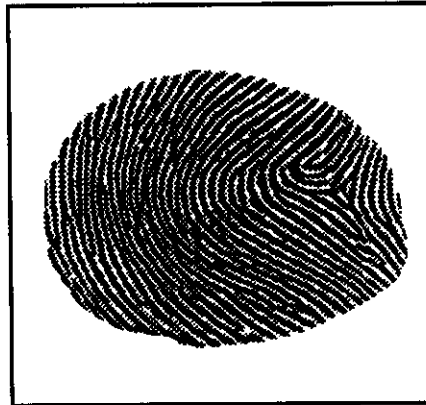


Fig 1.3: Fingerprint

Numerous procedures have been proposed for the extraction of minutiae points from fingerprint images. Most of them, however, involve extensive pre-processing of the fingerprint image. A common approach for example, involves the thinning of the ridges, also known as skeletonization or ridge extraction. The process of ridge extraction requires extensive pre-processing, which is time consuming. Each

technique improves upon the previous, and the final approach combines the results of the initial three in an attempt to classify the data through clustering.

Each technique classifies an extracted frame into one of three groups: (1) a ridge ending, (2) a bifurcation, or (3) a plain simply, no minutia point).

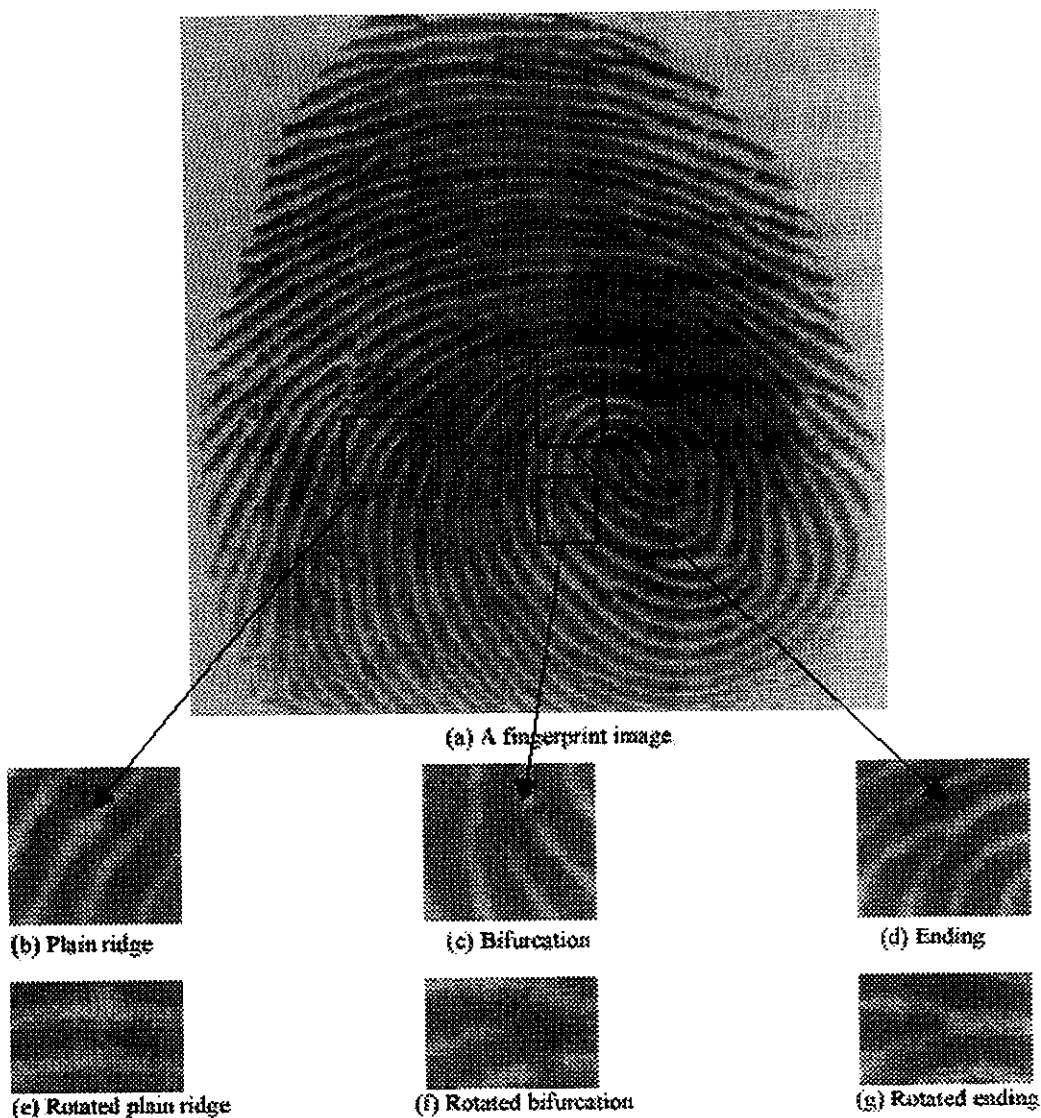


Fig 1.4 (a-d): Ridge bifurcation

The human fingerprint is comprised of various types of ridge patterns, traditionally classified according to the decades-old Henry system: left loop, right loop, arch, whorl,

and tented arch. Loops make up nearly 2/3 of all fingerprints, whorls are nearly 1/3, and perhaps 5-10% is arches.

4.5.Minutiae

Most fingerprint recognition systems make use of *minutiae*. These are local discontinuities in the *ridge-valley* pattern. In (see Fig1.4), the ridges are represented by black lines and the valleys are white. All minutiae can be described as a combination of *ridge endings* and *bifurcations*. The types, positions and orientations of the minutiae are reliable features for fingerprint matching.

In order to extract minutiae from a gray-scale fingerprint image, typical image processing transformations, like determination of the directional field, filtering, binarization, and thinning, are applied to an image. The comparison of the original minutiae's stored in the template with the minutiae's obtained from the fingerprint is done in the algorithm illustrated in Fig 1.5.

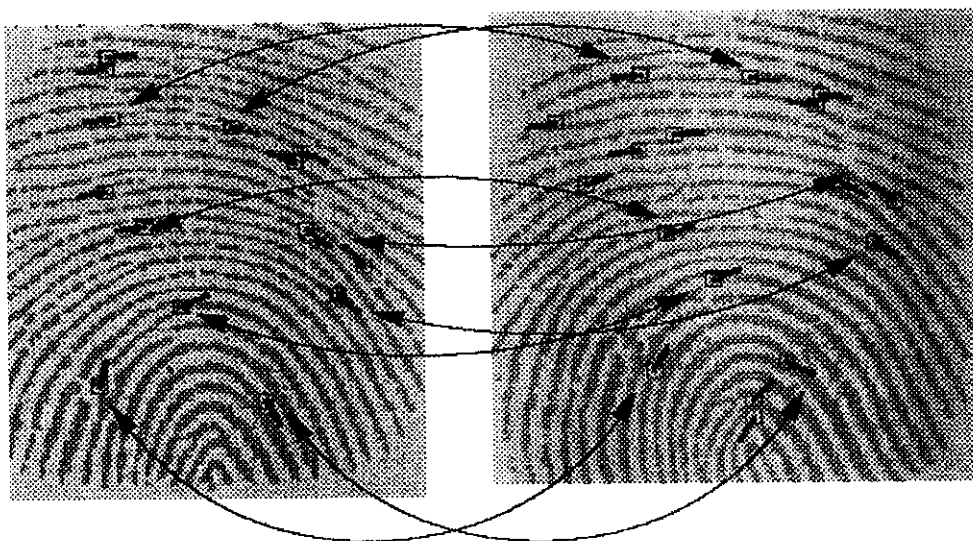


Fig 1.5. Template matching

4.6. Methods for Extraction of minutiae points

Minutiae (see Fig 1.6), the discontinuities that interrupt the otherwise smooth flow of ridges, are the basis for most finger-scan authentication. Codified in the late 1800's as Galton features, minutiae are at their most rudimentary ridge endings, the points at which a ridge stops, and bifurcations, the point at which one ridge divides into 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). Other features are essential to finger-scan authentication.

The core is the inner point, normally in the middle of the print, around which swirls, loops, or arches center. It is frequently characterized by a ridge ending and several acutely curved ridges. Deltas are the points, normally at the lower left and right hand of the fingerprint, around which a triangular series of ridges center.

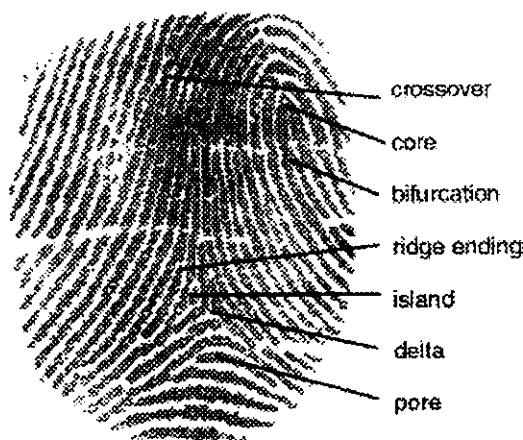


Fig 1.6. Fingerprint image

The location and distribution of the pores as a means of authentication, but the resolution required to capture pores consistently is very high. Once a high-quality image is captured, there are a several steps required to convert its distinctive features

into a compact template. This process, known as feature extraction, is at the core of finger-scan technology. The image must then be converted to a usable format. If the image is grayscale, areas lighter than a particular threshold are discarded, and those darker are made black. The ridges are then thinned from 5-8 pixels in width down to one pixel, for precise location of endings and bifurcations.

Minutiae localization begins with this processed image. At this point, even a very precise image will have distortions and false minutiae that need to be filtered out. For example, an algorithm may search the image and eliminate one of two adjacent minutiae, as minutiae are very rarely adjacent.

Anomalies caused by scars, sweat, or dirt appear as false minutiae, and algorithms locate any points or patterns that don't make sense, such as a spur on an island (probably false) or a ridge crossing perpendicular to 2-3 others (probably a scar or dirt). A large percentage of would-be minutiae are discarded in this process. The point at which a ridge ends, and the point where a bifurcation begins, are the most rudimentary minutiae, and are used in most applications.

There is variance in how exactly to situate a minutia point: whether to place it directly on the end of the ridge, one pixel away from the ending, or one pixel within the ridge ending (the same applies to bifurcation). Once the point has been situated, its location is commonly indicated by the distance from the core, with the core serving as the 0, 0 on an X, Y-axis. Some company uses the far left and bottom boundaries of the image as the axes, correcting for misplacement by locating and adjusting from the core. In addition to the placement of the minutia, the angle of the minutia is normally used.

When a ridge ends, its direction at the point of termination establishes the angle (more complicated rules can apply to curved endings). This angle is taken from a horizontal line extending rightward from the core, and can be up to 359.

In addition to using the location and angle of minutiae, some vendors classify minutia by type and quality. The advantage of this is that searches can be quicker, as a particularly notable minutia may be distinctive enough to lead to a match. A vendor can also rank high versus low quality minutia and discard the latter.

7. THREE POINT MATCHING ALGORITHM

In order to define an effective matching technique, a deformation model must be identified. When looking at different images containing the same fingerprint, a strong global deformation of the images can be observed. This distortion is due to static friction on a sensor surface, or paper, or rolling of the finger during ink-based registration techniques. However, small portions of the images do not appear to exhibit manifest distortion. The idea of how to take into account small local variations, which may lead to huge global distortion, is discussed as follows.

A pair of triangles forming a square are reported in fig 1.7. in particular fig 1.7.a. By shortening or extending some of the edges by less than 10 percent of their original length, the two images shown in fig 1.7.b and fig 1.7.c may be obtained. This small distortion makes the original form easily recognizable and may be comparable with the local distortion of a fingerprint image. In particular, the area around minutiae may be slightly distorted due to static friction caused by the fingertip touching a hard surface. By replicating the square of fig 15 several times, it is possible to create a big image.

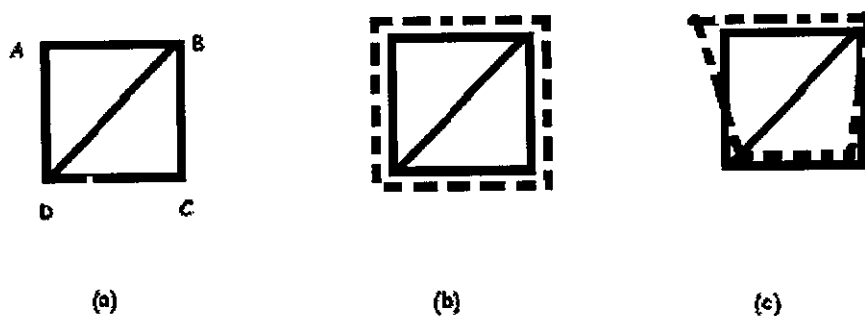


Fig 1.7.(a) Two triangles forming a square, (b) and (c) distortion of the triangles, where each edge is shortened or extended by less than 10 percent of its original length

The Three point matching algorithm is used to extract a possible minutiae set where the matching is established. The result is a number of matching minutiae pairs on the reference and test images.

Three point matching algorithm creates a connected graph on both images. The corresponding minutiae are identified. When any new minutia comes in the picture. This procedure allows for matching on a local basis, where distortion is small and can also identify two fingerprints with considerable global distortion.

4.8.Function used

- **GetNextMinutiae()** : gets the next, already not considered, reference minutiae from the ordered set.
- **GetNextTestMinutiae (ref)**: gets the next, already not considered, test minutiae corresponding to ref minutiae from the ordered set with decreasing similarity between test and reference regions.
- **LineLength (ref1, ref2, test1, test2)**: computes the length difference between the lines connecting two minutiae in the reference image and two in the test image.
- **ComputeAngle (ref1, ref2, test1, test2)**: computes the angle between two lines connecting minutiae pair ref1, ref2 and pair test1, test2, respectively.
- **AddMatchedMinutiae (ref1, ref2, test1, test2)**: adds the pairs of matched minutiae in reference image and in test image to the already matched minutiae list.
- **GetNextMinutiae (ref1, ref2, test1, test2)**: gets the next minutiae pair belonging to an already matched line both in reference and test images.
- **MinimumMatch** is the minimum number of required minutiae to match for a valid identification.
- **GetNextCloseMinutiae(ref1,ref2)**: gets the next, already not considered reference minutiae from the ordered minutiae list minimizing the function $\max(L2(\text{ref1},\text{ref2}), L2(\text{ref2},\text{ref3}))$ where $\max()$ is the larger between two real numbers and $L2()$ is the Euclidean norm on the bidimensional Cartesian plane.

- **ComputeTestCoordinates(X,Y,ref3,ref1,test1,ref2,test2):** Computes the (X,Y) coordinates corresponding to ref3 using the linear root-translation defined but the mapping ref1-> test1,ref2->test2 according to the formula:

$$X=ax_3+by_3+C_x \quad Y= -bx_3+ay_3+C_y$$

Where

$$a= ((X_2-X_1)*(x_2-x_1) + (Y_2-Y_2) (y_2-y_1))/ ((x_2-x_1) ^2+ (y_2-y_2) ^2)$$

$$b= ((X_2-X_1)*(y_2-y_1) + (Y_2-Y_2) (x_2-x_1))/ ((x_2-x_1) ^2+ (y_2-y_2) ^2)$$

$$C_x= X_1-ax_1-by_1$$

$$C_y= Y_1-ay_1+bx_1$$

With the uppercase variables referring to the test coordinate space and the lowercase coordinates to the reference coordinate space.

- **OrderTestMinutiae (ref, X, Y):** orders the test minutiae corresponding to the reference minutiae ref according to the increasing distance from point of coordinate (x, y) in the test image.
- **NumberOfMatchedMinutiae():** computes the number of already matched minutiae.
- **MaxDistance** is the maximum allowable deformation between segments.
- **MaxAngle** allows controlling the maximum rotation.

4.9. Algorithm

```
Do
    // sets two minutiae on reference image
    ref1=GetNextMinutiae();
    ref2=GetNextMinutiae();

Do
    // sets two corresponding two minutiae on test image
    test1=GetNextTestMinutiae(ref1);

Do
    test2=GetNextTestMinutiae(ref2);
    If(test1 != null)
Dist=LineLength (ref1, ref2, test1, test2);
Angle=ComputAngle (ref1, ref2, test1, test2);
    Endif
    // corresponding is valid if deformation is less than threshold
    Until (dist < MaxDistance and angle < MaxAngle) OR (test == null )

Until (test1 != null or test2 == null )
If(test1 == null or test2 == null )
    Goto end_NO_MATCH;
Endif
// adds minutiae pair to already matched list
AddMatchedMinutiae (ref1, test1, ref2, test2);
Do
    // sets third minutiae for each pair of reference and test matching minutiae
pairs // according to roto-translation defined by minutiae pairs
Validpair = GetNextNubytaePair (&ref1, &test1, &ref2, &test2);
    If (validpair)

Do
```

```

ref3=GetNextCloseMinutiae (ref1, ref2);
If (ref3 != null)
    //defines 3rd minutiae corresponding area on test image
    ComputeTestCoordinates (&X, &Y, ref3, ref1, test1, ref2, test2);
    OrderTestMinutiae (ref3, x, y);
Endif
Do
    test3=GetNextTestMinutiae (ref3);
    If(test3 != null)
        dist1=LineLength (ref1, ref3, test1, test3);
        dist2=LineLength (ref2, ref3, test2, test3);
        angle1=ComputeAngle (ref1, ref3, test1, test3);
        angle2=ComputeAngle (ref2, ref3, test2, test3);

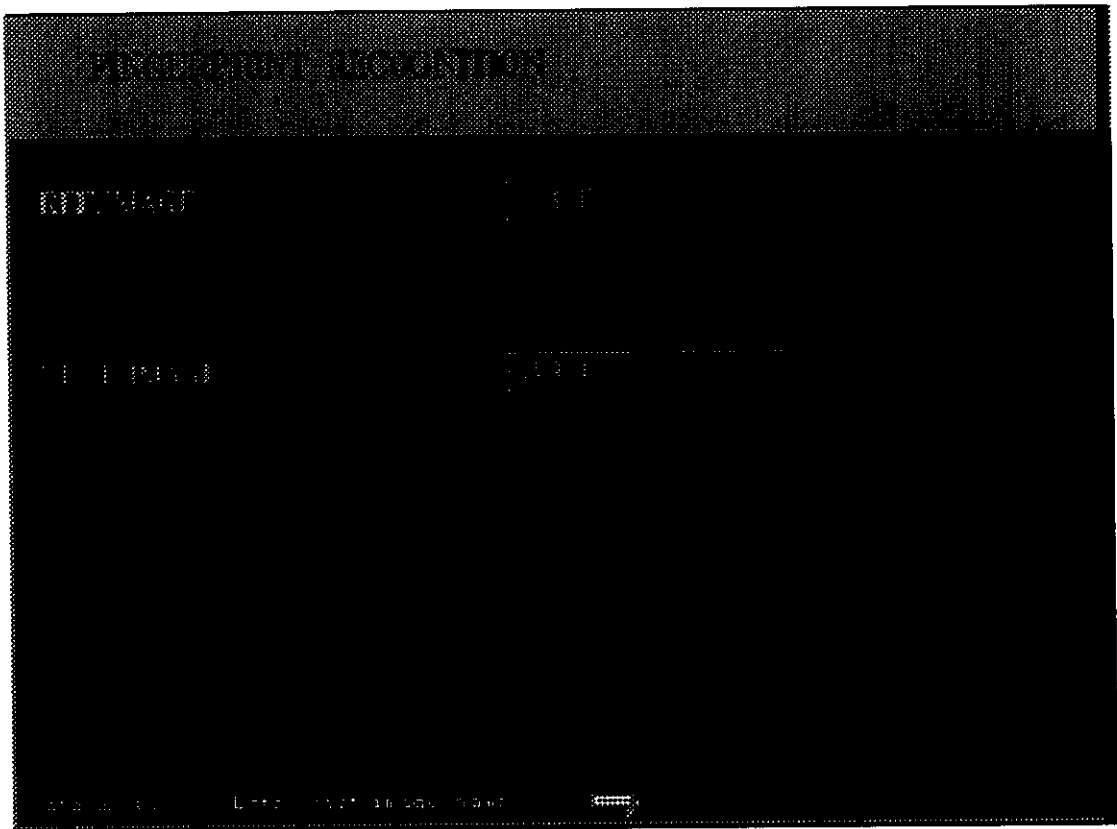
    Endif
    // corresponding is valid if deformation is less than threshold
Until ((dist1 < MaxDistance AND dist2 < MaxDistance AND angle1 < MaxAngle
AND angle2 < MaxAngle) OR (test3 == null))
    Until( test3 != null OR ref3 == null)
        If( ref3 != null AND test3 != null)
            AddMatchedMinutiae(ref1,test1,ref3,test3);
            AddMatchedMinutiae(ref2,test2,ref3,test3);
        Endif
    Endif
    Until validpair == false
Until NumberOfMatchedMinutiae() > MinimumMatch

```

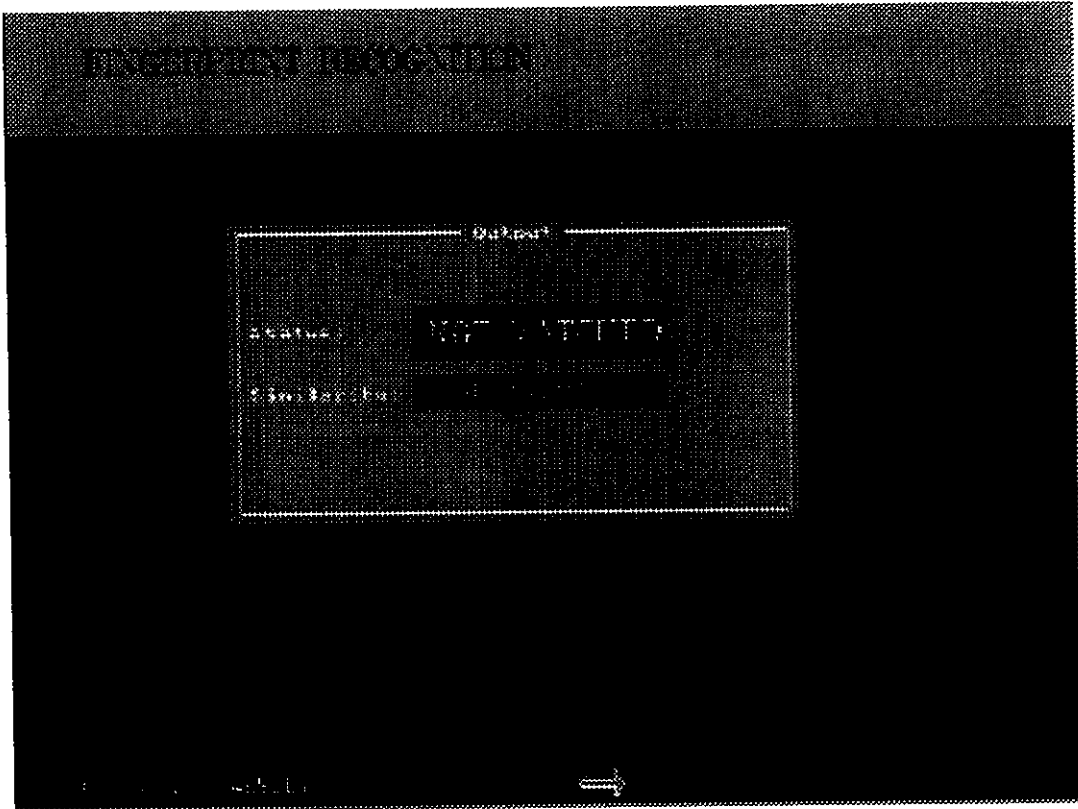
5. Product Testing

The product has been tested and the output screen of the product are shown below. Here the inputs are the two fingerprint images and the output is whether matched or not matched.

INPUT SCREEN



OUTPUT SCREEN



6. Future Enhancements

This model is applicable to all the fingerprint verification systems because the algorithm what we have approached is in the minutia based comparison is the best one and the proposed model will give you 100% accuracy. So this model is the base for any kind of fingerprint recognition system and in further it can be implemented in the real time process .

7. Conclusion

A description of a fingerprint verification system using Three point matching technique, which utilizes minutia position information, has been presented. The verification algorithm has been implemented using language 'C'. Furthermore, it has been proposed that an added measure of security may result from the inclusion of pores in the verification system. Based on the results of the work presented in this paper, further research on the use of Multi-Biometric authentication systems is warranted.

3.BIBLIOGRAPHY

- 1] Zsolt Miklos Kovacs-Vajna, "A Fingerprint Verification System Based on Three point matching and Dynamic Time Warping," IEEE Transaction on Pattern Analysis and Machine Intelligence, VOL. 22, No. 11, pp 1266 - 1276, November 2000.
- 2] Toshio Kamei, Masanori Mizoguchi, "Fingerprint Reselection Using Eigenvectors," 0-1186-8497-6/98, IEEE 1998.
- 3] Jonathan D.stosz, "Automated fingerprint verification system" IEEE 1997.
- 4] Dmitri Linde, "A New Approach to Fingerprint Verification," Computer Science Department, California Institute of Technology, May 31, 1996.

CODING:

```
#include "display.c"
short the_image[ROWS][COLS];
float compute_line_length(int ref1_x,int ref1_y,int ref2_x,int ref2_y,int test1_x,int
test1_y,int test2_x,int test2_y);
float compute_angle_length(int ref1_x,int ref1_y,int ref2_x,int ref2_y,int test1_x,int
test1_y,int test2_x,int test2_y);
void main()
{
    char color_transform[80],monitor_type[80],name2[80],rep[80];
    intcolor,display_colors,file_d,first_element,first_line,i,ie,il,image_colors,
invert,j,le,ll,not_finished,response;
    long mean_of_pixels;
    struct tiff_header_struct image_header;
    int gdriver = DETECT, gmode, errorcode;
    int midx, midy, disp;
    initgraph(&gdriver, &gmode, "");
    errorcode = graphresult();
    if (errorcode != grOk)
    {
        printf("Graphics error: %s\n", grapherrormsg(errorcode));
        printf("Press any key to halt:");
        getch();
        exit(1); /* terminate with an error code */
    }
    not_finished =1;
    response =99;
    il =1;
    ie =1;
    ll =ROWS+1;
    le =COLS+1;
    display_colors =16;
    image_colors =16;
    invert =0;
    strcpy(color_transform,"Straight mode");
    strcpy(monitor_type,"VGA");
    top();
    gui();
    run();
    cleardevice();
    no_of_minutiae=0;
    top();
    setcolor(WHITE);
    settextstyle(2,0,5);
    outtextxy(130,getmaxy()-25,"Reference Image ");
```



```

read_tiff_header(ref_image,&image_header);
display_image(ref_image,the_image,il,ie,ll,le,&image_header,1);
no_of_minutiae=0;
top();
setcolor(0);
settextstyle(2,0,5);
outtextxy(130,getmaxy()-25,"Reference Image ");
cleardevice();
top();
setcolor(WHITE);
settextstyle(2,0,5);
outtextxy(130,getmaxy()-25,"Test Image ");
read_tiff_header(test_image,&image_header);
display_image(test_image,the_image,il,ie,ll,le,&image_header,2);
cleardevice();
top();
setcolor(WHITE);
settextstyle(2,0,5);
outtextxy(130,getmaxy()-25,"Please Wait... ");
setcolor(WHITE);
settextstyle(8,0,1);
outtextxy(170+40+50,200-20,"MATCHING.....");
matching();
cleardevice();
top();
result();
getch();
run();
}
result()
{
    char *str;
    char *app="";
    int dd=0;
    int dec, sign, ndig = 2;
    double similarity=.51;
    double sim_percent=0;

    similarity=no_of_valid_pair/(no_of_ref_minutiae*1.0);
    sim_percent=similarity*100;

    setcolor(0);
    settextstyle(2,0,5);
    outtextxy(130,getmaxy()-25,"Test Image ");

    setcolor(WHITE);

```

```
settextstyle(2,0,5);
outtextxy(130,getmaxy()-25,"RESULT ");
```

```
setcolor(0);
setfillstyle(11,3);
bar3d(115+30,85+30,getmaxx()-185+30,getmaxy()-215+30,0,0);
setcolor(0);
setfillstyle(1,7);
bar3d(100+30,100+30,getmaxx()-200+30,getmaxy()-200+30,1,1);
setcolor(0);
setfillstyle(1,0);
rectangle(150+40+50,200-20,350+40,230-20);
floodfill(155+40+50,205-20,0);
setcolor(15);
settextstyle(11,0,1);
outtextxy(145,200-9,"Status:");
```

```
if(sim_percent>=MATCH_THRESHOLD)
{
settextstyle(8,0,1);
outtextxy(170+40+50,200-20,"MATCHED");
}
else
{
settextstyle(8,0,1);
outtextxy(170+30+50,200-20,"NOT MATCHED");
}
}
```

```
settextstyle(11,0,1);
outtextxy(145,240-12,"Similarity:");
```

```
setcolor(0);
setfillstyle(1,0);
rectangle(150+40+50,240-20,350+40,260-20);
floodfill(155+40+50,245-20,0);
setcolor(15);
settextstyle(12,0,1);
setcolor(15);
rectangle(135,135,getmaxx()-200+30-5,getmaxy()-200+30-5);
rectangle(137,137,getmaxx()-200+30-7,getmaxy()-200+30-7);
settextstyle(12,0,1);
```

```
setcolor(7);
setfillstyle(1,7);
```

```

rectangle(270,135,330,155);
floodfill(275,137,7);
setcolor(15);
outtextxy(277,133,"Output");
gotoxy(35,15);
printf("%f %",sim_percent);
}
gui()
{

int ch=0;
int count=0;
unsigned int size=0;

int effect=0;
int order=0;
int del=50;
int arrow=0;
int ii=0,jj=0;

setcolor(YELLOW);
settextstyle(1,0,1);
outtextxy(20,100,"REF.IMAGE");
delay(50);
setcolor(0);
outtextxy(20,100,"REF.IMAGE");

size = imagesize(20, 80, 200, 120);
buf=malloc(size);

ch=0;
setcolor(WHITE);
rectangle(290,102,500,125);

while(1)
{
setcolor(WHITE);
settextstyle(2,0,5);
outtextxy(130,getmaxy()-25,"Enter reference image name");
settextstyle(1,0,1);
setcolor(11);

while(!kbhit())
{

```

```

if(effect==0)
{
    if(order==0)
    {
        setcolor(0);
        outtextxy(20,100,"REF.IMAGE");

        setcolor(YELLOW);
        outtextxy(21,100,"REF.IMAGE");

        delay(del);

        effect=1;
    }
    if(order==1)
    {
        delay(50);
        putimage(20,80,buf,XOR_PUT);

        setcolor(0);
        outtextxy(21,100,"REF.IMAGE");

        setcolor(YELLOW);

        outtextxy(20,100,"REF.IMAGE");

        delay(del);
        effect=0;
        order=0;
    }
}
if(effect==1)
{
    if(order==0)
    {
        setcolor(0);
        outtextxy(21,100,"REF.IMAGE");

        setcolor(YELLOW);
        outtextxy(22,100,"REF.IMAGE");

        delay(del);
    }
}

```

```

        effect=2;
    }
    if(order==1)
    {

        setcolor(0);
        outtextxy(22,100,"REF.IMAGE");

        setcolor(YELLOW);
        outtextxy(21,100,"REF.IMAGE");

        delay(del);

        effect=0;
    }
}
if(effect==2)
{
    if(order==0)
    {
        setcolor(0);
        outtextxy(22,100,"REF.IMAGE");

        setcolor(YELLOW);
        outtextxy(23,100,"REF.IMAGE");

        delay(del);

        effect=3;
    }
    if(order==1)
    {
        setcolor(0);
        outtextxy(23,100,"REF.IMAGE");

        setcolor(YELLOW);
        outtextxy(22,100,"REF.IMAGE");

        delay(del);

        effect=1;
    }
}
if(effect==3)

```

```

{
    if(order==0)
    {
        setcolor(0);
        outtextxy(23,100,"REF.IMAGE");

        setcolor(YELLOW);
        outtextxy(24,100,"REF.IMAGE");

        //putimage(21,80,buf,XOR_PUT);
        delay(del);

        effect=4;
    }
    if(order==1)
    {

        setcolor(0);
        outtextxy(24,100,"REF.IMAGE");

        setcolor(YELLOW);
        outtextxy(23,100,"REF.IMAGE");

        //putimage(21,80,buf,XOR_PUT);
        delay(del);

        effect=2;
    }
}
if(effect==4)
{
    if(order==0)
    {
        setcolor(0);
        outtextxy(24,100,"REF.IMAGE");

        setcolor(YELLOW);
        outtextxy(25,100,"REF.IMAGE");

        //putimage(21,80,buf,XOR_PUT);
        delay(del);

        effect=0;
        order=1;
    }
    if(order==1)

```

```

        {
            setcolor(0);
            outtextxy(25,100,"REF.IMAGE");

            setcolor(YELLOW);
            outtextxy(24,100,"REF.IMAGE");

            //putimage(21,80,buf,XOR_PUT);
            delay(del);

            effect=3;
        }
        //delay(10);
    }

}
setcolor(11);
ch=getch();
if(ch==8)
{
    setcolor(0);
    outtextxy(300,100,ref_image);
    ref_image[count--]=ch;
    //count++;
    ref_image[count]='\0';
    setcolor(11);
    outtextxy(300,100,ref_image);

}
else
{

ref_image[count]=ch;
ref_image[count+1]='\0';
outtextxy(300,100,ref_image);
count++;
}

ch=0;

}
setcolor(0);
for(ii=20;ii<205;ii++)
for(jj=80;jj<120;jj++)
putpixel(ii,jj,0);

```

```

setcolor(YELLOW);
setttextstyle(1,0,1);
outtextxy(20,100,"REF.IMAGE");
setttextstyle(1,0,1);
setcolor(YELLOW);
outtextxy(20,200,"TEST IMAGE:");

getimage(20,200,140,240,buf);
delay(10);
putimage(20,200,buf,XOR_PUT);

effect=0,order=0,count=0;
ch=0;
while(1)
{
    setcolor(WHITE);
    rectangle(290,202,500,225);
    setcolor(0);
    setttextstyle(2,0,5);
    outtextxy(130,getmaxy()-25,"Enter reference image name");
    setcolor(WHITE);

    outtextxy(130,getmaxy()-25,"Enter test image name");
    setttextstyle(1,0,1);
    setcolor(11);
    while(!kbhit())
    {
        if(effect==0)
        {
            if(order==0)
            {
                putimage(21,200,buf,XOR_PUT);
                delay(50);
                putimage(21,200,buf,XOR_PUT);
                effect=1;
            }
            if(order==1)
            {
                putimage(20,200,buf,XOR_PUT);
                delay(50);
                putimage(20,200,buf,XOR_PUT);
                effect=0;
                order=0;
            }

            //delay(10);

```



```

}
if(effect==1)
{
    if(order==0)
    {
        putimage(22,200,buf,XOR_PUT);
        delay(50);
        putimage(22,200,buf,XOR_PUT);
        effect=2;
    }
    if(order==1)
    {
        putimage(21,200,buf,XOR_PUT);
        delay(50);
        putimage(21,200,buf,XOR_PUT);
        effect=0;
    }
    //delay(10);
}
if(effect==2)
{
    if(order==0)
    {
        putimage(23,200,buf,XOR_PUT);
        delay(50);
        putimage(23,200,buf,XOR_PUT);
        effect=3;
    }
    if(order==1)
    {
        putimage(22,200,buf,XOR_PUT);
        delay(50);
        putimage(22,200,buf,XOR_PUT);
        effect=1;
    }
    //delay(10);
}
if(effect==3)
{
    if(order==0)
    {
        putimage(24,200,buf,XOR_PUT);
        delay(50);
        putimage(24,200,buf,XOR_PUT);
        effect=4;
    }
}

```

```

        if(order==1)
        {
            putimage(23,200,buf,XOR_PUT);
            delay(50);
            putimage(23,200,buf,XOR_PUT);
            effect=2;
        }
        //delay(10);
    }
    if(effect==4)
    {
        if(order==0)
        {
            putimage(25,200,buf,XOR_PUT);
            delay(50);
            putimage(25,200,buf,XOR_PUT);
            effect=0;
            order=1;
        }
        if(order==1)
        {
            putimage(24,200,buf,XOR_PUT);
            delay(50);
            putimage(24,200,buf,XOR_PUT);
            effect=3;
        }
        //delay(10);
    }
}
ch=getch();
if(ch==8)
{
    setcolor(0);
    outtextxy(300,200,test_image);
    test_image[count--]=ch;
    test_image[count]='\0';
    setcolor(11);
    outtextxy(300,200,test_image);
}
else
{
    test_image[count]=ch;
    test_image[count+1]='\0';
}

```

```

        //itoa(ch)
        outtextxy(300,200,test_image);
        count++;
    }
    ch=0;

}
putimage(20,200,buf,XOR_PUT);

}
print_all()
{
    int ii=0;
    for(ii=0;(ii)<no_of_ref_minutiae;ii++)
    {
        printf("\n ref1_x=%d ref1_y=%d ",ref_minutiae_x[ii],ref_minutiae_y[ii]);
    }

}
matching()
{
    int
ref1_x=0,ref1_y=0,ref2_x=0,ref2_y=0,test1_x=0,test1_y=0,test2_x=0,test2_y=0;
    int ref3_x=0,ref3_y=0,test3_x=0,test3_y=0,new_x=0,new_y=0;
    int ii=0,jj=0,kk=0,iii=0,jjj=0;
    float temp=0,diff_line_length=0,diff_line_length2=0,diff_line_length3=0;
    float diff_angle_length=0,diff_angle_length2=0,diff_angle_length3=0;
    int minutiae_found=0;
    associate_ref_test();
    for(ii=0;ii<(no_of_ref_minutiae-1);ii+=2)
    {
        minutiae_found=0;
        ref1_x=ref_minutiae_x[ii];
        ref1_y=ref_minutiae_y[ii];
        ref2_x=ref_minutiae_x[ii+1];
        ref2_y=ref_minutiae_y[ii+1];
        for(jj=0;jj<10;jj++)
        {
            test1_x=ref_test_x[ii][jj];
            test1_y=ref_test_y[ii][jj];

            for(kk=0;kk<10;kk++)
            {
                test2_x=ref_test_x[ii+1][kk];
                test2_y=ref_test_y[ii+1][kk];
            }
        }
    }
}

```



```

                break;
            }
        }
        if(minutiae_found==1)
        {
            break;
        }
    }
}
if(minutiae_found==1)
{
    break;
}
}
}
}
if(minutiae_found==1)
{
    break;
}
}
minutiae_found=0;
}
//printf("no of valid paris=%d",no_of_valid_pair);
}
order_test_minutiae(int n_x,int n_y)
{
    int xx=0,yy;
    int tt1=0;
    float tt=0,temp[100];

    for(xx=0;xx<no_of_test_minutiae;xx++)
    {
        ref3_test3[xx]=xx;
    }
    for(xx=0;xx<no_of_test_minutiae;xx++)
    {
        temp[xx]=sqrt(fabs((test_minutiae_x[xx]-n_x)*(test_minutiae_x[xx]-
n_x))+fabs((test_minutiae_y[xx]-n_y)*(test_minutiae_y[xx]-n_y)));
    }
}

```

```

}
for(xx=0;xx<no_of_test_minutiae;xx++)
{
    for(yy=xx+1;yy<no_of_test_minutiae;yy++)
    {
        if(temp[xx]>temp[yy])
        {

            tt=temp[xx];
            temp[xx]=temp[yy];
            temp[yy]=tt;

            tt1=ref3_test3[xx];
            ref3_test3[xx]=ref3_test3[yy];
            ref3_test3[yy]=tt1;

        }
    }
}
}
compute_test_coordinates(int *test3_x,int *test3_y,int ref1_x,int ref1_y,int ref2_x,int
ref2_y,int ref3_x,int ref3_y,int test1_x,int test1_y,int test2_x,int test2_y)
{
    int a=0;
    int b=0;
    int cx,cy,x1,y1;

    a=(((test2_x - test1_x)*(ref2_x -ref1_x ))+((test2_x -test1_y)*(ref2_y -
ref1_y)))/(((ref2_x -ref1_x)*(ref2_x -ref1_x))+((ref2_y -ref1_y)*(ref2_y -ref1_y )));

    b=(((test2_x - test1_x)*(ref2_y -ref1_y ))+((test2_y -test1_y)*(ref2_x -
ref1_x)))/(((ref2_x -ref1_x)*(ref2_x -ref1_x))+((ref2_y -ref1_y)*(ref2_y -ref1_y )));

    cx=(test1_x-(a*ref1_x)-(b*ref1_y));

    cy=(test1_y-(a*ref1_y)+(b*ref1_x));

    x1=(a*ref3_x)+(b*ref3_y)+cx;

    y1=((-b)*ref3_x)+(a*ref3_y)+cy;

    *test3_x=x1;
    *test3_y=y1;
}

```

```

}
float compute_line_length(int ref1_x,int ref1_y,int ref2_x,int ref2_y,int test1_x,int
test1_y,int test2_x,int test2_y)
{
    int t1=0;
    int t2=0;
    float f1=0;
    float f2=0;
    float res=0;

    // printf("\n***ref1_x=%d ref2_x=%d ref1_y=%d
ref2_y=%d",ref1_x,ref2_x,ref1_y,ref2_y);
    f1=pow((ref1_x-ref2_x),2)+pow((ref1_y-ref2_y),2);

    f1=sqrt(f1);

    f2=pow((test1_x-test2_x),2)+pow((test1_y-test2_y),2);
    f2=sqrt(f2);

    res=fabs(f1-f2);
    return(res);

}
float compute_angle_length(int ref1_x,int ref1_y,int ref2_x,int ref2_y,int test1_x,int
test1_y,int test2_x,int test2_y)
{
    float f1=0,f2=0, res=0;
    int t1=0,t2=0;

    t1=abs(ref1_x-ref2_x);
    if(t1>0)
    {
        f1=fabs(ref1_y-ref2_y)/t1;
    }
    else
    {
        f1=abs(ref1_y-ref2_x)/1;
    }
    //printf("\n f1=%f",f1);
    t2=abs(test1_x-test2_x);
    if(t2>0)
    {
        f2=fabs(test1_y-test2_y)/t2;
    }
}

```

```

}
else
{
f2=fabs(test1_y-test2_y)/1;
}
//printf("\n f2=%f",f2);
res=fabs(f1-f2);
return(res);
}

associate_ref_test()
{
int ii=0,jj=0,ref_x=0,ref_y=0,test_x=0,test_y=0,kk=0;
float temp[100];
for(ii=0;ii<no_of_ref_minutiae;ii++)
{
ref_x=ref_minutiae_x[ii];
ref_y=ref_minutiae_y[ii];
for(jj=0;jj<no_of_test_minutiae;jj++)
{
test_x=test_minutiae_x[jj];
test_y=test_minutiae_y[jj];
temp[jj]=sqrt(fabs((ref_x-test_x)*(ref_x-test_x))+fabs((ref_y-
test_y)*(ref_y-test_y)));
}
sort(temp);
map(ii,temp);
}
}
map(int ii,float temp1[100])
{
int kk=0;
for(kk=0;kk<10;kk++)
{
ref_test_x[ii][kk]=test_minutiae_x[test_order[kk]];
ref_test_y[ii][kk]=test_minutiae_y[test_order[kk]];
}
}
}
sort(float temp1[100])

```



```

{
    int ii=0,jj=0,kk=0,tt1;
    float tt=0,temp=0;

    for(ii=0;ii<no_of_test_minutiae;ii++)
    {
        test_order[ii]=ii;
    }
    for(ii=0;ii<no_of_test_minutiae;ii++)
    {
        for(jj=ii+1;jj<no_of_test_minutiae;jj++)
        {
            if(temp1[ii]>temp1[jj])
            {
                tt=temp1[ii];
                temp1[ii]=temp1[jj];
                temp1[jj]=tt;

                tt1=test_order[ii];
                test_order[ii]=test_order[jj];
                test_order[jj]=tt1;
            }
        }
    }
}

show_image(short image[ROWS][COLS],int il, int ie)
{
    int i,j;
    printf("\n ");
    for(i=0;i<18;i++)
    {
        printf(" -%3d", i+ie);
    }
    for(i=0;i<20;i++)
    {
        printf("\n %2d ", i+il);
        for(j=0;j<18;j++)
        {
            printf("-3%d",image[i][j]);
        }
    }
    printf("\n Press enter to continue");
}

```

SAMPLE SCREEN:

INPUT SCREEN



OUTPUT SCREEN

