P - 1599

FACE RECOGNITION SYSTEM

A PROJECT REPORT

Submitted by

PARTHIBAN R	71202104027
RIAZ GAFOOR S	71202104034
SARANYA LAKSHMI J	71202104035

In the partial fulfillment for the award of the degree of

BACHELOR OF ENGINNERING

in

COMPUTER SCIENCE

KUMARAGURU COLLEGE OF TECHNOLOGY , COIMBATORE

ANNA UNIVERSITY : CHENNAI 600 025

APRIL 2006

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"FACE RECOGNITION SYSTEM"** is the bonafide work of **PARTHIBAN R**, **RIAZ GAFOOR S**, and **SARANYA LAKSHMI J**, who carried out their work under my supervision.

[illegible]	[illegible]
SIGNATURE 26/4/06	**SIGNATURE** 26/4/06
Dr.S.Thangasamy	Dr.S.Thangasamy
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Professor, Computer Science, Kumaraguru college of Technology, Coimbatore 641006.	Professor, Computer Science, Kumaraguru college of Technology, Coimbatore 641006.

The candidates with the University Register Nos. 71202104027, 71202104034, 71202104035 ,were examined by us in the project viva- voce examination held on 2/5/06

[illegible]	[illegible] 02/05/06
INTERNAL EXAMINER	**EXTERNAL EXAMINER**

DECLARATION

We ,

PARTHIBAN R	**71202104027**
RIAZ GAFOOR S	**71202104034**
SARANYA LAKSHMI J	**71202104035**

Declare that the project entitled " **FACE RECOGNITION SYSTEM** " submitted in the partial fulfillment to the Anna University as the project work of **Bachelor Of Engineering (Computer Science**) Degree, is a record of the original work done by us under the supervision and guidance of **Dr.S.Thangasamy B.E (Hons),Ph.D.,** Professor ,Department of Computer Science , Kumaraguru College Of Technology , Coimbatore.

Place :Coimbatore

Date : 26-04-06

R. Parthiban

[Parthiban R]

[illegible]

[Riaz Gafoor S]

[illegible]

[Saranya Lakshmi J]

Project Guided by

[illegible] 26/4/06

[Dr.S.Thangasamy B.E(Hons),Ph.D.]

ACKNOWLEDGEMENT

We are extremely grateful to **Dr.K.K.Padmanaban**, B.Sc.(Engg.), M.Tech., Ph.D., Principal , Kumaraguru College of Technology for having given us a golden opportunity to embark on this project.

We are deeply obliged to **Dr.S.Thangasamy** B.E(Hons),Ph.D., Head of the Department ,Computer Science and Engineering, Kumaraguru College Of Technology , Coimbatore , for his valuable guidance and useful suggestions.

We are grateful to our Course Coordinator **Dr.P.Devaki** MS ,Department of Computer Science, Kumaraguru College of Technology, Coimbatore, for her encouragement and support at various levels of this project work.

We also express our sincere thanks to **Dr.S.Thangasamy** B.E(Hons),Ph.D., Professor Computer Science and Engineering, Kumaraguru College Of Technology , Coimbatore, for his valuable guidance and encouragement at every stage of the project.

Most of all , we thank **Our Parents and Friends** for their blessings and support without which we would not be able to do anything.

ABSTRACT

The Human face is an important part of an individual. It well defines who you are and how people identify you. The face is arguably a person's most unique physical characteristic. With the help of a face, an individual can be uniquely identified or recognized. While humans have had the innate ability to recognize and distinguish different faces for millions of years, computers are just now catching up. One such venture is the

" FACE RECOGNITION SYSTEM ".

This project is developed for the recognition of human faces. The collections of pictures of human faces are captured and are stored. The textual descriptions and details of the images are simultaneously stored in the database. When a person is to be recognized (i.e) to determine if the person is one among the group, the persons image is captured and the match is determined by the comparison made between the collection of images and the newly captured image.

Face Recognition System is implemented by applying the basic methodologies and techniques of **Feature Classification** for the recognition of individuals uniquely. This project has facilities to store and update the individual's textual details and description in the database. The FRS uses the skin texture and color to take apart the skin regions from other regions. The region which has two holes representing the eyes is taken as the face. The features are classified using SVM classifier working on the basis of Local Binary Patterns. The comparison or the match between two images is also made with the SVM classifier.

TABLE OF CONTENTS

CHAP NO.	TITLE	PAGE NO.
	ABSTRACT	V
	LIST OF TABLES	VIII
	LIST OF FIGURES	VIII
	LIST OF ABBREVATIONS	IX
1	INTRODUCTION	
	1.1 Description	1
	1.2 Problem Description	2
2	SOFTWARE REQUIREMENTS ANALYSIS	
	2.1 Existing System	3
	2.2 Proposed System	3
	2.3 Limitations	3
3	SOFTWARE REQUIREMENTS SPECIFICATION	
	3.1 Introduction	
	3.1.1 Purpose	4
	3.1.2 Scope	4
	3.1.3 Definition	4
	3.2 General Description	
	3.2.1 Product Overview	5
	3.2.2 User Characteristics	5
	3.2.3 General Constraints	5
	3.2.4 General Assumptions	5

	3.3	Specific Requirements	
	3.3.1	Input and Output Format	6
	3.3.2	Functional Requirements	6
	3.3.3	System Requirements	7
	3.3.4	Performance Constraints	7
	3.3.5	Software Constraints	7
4		SOFTWARE STUDY	
	4.1	Introduction to MATLAB	8
	4.2	Introduction to SVM	12
5		PROJECT DESCRIPTION	14
6		SOFTWARE DESIGN	
	6.1	Use Case Diagram	17
	6.2	Sequence Diagram I	18
	6.3	Sequence Diagram II	19
	6.4	DFD -Level 0	20
	6.5	DFD –Level 1	21
	6.6	Database Design	22
7		PRODUCT TESTING	22
8		FUTURE ENHANCEMENTS	23
9		CONCLUSION	24
10		APPENDIX	25
11		REFERENCES	52

LIST OF TABLES

TABLE NO	TABLE NAME	PAGE NO
1	STAFF DETAILS	22

LIST OF FIGURES

FIGURE NO	DESCRIPTION	PAGE NO
1	ARCHITECTURE OF FACE RECOGNITION SYSTEM	2
2	USE CASE DIAGRAM	17
3	SEQUENCE DIAGRAM I	18
4	SEQUENCE DIAGRAM II	19
5	DFD – LEVEL 0	20
6	DFD – LEVEL I	21

LIST OF ABBREVATIONS

FRS	Face Recognition System
SRS	Software Requirements Specification
Matlab	Matrix Laboratory
SVM	Support Vector Machine
LBP	Local Binary Pattern
SV	Support Vector
GUI	Graphical User Interface
DFD	Data Flow Diagram

1. INTRODUCTION

1.1 DESCRIPTION

" An infant innately responds to face shapes at birth and can discriminate his or her mother's face from a stranger's at the tender age of 45 hours.. " (Voth 2003)

The simple indication above reveals the remarkable abilities of humans to perform a vital survival skill which is face recognition. Indeed, face recognition is a natural and straightforward biometric method that human beings use to identify each other. However, developing technologies that can mimic such ability is a challenging problem which has received much attention during the recent years. This increased interest in automatic face recognition is due to the many potential applications in different fields

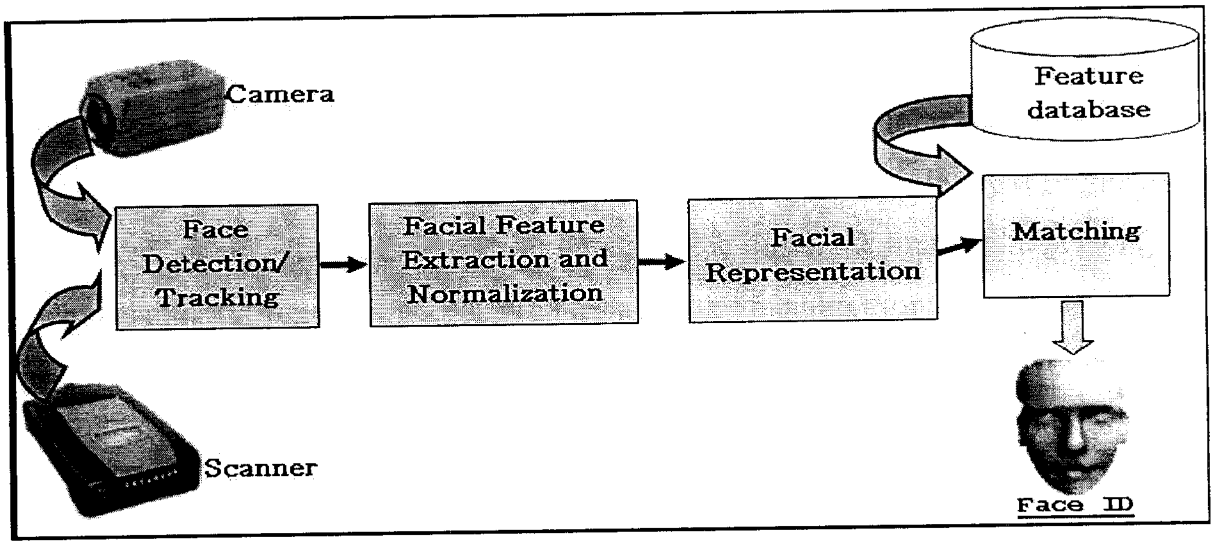
Verification of person identity based on biometric information is important for many security applications. Examples include access control to buildings, surveillance and intrusion detection. Furthermore, there are many emerging fields that would benefit from developments in person verification technology such as advanced human-computer interfaces and tele-services including tele-shopping and tele-banking. The field of face recognition is well established and a large number of algorithms have been proposed in the literature.

To recognize faces, this PROJECT proposes a new approach based on Local Binary Patterns (LBP) which consists of dividing the facial image into small regions from which LBP features are extracted and concatenated into a single feature histogram efficiently representing the face image. Then, face recognition is performed using a nearest neighbor classifier in the computed feature space

1.2 PROBLEM DESCRIPTION

This project describes about the design, development and implementation of a Face Recognition System. The FRS automatically recognizes images o f persons. This system, in outline, gets the input, the image of a person, separates the skin regions, records or takes apart the facial region. From the detected face the features are extracted and the classification and recognition of two images is done based on the extracted features. The FRS is divided into various modules for easy implementation.

Architecture of Face Recognition System



The initial step is the reporting of the skin regions from the overall image. The initial step of skin region identification is performed as below. The image is described in the RGB color space. From it, the chromatic or pure color space description of the image is made. Based on the Gaussian model for skin color distribution the likeliness of the pixel being skin is determined from the skin regions are obtained. From the facial regions, based on the concept of holes in the facial region in a gray scale image, the face is detected from the other regions. Local Binary Pattern technique is used as an aid to the feature extraction. The binary patterns of the face are converted into a histogram and the input is given to the SVM classifier.

The classifier automatically classifies the features and reports if a match exists or not. Database connectivity is established and the textual description of the user is added to the database. These data are retrieved when the recognition is made.

2. SOFTWARE REQUIREMENTS ANALYSIS

2.1 EXISTING SYSTEM

There are various types of techniques and methodologies in order to accomplish Face recognition. Various algorithms have been implemented; each has its own percentage of success. The classifiers are implemented separately and the algorithms based systems are implemented separately. The classification is done either based on color or any other technique. But in existing systems, no two methodologies have been combined to develop a single Face recognition system.

2.2 PROPOSED SYSTEM

Our Face Recognition System has been designed in such away that it incorporates the best techniques and methodologies. It combines the currently or recently developed approaches and techniques. The system uses different techniques in order to handle different phases in the system. The determination of skin regions is based on the color space and the color distribution model. The facial region detection is based on the eye based identification technique. The classification of the features is based on a latest texture pattern based technique called as Local Binary Pattern. The final extraction of features and the classification and recognition of the faces is accomplished by the SVM Classifier. The database updates of the textual details are also made using a back end tool.

2.3 LIMITATIONS

The system has certain limitation that hinders the performance efficiency of the recognition system. The design and the implementation of this system require that SVM classifier has to be trained largely. It requires that for a single person, a minimum of ten samples be given as input to the classifier. The classifier has to be trained well in order to obtain higher percentage of success in the correct recognition of faces. The rate of failure can be greatly reduced if the training and the detection of the face samples is done for a sufficient number of times.

The system also requires constant lighting and background conditions. The small variations in the lighting would cause considerable change in the extraction and classification of images causing higher rates of failure in recognition.

3. SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Introduction

3.1.1 Purpose

The purpose of this document is to specify the requirements of the project "Face Recognition System ". It serves as an automatic recognition system used for various authorization applications.

3.1.2 Scope

SRS forms the basics for agreement between the client and the supplier and what the software product will do. It also provides a reference for the validation of the final project. Any changes made to the SRS in the future will have to go through formal change approval process.

3.1.3 Definition

Customer

A person or organization, internal or external to the producing organization, which takes financial responsibility for the system. In a large system this may not be the end user. The customer is the ultimate recipient of the developed product and its artifacts.

User

A person who uses the system that is developed.

Analyst

The analyst details the specification of the system's functionality by describing the requirements aspect and the other supporting software.

3.2 General Description

3.2.1 Product Overview

Our project aims at making the authorization applications more secure using biometric techniques. This project can perform the recognition of static images of persons. The project has been implemented based on the latest techniques and tools. Customizable user interface is also provided in our project.

3.2.2 User Characteristics

The main user of this system is the person who is to be recognized. He/She is expected to have a basic knowledge about the usage of forms in a computer.

3.2.3 General Constraints

The system is dependent on the Windows Operating system and is designed to work using the current and latest version of the MATLAB.

3.2.4 General Assumptions

The Operating System of the computer has been installed with a driver to support the Web Cam. An interface between the Web Cam and the Matlab software has been established.

3.3 Specific Requirements

3.3.1 Inputs and Outputs

Capturing image

Input : video from webcam.

Output : captured still image displayed on the axes.

Face detection and database addition

Input : static image with file name to be saved and other textual details

Output : detected face displayed on the axes.

Face recognition

Input : newly captured static image with a different file name.

Output : if the match exists, corresponding details from the database id displayed, else "no match" comment is displayed.

3.3.2 Functional Requirements

- **i.** The system should be able to extract the static image from the video input and display it.
- **ii.** The system should be able to detect the face alone from the static image and display the detected face in the axes.
- **iii.** The system should update the details of the user in the database.
- **iv.** The system should classify and extract the features from the face.
- **v.** The system should compare the images and display the matched image and the corresponding details from the database.

3.3.3 System requirements

Hardware Requirements

Processor	:	Pentium IV
RAM size	:	128 / 256 MB RAM
Hard Disk Capacity	:	20 / 40 GB
Other devices	:	Web Cam

Software Requirements

Operating System	:	Windows XP / 98
Software	:	MATLAB 7.0, MS Access

3.3.4 Performance constraints

i. The lighting conditions should be maintained constant. There should be no small variations in illumination and intensity.
ii. The background of the images should not vary.
iii. The angle of the web cam should not be altered.
iv. The input image should contain only one human face.
v. The frontal face only can be recognized (side views cannot be handled).

3.3.5 Software constraints

The system runs on windows platform. The system requires the current version of MATLAB(7.0) be installed in the computer. The driver to support web cam should also be installed in the computer prior to the execution of the system.

4. SOFTWARE STUDY

4.1 INTRODUCTION TO MATLAB :

Matlab is a commercial "Matrix Laboratory" package which operates as an interactive programming environment. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar noninteractive language such as C or Fortran.

The name MATLAB stands for ***matrix laboratory***. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB engines incorporate the LAPACK and BLAS libraries, embedding the state of the art in software for matrix computation. MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of add-on application-specific solutions called ***toolboxes***. Very important to most users of MATLAB, toolboxes allow you to ***learn*** and ***apply*** specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others. Matlab is especially designed for matrix computations: solving systems of linear equations, computing eigen values and eigen vectors, factoring matrices, and so forth. It has a variety of graphical capabilities, and can be extended through programs written in its own programming language.

MATLAB is a high-level language and interactive environment that enables us to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran. MATLAB is an interpretative (not compiled) language. Such a file is called an m-file.

User-defined Matlab functions are interpreted not compiled. This means roughly that when an m-file is executed, each statement is read and then executed, rather than the entire program being parsed and compiled into machine language. For this reason, Matlab programs can be much slower than programs written in a language such as Fortran or C.

It is an interactive program for numerical computation and data visualization; it is used extensively by control engineers for analysis and design. There are many different toolboxes available which extend the basic functions of Matlab into different application areas; in these tutorials, we will make extensive use of the Control Systems Toolbox. Matlab is supported on Unix, Macintosh, and Windows environments; a student version of Matlab is available for personal computers. MATLAB provides a powerful interactive computing environment for numeric computation, visualization, and data analysis. Its wide range of commands, functions, and language constructs permit users to solve and analyze difficult computational problems from science and engineering without programming in a general purpose language. Matlab program and script files always have filenames ending with ".m"; the programming language is exceptionally straightforward since almost every data object is assumed to be an array. Graphical output is available to supplement numerical results. Matlab is a versatile program which allows you to use a computer to solve a vast number of problems in science and mathematics both numerically and through visualization.

MATLAB System :

The MATLAB system consists of five main parts:

Desktop Tools and Development Environment. This is the set of tools and facilities that help you use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, an editor and debugger, a code analyzer and other reports, and browsers for viewing help, the workspace, files, and the search path.

The MATLAB Mathematical Function Library. This is a vast collection of computational algorithms ranging from elementary functions, like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigen values, Bessel functions, and fast Fourier transforms.

The MATLAB Language. This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create large and complex application programs.

Graphics. MATLAB has extensive facilities for displaying vectors and matrices as graphs, as well as annotating and printing these graphs. It includes high-level functions for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level functions that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

The MATLAB External Interfaces/API. This is a library that allows you to write C and Fortran programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

Strengths of MATLAB :

- Flexibility to solve a large number of problems
- Good tools for visualization
- Can be used on most computer systems (Mac, PC, Unix Workstation, Cray, ...)
- Used at many colleges, universities and industry in many different disciplines (science, math, computer science, finance)
- The scripting language is relatively easy to learn, and is complementary with traditional programming languages (BASIC, FORTRAN, C, Pascal)

Applications / Uses of MATLAB :

- Plotting and analyzing mathematical relationships (2D and 3D)
- List & Matrix Operations
- Writing script files (a type of programming)
- Symbolic manipulation of equations
- Advanced visualization, animation and GUI interface tools
- Algorithm development
- Data acquisition
- Modeling, simulation, and prototyping
- Data analysis, exploration, and visualization
- Scientific and engineering graphics
- Application development, including graphical user interface building

4.2 INTRODUCTION TO SUPPORT VECTOR MACHINE

The study describes Support Vector Machines (SVMs) in the context of face authentication. Our study supports the hypothesis that the SVM approach is able to extract the relevant discriminatory information from the training data. We believe this is the main reason for its superior performance over benchmark methods. When the representation space already captures and emphasizes the discriminatory information content as in the case of Fisherfaces, SVMs loose their superiority. SVMs can also cope with illumination changes, provided these are adequately represented in the training data. However, on data which has been sanitized by feature extraction (Fisherfaces) and/or normalization, SVMs can get over-trained, resulting in the loss of the ability to generalize. SVMs involve many parameters and can employ different kernels. This makes the optimization space rather extensive, without the guarantee that it has been fully explored to find the best solution.

SVMs are based on the principle of structural risk minimization. The aim is to minimize the upper bound on the expected (or actual) risk defined as

$$R(\alpha) = \int \tfrac{1}{2} z - |f(x,\alpha)| \; dP(x,z)$$

where α is a set of parameters defining the trained machine, z a class label associated with a training sample x, f(x; α) a function providing a mapping from training samples to class labels, and P(x; z) the unknown probability distribution associating a class label with each training sample. Let l denote the number of training samples and choose some η such that $0 < \eta < 1$. Then, with probability $1-\eta$,the following bound on the expected risk holds:

$$R(\eta) = Remp(\eta) + rh(\log(2l=h) + 1) - \log(x=4)$$

where Remp(η) is the empirical risk as measured on the training set and h is the so called
Vapnik Chervonenkis (VC) dimension. The second term on the right hand side is called the VC confidence. There are two strategies for minimizing the upper bound. The first one is to keep the VC confidence fixed and to minimize the empirical risk and the second one to fix the empirical risk (to a small value) and minimize the VC confidence. The latter approach is the basis for SVMs and below we will briefly outline this procedure. First consider the linear separable case. We are looking for the optimal hyperplane in the set of hyperplanes separating the given training samples. This hyperplane minimizes the VC confidence and provides the best generalization capabilities.

The objective of SVM is to construct an optimal hyperplane that correctly classifies data points as much as possible and separates the points of two or more classes as far as possible. In other words it finds the best possible decision surface such that the margin of separation between the positive and negative example is maximized.

Optimal hyperplane for linearly separable patterns

Given that a training sample are linearly separable, the equation of a decision surface in the form of a hyperplane that does the separation is

$$w^T x + b = 0$$

where x is an input vector ,w is an adjustable weight vector representing the normal to the hyperplane and b, the bias of the hyperplane from the origin.

Optimal hyperplane for the nonlinearly separable patterns

When patterns are non linearly separable ,some transformations are needed for Support Vector Machine. It involves transforming a non linear mapping into a linearly separable form and then proceeding as above. The SVM kernel based methods solve the non linear mapping. The SVM classifier is the best classifier up to date. The classifier can be used for large databases also.

5. PROJECT DESCRIPTION

The project implementation is done in different phases so as to support future enhancements and easy modifications can be made to the system with ease. The major modules implemented in the system are

- The color space conversion
- The skin region separation
- The face region detection
- The detected face/feature extraction
- The feature classification and recognition

The color space conversion - The skin region separation –face detection module

The skin region separation module is based on different color space. The image is initially obtained in RGB color space. Since RGB color scale is based on luminance (lighting conditions) it causes certain problems to many image processing algorithms. For skin region separation, this scale causes error in differentiation of skin and non skin regions. Hence to remove luminance we use **Chromatic color** (pure color –in the absence of luminance) **space.** A pixel that is transformed from RGB color space to chromatic color space has a chromatic pair value (r,b).The change in different color space is brought about by the technique of Normalization. The **normalization** process to obtain the chromatic colors is given by

$$r = R / (R + G + B)$$

$$b = B / (R + G + B)$$

The third chromatic color can be replicated using the other two colors. In the chromatic color space the skin colors of different people is plotted and marked. After plotting it is obtained that the skin colors are clustered in a small area of the chromatic color space. This shows that the skin colors of people are very close but the intensities vary a lot. The skin color distribution is given by a Gaussian model N (m, c) where m is mean and c is the covariance.With the **Gaussian fitted skin color model**, we can obtain the likelihood of the skin or any pixel of an image. The likelihood of the pixel being skin is given and computed by using

$$\text{Likelihood} = P(r,b) = \exp\left[-0.5\,(x-m)^T c^{-1} (x-m)\right]$$

Where $x=(r,b)^T$ .each and every pixel will have only one value. Thus the skin color model converts color image to gray scale image. For skin regions gray scale value is less. The skin regions where brighter than the non skin regions.

The best method for skin identification, implemented here is the **"Adaptive Threshold Process"**. Since a fixed threshold cannot be used, "adaptive threshold" is employed. Different people have different likelihood so threshold has to change accordingly. The technique is based on the principle that "stepping down the threshold value may intuitively increase the segmented region. The threshold value at which the minimum increase in region size is observed while stepping down the threshold value will be the optimal threshold. The threshold depends on the gray values of the image.

For removal of regions **"binarization technique"** is utilized. After the thresholding is done, the image obtained is a binary image. A '0' indicates a non skin region and '1' indicates a skin region. The skin regions have boundaries. We label the regions. The labeling is done by considering the 8 neighborhood pixels of each pixel. Initially we label one pixel which has 1.next pixel is considered its neighborhood, if it's labeled the same is assigned. If not the new label is given to the pixel. Hence the skin regions are labeled. Considering each skin region, checking is done to see if the region corresponds to a face .This is done using the determination of the number of holes in the region. Based on **"Euler Number"**, the number of holes is determined.

$$H = 1 - E$$

Where H indicates the number of holes and E denotes the Euler Number.

The region which has two holes in it is considered as the facial region where the two holes are indications of the eyes of the face. This region with the same label is separated. Thus the face is detected from the entire image.

Feature Extraction –Face Classification –Recognition module

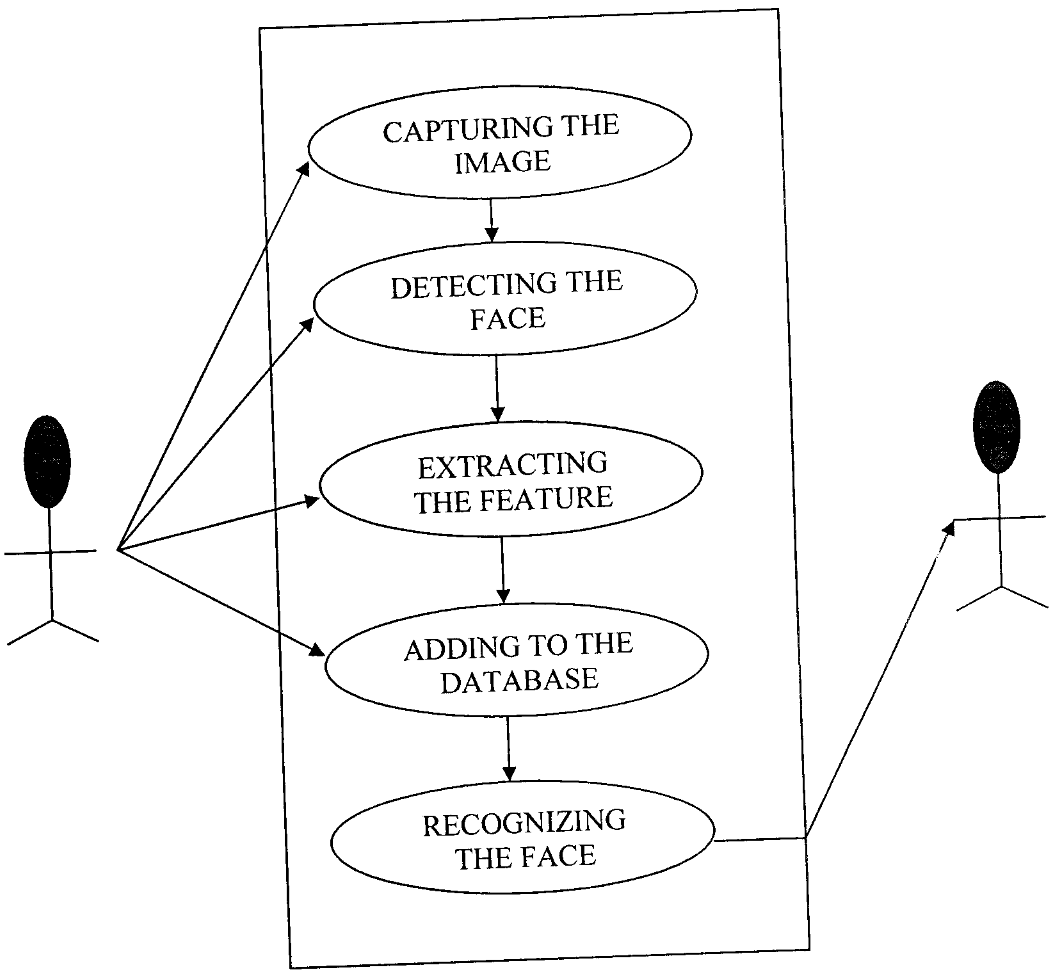
Once the face has been detected, the next step is the extraction of the features. For this step, we implement a latest and efficient technique called as **"Local Binary Pattern"**. Local Binary Pattern (LBP) is a new feature space for representing face images and shows its efficiency in face detection and recognition problems. Texture operator which labels the pixels of an image by thresholding 3*3 neighborhood of each pixel with the value of the center pixel and considers the result as a binary number. Each bin code is a micro texton .Many local primitives are codified by these bins include different types of curved edges, spots; flat areas etc.The neighborhood consideration for each pixel can be modified.

LBP $_{P,R}$ is the notation used to indicate the pattern where P denotes the number of neighborhood pixels and R indicates the radius of covering circle. For P pixels, 2^P combinations would be obtained. Patterns with minimum transitions are called as fundamental patterns. These patterns are called as **uniform pattern.** So patterns with more than 2 transitions are only considered. The patterns based on uniform patterns is given as **LBP** $^{u2}_{P,R}$.

The occurrence of different micro patterns gives various features in the face without any indication of the location. For efficient face representation one should also have the spatial information. For this reason, the face is divided into several regions (blocks) from which the histogram are computed. The various facial regions are given by the LBP histograms and the facial shape is represented by the various blocks that are divided. The standard size of each assumed window is to be about 18*21pixels. For each and every bin, the weights are assigned. Based on the weights, the various features are classified. The classification and the recognition is performed by the **Support Vector Machine (SVM) classifier**. The classifier takes as input the samples and their labels along with the feature vectors. The feature space is converted into a higher dimensional vector space by using the kernel function to make it linearly separable. The SVMClass module is implemented which classifies the features based on the weights. These weights are based on the template data. The labels and the samples which are given more weightage are considered and the values are stored. For the facial recognition the **SVMTest** module compares the given weightage of the new image and the already stored image. If the values match then the correct image is recognized. Thus the classification and recognition is done image. If the values match then the correct image is recognized. Thus the classification and recognition is done.

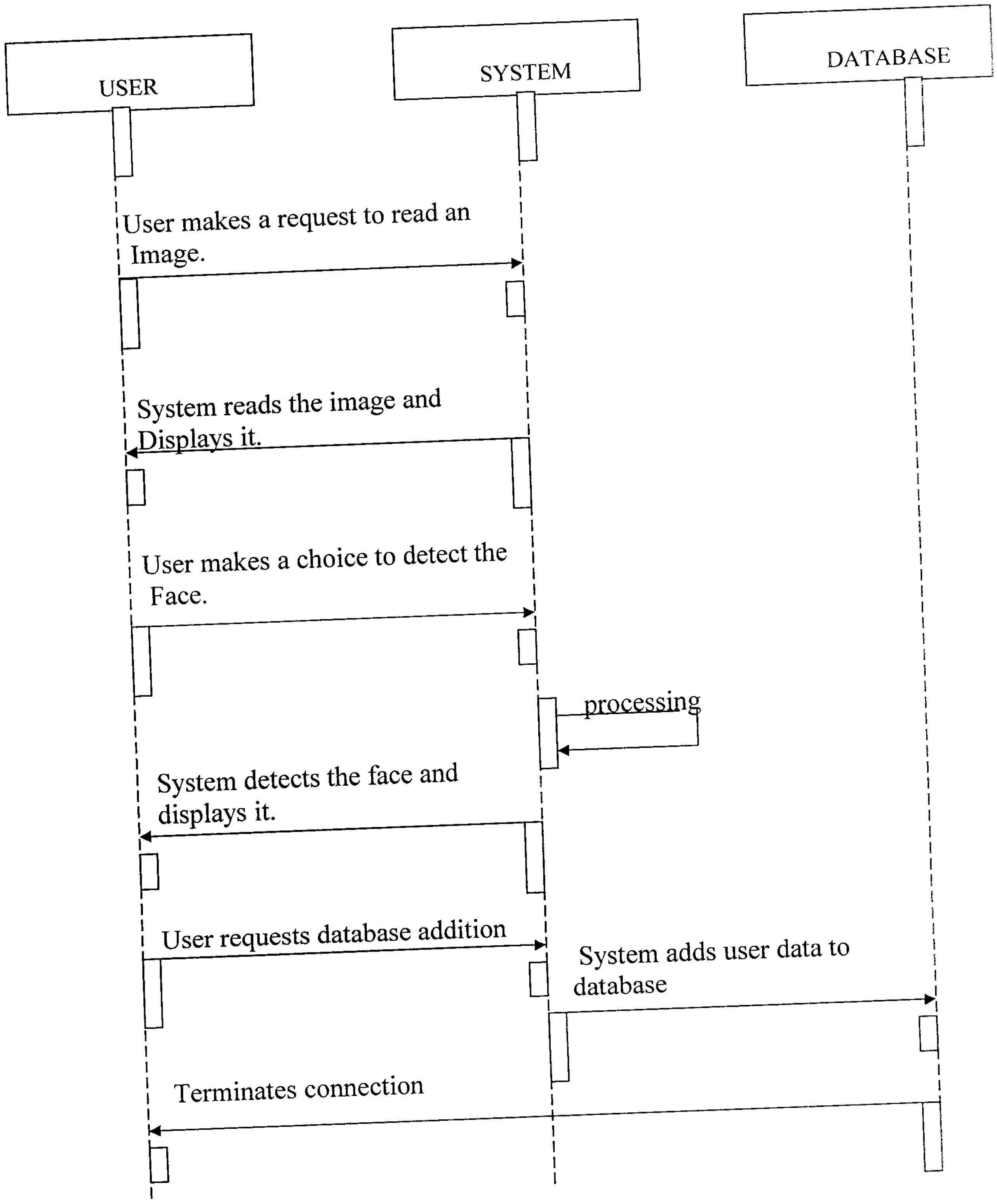
6. SOFTWARE DESIGN

6.1 USE CASE DIAGRAM

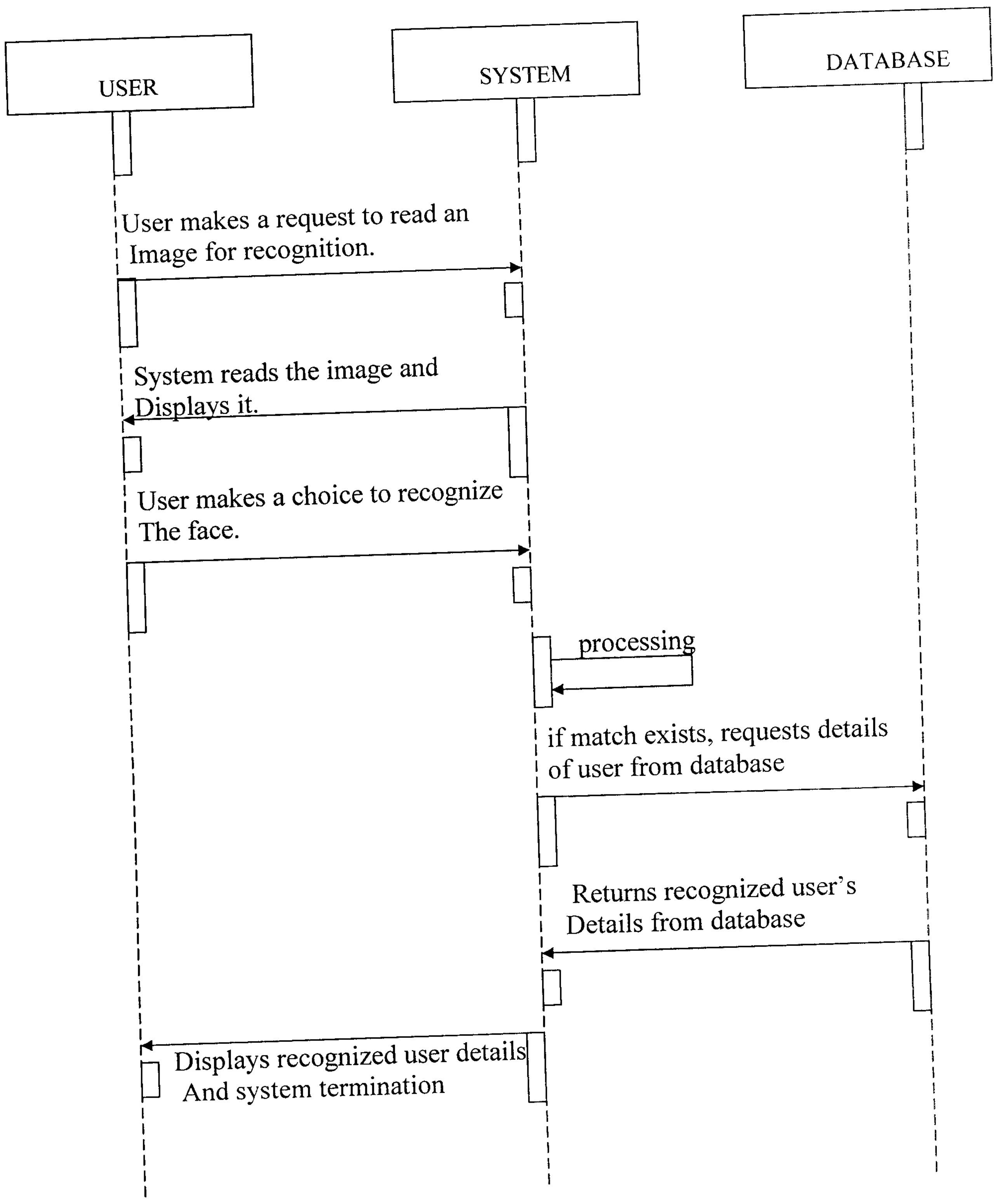


6.2 SEQUENCE DIAGRAM I

FACE DETECTION & DATABASE ADDITION

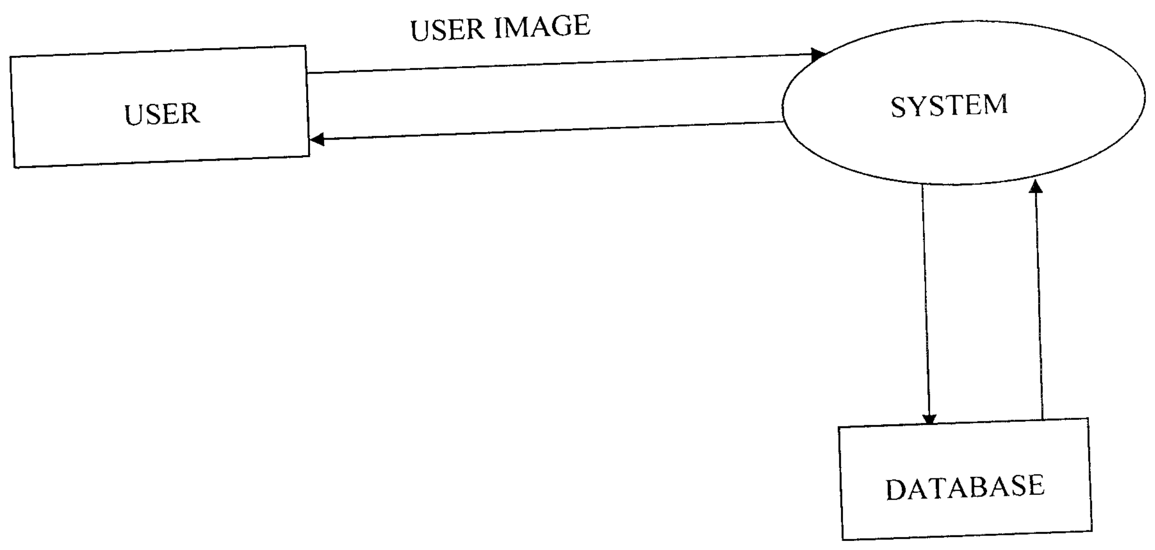

USER
SYSTEM
DATABASE
User makes a request to read an Image.
System reads the image and Displays it.
User makes a choice to detect the Face.
processing
System detects the face and displays it.
User requests database addition
System adds user data to database
Terminates connection

6.3 SEQUENCE DIAGRAM II
FACE RECOGNITION

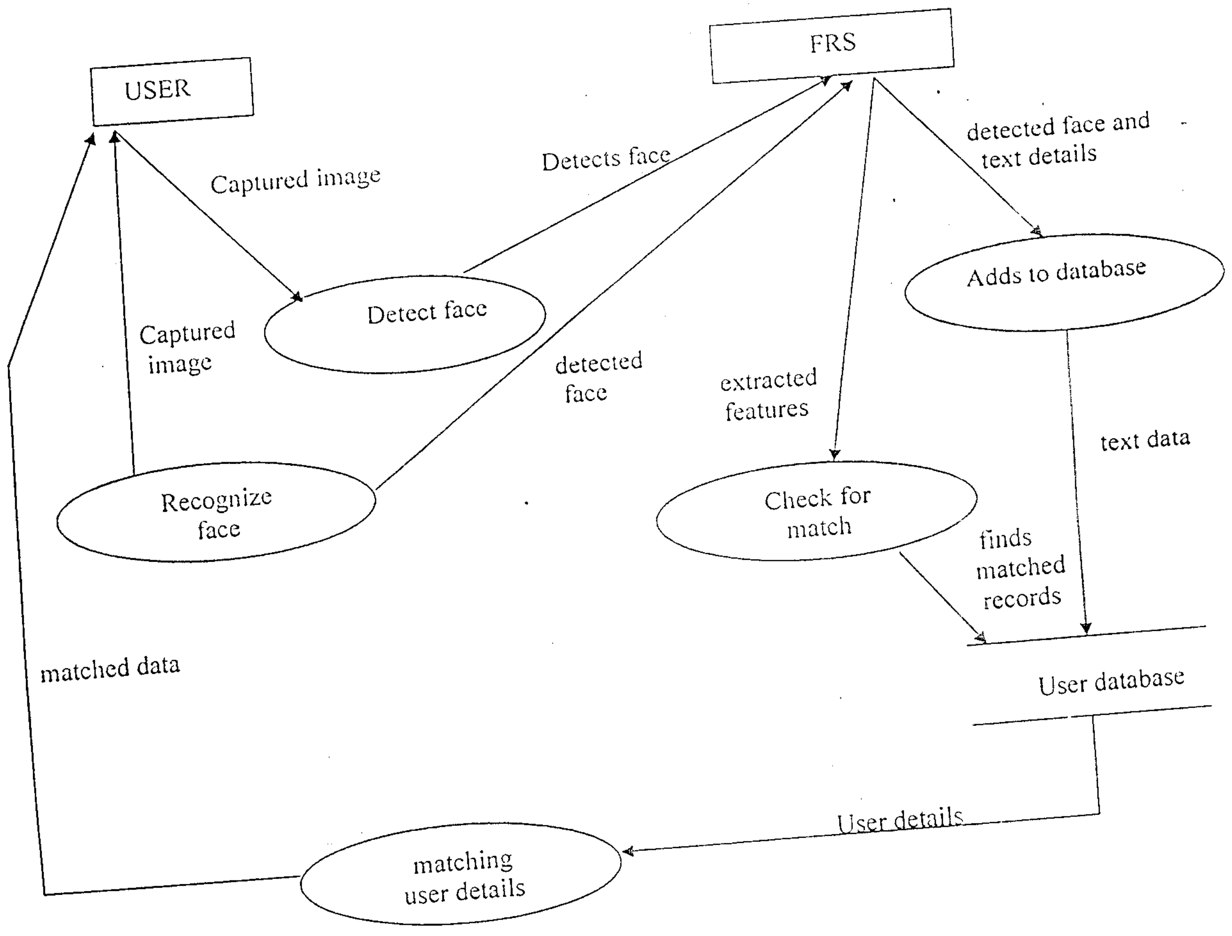

USER
SYSTEM
DATABASE
User makes a request to read an Image for recognition.
System reads the image and Displays it.
User makes a choice to recognize The face.
processing
if match exists, requests details of user from database
Returns recognized user's Details from database
Displays recognized user details And system termination

6.4 DATA FLOW DIAGRAMS

LEVEL 0



6.5 DFD LEVEL 1



6.6 DATABASE DESIGN

The database consists of a single table.

Staffdet table

E_No	E_Name	Sex	Department	Qualification

7. PRODUCT TESTING

Testing is done to detect errors in the software. It implies not only to the coding phase but also to all the phases in the development cycle. The following were the tests that were performed.

Unit Testing :

Each and every module is tested separately to check if its intended functionality is met. Some of the unit testing performed in the project are

- Checking for the proper image capturing and displaying the captured image within the specified axes.
- Correct detection of the face from the captured image.
- Checking for the proper Database Connectivity.
- Verification of proper recognition of the detected face.

Integration Testing :

Integration testing is performed to detect errors on the interconnection between modules. Here, the modules for image capturing , face detection, database addition and face recognition are combined and tested to ensure that they work in synchronization and without interference with each other.

System Testing :

The system is tested against the system requirements to see if all the requirements are met and to see if all the system performs as per the specified requirements. The system is tested as a whole to check for its functionality.

Validation Testing :

The validation testing is performed to check the validity of the entered input. In this system the validation testing is performed to see to that the employee details that is entered is in the appropriate required format.

8. FUTURE ENHANCEMENTS

The project Face Recognition System is designed in such a way that future enhancements can be made in an easy and effective manner. The project is quite flexible and can be easily customized according to the user's requirements.

One of the enhancements is the recognition of dynamic images. The system can be extended to moving or video input. The system can be made to work in lighting and background independent environment. The project serves as a base for many biometric techniques. The system can be made to recognize more than one faces, by giving an image with more than one human image as input. The system can be made to recognize side views and other inclinations of the person.

9. CONCLUSION

Face Recognition System enables easy and simple authorization of individuals. Verification of person identity based on biometric information is important for many security applications. Examples include access control to buildings, surveillance and intrusion detection. Furthermore, there are many emerging fields that would benefit from developments in person verification technology such as advanced human-computer interfaces and tele-services including tele-shopping and tele-banking. Thus Face Recognition System provides a base for many future developments in security applications. This system based on the analysis of faces is often effective without the user's cooperation or knowledge.

10.0 APPENDIX

SAMPLE CODE

FACE DETECTION MODULE

```
function varargout = mainface(varargin)
%face recognition
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                'gui_Singleton',  gui_Singleton, ...
                'gui_OpeningFcn', @mainface_OpeningFcn, ...
                'gui_OutputFcn',  @mainface_OutputFcn, ...
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
 if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
 end

 if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
 else
    gui_mainfcn(gui_State, varargin{:});
 end



 % --- Executes just before mainface is made visible.
 function mainface_OpeningFcn(hObject, eventdata, handles, varargin)
 % This function has no output args, see OutputFcn.
 % hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to mainface (see VARARGIN)
% Choose default command line output for mainface
% --- Outputs from this function are returned to the command line.
function varargout = mainface_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

% --- Executes on button press in facetestdata.
function facetestdata_Callback(hObject, eventdata, handles)
% hObject    handle to facetestdata (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
d =dir('*.jpg');
labels=[];
samples=[];
s1=size(d);
countI=1;
for i=1:s1(1)
pic=double(imread(d(i).name,'jpg'));
axes(handles.xpic);
imshow(uint8(pic));
while ((size(pic,1)*size(pic,2))>(400*300))
    pic = getpic(pic,1,2);
```

```
end;
 [pos,facesamples] = findfaces(pic);


faceNumber=[];
if (isempty(faceNumber))
   faceNumber=size(pos,1);
   i = 1;
end
if i > faceNumber
   noMoreFaces=imread('no_more_faces.jpg','jpg');
   imshow(noMoreFaces);


else
   scale=pos(i,3);
   bigy=double(pos(i,1))*scale;
   bigx=double(pos(i,2))*scale;
   binpic(bigy:bigy+20*scale,bigx:bigx+20*scale)=0;
   if (bigy>2*scale)
      bigy=bigy-2*scale;
   end


   if (bigx>scale)
      bigx=bigx-scale;
   end


   im=pic(bigy:bigy+19*scale,bigx:bigx+19*scale)/255;
   dim=size(im,1);
   im_exp = pic(max(bigy-2*scale,1):min(bigy+23*scale,size(pic,1)),max(bigx-
1*scale,1):min(bigx+21*scale,size(pic,2)))/255;
```

```
  %im = histeq(im,255);
  im = resample(im_exp',50,dim);
  im = resample(im',50,dim);
  axes(handles.xface);
  imshow(im);

end
[nosey,nosex]=locate(im,1);
[mouthy,mouthx]=locate(im,2);
[lefteyey,lefteyex]=locate(im,3);
[righteyey,righteyex]=locate(im,4);
[nby,nbx]=locate(im,5);
coord=[nosey,nosex;mouthy,mouthx;lefteyey,lefteyex;righteyey,righteyex;nby,nbx];
nose = reshape(im(coord(1,1):coord(1,1)+17,coord(1,2):coord(1,2)+12),234,1);
mouth = reshape(im(coord(2,1):coord(2,1)+12,coord(2,2):coord(2,2)+26),351,1);
lefteye = reshape(im(coord(3,1):coord(3,1)+14,coord(3,2):coord(3,2)+14),225,1);
righteye = reshape(im(coord(4,1):coord(4,1)+14,coord(4,2):coord(4,2)+14),225,1);
nb = reshape(im(coord(5,1):coord(5,1)+13,coord(5,2):coord(5,2)+15),224,1);
tset=[nose;mouth;lefteye;righteye;nb];
newLabel=countI;
labels=[labels newLabel];
samples=[samples tset];
countI=countI+1;
end cd \
[AlphaY,SVs,Bias,Parameters,nSV,nLabel]=RbfSVC(samples,labels);
save Tm AlphaY SVs Bias Parameters nSV nLabel;
```

```
% --- Executes on button press in adddata.
function adddata_Callback(hObject, eventdata, handles)
% hObject    handle to adddata (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

d1=handles.rdata.rname;
d2=handles.rdata.rno;
d3=handles.rdata.rsex;
d4=handles.rdata.rdpart;
d5=handles.rdata.rqualifi;
datab(d2,d1,d3,d4,d5);

% --- Executes on button press in resetd.
function resetd_Callback(hObject, eventdata, handles)
% hObject    handle to resetd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

mm='   ';
set(handles.rname,'string',mm);
set(handles.rno,'string',mm);
set(handles.rsex,'string',mm);
set(handles.rdpart,'string',mm);
set(handles.rqualifi,'string',mm);
set(handles.rresult,'string',mm);

function rname_Callback(hObject, eventdata, handles)
% hObject    handle to rname (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


mname=(get(hObject,'String'));
if isnan(mname)
   set(hObject, 'String', 0);
   errordlg('Input must be a number','Error');
end
handles.rdata.rname=mname;
guidata(hObject,handles);



% --- Executes during object creation, after setting all properties.
function rname_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rname (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end
```

```
function rno_Callback(hObject, eventdata, handles)
% hObject    handle to rno (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of rno as text
%        str2double(get(hObject,'String')) returns contents of rno as a double
mno=(get(hObject,'String'));
if isnan(mno)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
handles.rdata.rno=mno;
guidata(hObject,handles)




% --- Executes during object creation, after setting all properties.
function rno_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rno (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
```

```
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


function rsex_Callback(hObject, eventdata, handles)
% hObject    handle to rsex (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of rsex as text
%        str2double(get(hObject,'String')) returns contents of rsex as a double

msex=(get(hObject,'String'));
if isnan(msex)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
handles.rdata.rsex=msex;
guidata(hObject,handles)



% --- Executes during object creation, after setting all properties.
function rsex_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rsex (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function rdpart_Callback(hObject, eventdata, handles)
% hObject    handle to rdpart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of rdpart as text
%        str2double(get(hObject,'String')) returns contents of rdpart as a double

mdpart=(get(hObject,'String'));
if isnan(mdpart)
   set(hObject, 'String', 0);
   errordlg('Input must be a number','Error');
end
handles.rdata.rdpart=mdpart;
guidata(hObject,handles)



% --- Executes during object creation, after setting all properties.
function rdpart_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to rdpart (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
   set(hObject,'BackgroundColor','white');
else
   set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function rqualifi_Callback(hObject, eventdata, handles)
% hObject    handle to rqualifi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of rqualifi as text
%        str2double(get(hObject,'String')) returns contents of rqualifi as a double
mqualifi=(get(hObject,'String'));
if isnan(mqualifi)
   set(hObject, 'String', 0);
   errordlg('Input must be a number','Error');
end
handles.rdata.rqualifi=mqualifi;
guidata(hObject,handles)
```

```
% --- Executes during object creation, after setting all properties.
function rqualifi_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rqualifi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end



function rresult_Callback(hObject, eventdata, handles)
% hObject    handle to rresult (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



% --- Executes during object creation, after setting all properties.
```

```
function rresult_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rresult (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end




function faceregdata_Callback(hObject, eventdata, handles)
% hObject    handle to faceregdata (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ConfMatrixm;
d=[cd '\image\' 'new.jpg'];
pic=double(imread(d));
axes(handles.xpic);
imshow(uint8(pic));

labels=[];
samples=[];
s1=size(d);
while ((size(pic,1)*size(pic,2))>(400*300))
    pic = getpic(pic,1,2);
end;
```

```
[pos,facesamples] = findfaces(pic);


faceNumber=[];
if (isempty(faceNumber))
   faceNumber=size(pos,1);
   i = 1;
end
if i > faceNumber
   noMoreFaces=imread('no_more_faces.jpg','jpg');
   imshow(noMoreFaces);


else
   scale=pos(i,3);
   bigy=double(pos(i,1))*scale;
   bigx=double(pos(i,2))*scale;
   binpic(bigy:bigy+20*scale,bigx:bigx+20*scale)=0;
   if (bigy>2*scale)
      bigy=bigy-2*scale;
   end

   if (bigx>scale)
      bigx=bigx-scale;
   end



   im=pic(bigy:bigy+19*scale,bigx:bigx+19*scale)/255;
   dim=size(im,1);
   im_exp = pic(max(bigy-2*scale,1):min(bigy+23*scale,size(pic,1)),max(bigx-
1*scale,1):min(bigx+21*scale,size(pic,2)))/255;
```

```
  %im = histeq(im,255);
  im = resample(im_exp',50,dim);
  im = resample(im',50,dim);
  axes(handles.xface);
  imshow(im);

end
[nosey,nosex]=locate(im,1);
[mouthy,mouthx]=locate(im,2);
[lefteyey,lefteyex]=locate(im,3);
[righteyey,righteyex]=locate(im,4);
[nby,nbx]=locate(im,5);
coord=[nosey,nosex;mouthy,mouthx;lefteyey,lefteyex;righteyey,righteyex;nby,nbx];
nose = reshape(im(coord(1,1):coord(1,1)+17,coord(1,2):coord(1,2)+12),234,1);
mouth = reshape(im(coord(2,1):coord(2,1)+12,coord(2,2):coord(2,2)+26),351,1);
lefteye = reshape(im(coord(3,1):coord(3,1)+14,coord(3,2):coord(3,2)+14),225,1);
righteye = reshape(im(coord(4,1):coord(4,1)+14,coord(4,2):coord(4,2)+14),225,1);
nb = reshape(im(coord(5,1):coord(5,1)+13,coord(5,2):coord(5,2)+15),224,1);
tset=[nose;mouth;lefteye;righteye;nb];
Samples=[samples tset];

load Tm;
%Samples=samples;
[ClassRate,DecisionValue,Ns,ConfMatrixm,PreLabels]= SVMTest(Samples,1,
AlphaY,SVs,Bias,Parameters, nSV,nLabel)
F1=find(ConfMatrixm(1,:)==1);
yd=datar(F1);
y1='N';
ydd=uint8(yd{1});
```

```
if(uint8(y1(1))==ydd(1))
       rr='does not match';
       set(handles.rresult,'string',rr)
else
     rr='match';
     set(handles.rname,'string',yd(2));
     set(handles.rno,'string',yd(1));
     set(handles.rsex,'string',yd(3));
     set(handles.rdpart,'string',yd(4));
     set(handles.rqualifi,'string',yd(5));
     set(handles.rresult,'string',rr);
  end


   function readimdata_Callback(hObject, eventdata, handles)
   % hObject    handle to readimdata (see GCBO)
   % eventdata  reserved - to be defined in a future version of MATLAB
   % handles    structure with handles and user data (see GUIDATA)
   obj = videoinput('winvideo', 1);
   preview(obj);
   pause(10)
   im =(getsnapshot(obj));
   closepreview(obj);
   delete(obj);
   axes(handles.xpic);
   imshow(im);
   imwrite(im,['image\' handles.rdata.rfname '.jpg']);
```

```
function rfname_Callback(hObject, eventdata, handles)
% hObject    handle to rfname (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of rfname as text
%        str2double(get(hObject,'String')) returns contents of rfname as a double

mfname=(get(hObject,'String'));
if isnan(mfname)
    set(hObject, 'String', 0);
    errordlg('Input must be a number','Error');
end
handles.rdata.rfname=mfname;
guidata(hObject,handles)



% --- Executes during object creation, after setting all properties.
function rfname_CreateFcn(hObject, eventdata, handles)
% hObject    handle to rfname (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
```

```
end

% --- Executes during object creation, after setting all properties.
function adddata_CreateFcn(hObject, eventdata, handles)
% hObject    handle to adddata (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called


% --- Executes during object creation, after setting all properties.
function facetestdata_CreateFcn(hObject, eventdata, handles)
% hObject    handle to facetestdata (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% --- Executes during object creation, after setting all properties.
function readimdata_CreateFcn(hObject, eventdata, handles)
% hObject    handle to readimdata (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

function [pos,facesamples] = findfaces(im)
pic = getpic(double(im),1,1);
pos=[];% pos=[y x scale]
facesamples=[];
binpic = ones(size(pic));

% finds "facelike" pictures and saves their positions along with scale in pos

for i= [1.9 2.3 2.8 3.5 4.3 5.1 6 7 8]
```

```
nom=10; den=10*i;
cur_pic=getpic(pic,nom,den);
cur_bin = getpic(binpic,nom,den);
cur_bin = cur_bin>0.5;
if (i<5)
cur_bin(:,2:2:size(cur_bin,2))=0;
cur_bin(2:2:size(cur_bin,1),:)=0;
end;
% if picture is too small go to bigger size (next loop)
if ((size(cur_pic,1)<20)|(size(cur_pic,1)<20))
   continue;
end
   % identifie "small face" with facepos]
[p,s]=facepos(cur_pic,cur_bin);

% choosing the most likly to be faces (we can fet positive results from facepos for
images 2 pixels apart
% for example, will pick only one from them
if (size(p,1) == 0)
   continue;
end

[p,s]=facelocation(p,s);
facesamples = [facesamples s];
for xx=1:size(p,1)
   starty = (p(xx,1)-10)*i;
   if (starty<1)  starty=1;end;
   endy = p(xx,1)*i+15*i;
   startx = (p(xx,2)-5)*i;
```

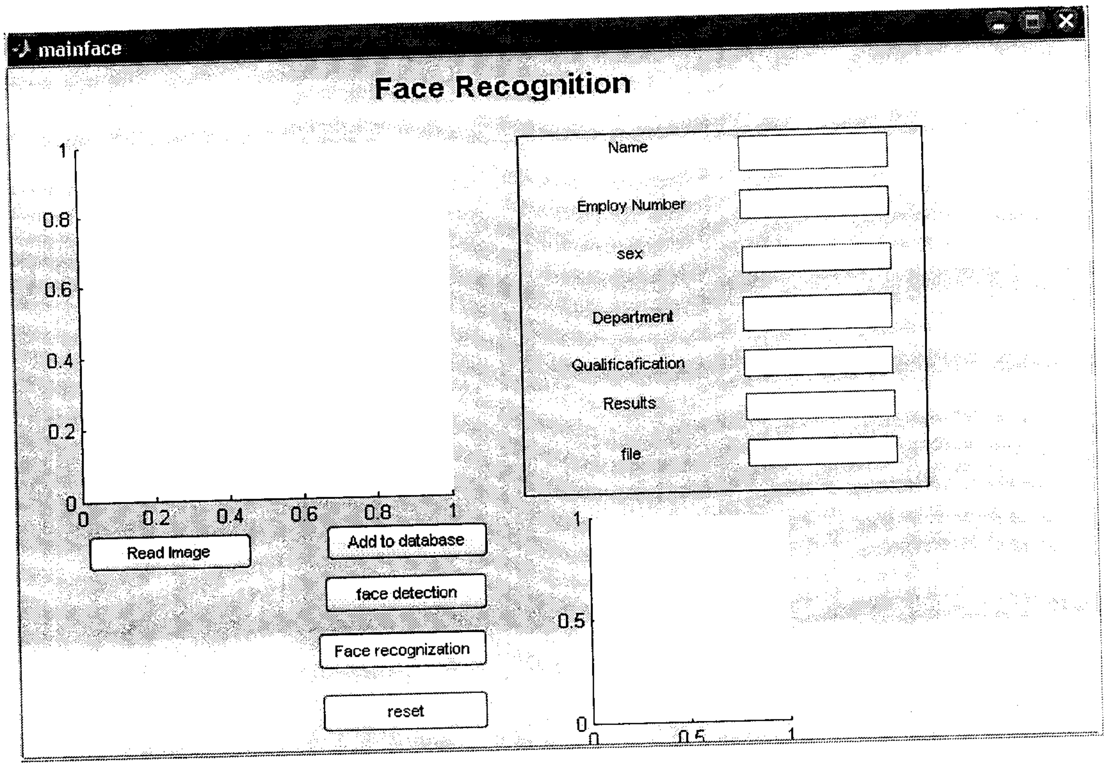
```
    if (startx<1) startx=1; end;
    endx = (p(xx,2)+15)*i;
   % binpic(starty:endy,startx:endx)=0;
  end

  if size(p,1) > 0
     tmp = [p i*ones(size(p,1),1)];
     pos = [pos;tmp];
   end
end


% a face  can be recognized in several scales, order to get the picture in the "best" scale
% we list them in a descending order by the SVM
posScore=[];
load FC;
for j=1:size(pos,1)
   [l,sc] = osuSVMclass(facesamples(:,j),FC.n, FC.alpha, FC.sv,FC.b, FC.params);
   posScore=[posScore;sc];
 end
 clear FC;
 [dummy,newPos]=sort(posScore);
 newPos=newPos(size(newPos,1):-1:1,:);
 newPos=newPos';
 pos=pos(newPos,:);
 facesamples=facesamples(:,newPos);



  function [y,x,val] = locate(im,facepart)
```

```
% this function locates a face part (left eye, right eye, nose, mouth)
% in a face
% reminder: in the eyes, noses ,mouth and nb SVM the picture should be in [0-1] scale
% load SVM parameters
load EC;
x=[];
y=[];
if (facepart == 1)
    start = round([15 10]./[50 50].*size(im));
    stop  = round([27 35]./[50 50].*size(im));
    compw=13;comph=18;
    C=NC;
    clear NC EC NBC MC;
  end;
  if (facepart == 2)
    start = round([25 5]./[50 50].*size(im));
    stop  = round([37 22]./[50 50].*size(im));
    if size(im,2) < start(1,2)+27, start(1,2)=size(im,2)-27;
    end
    comph = 13;
    compw = 27;
    C=MC;
    clear NC EC NBC MC;


  end



  if (facepart == 3)
     start = [1 1];
  stop  = round([20 16]./[50 50].*size(im));
  comph=15;compw=15;
  C=EC;
  clear NC EC NBC MC;
end

if (facepart == 4)
   start = round([0 25]./[50 50].*size(im)+[1 1]);
   stop  = round([20 35]./[50 50].*size(im));
   if size(im,2) < start(2)+15, start(2)=size(im,2)-15;
   end
   comph=15;compw=15;
   C=EC;
   clear NC EC NBC MC;
 end

 if (facepart == 5)
    start = round([0 10]./[50 50].*size(im)+[1 1]);
    stop  = round([12 25]./[50 50].*size(im));
    comph=14;compw=16;
    C=NBC;
    clear NC EC NBC MC;
  end
  search_im = uint8(im(start(1):stop(1)+comph-1,start(2):stop(2)+compw-1)*255);
  search_pos = zeros(size(search_im));
  search_pos(1:1:size(search_im,1),1:1:size(search_im,2))=1;
  search_pos = uint8(search_pos);
  [test,y,x]=comphisteq(search_im,search_pos,comph,compw);
  y = double(y)+start(1)-1;
```

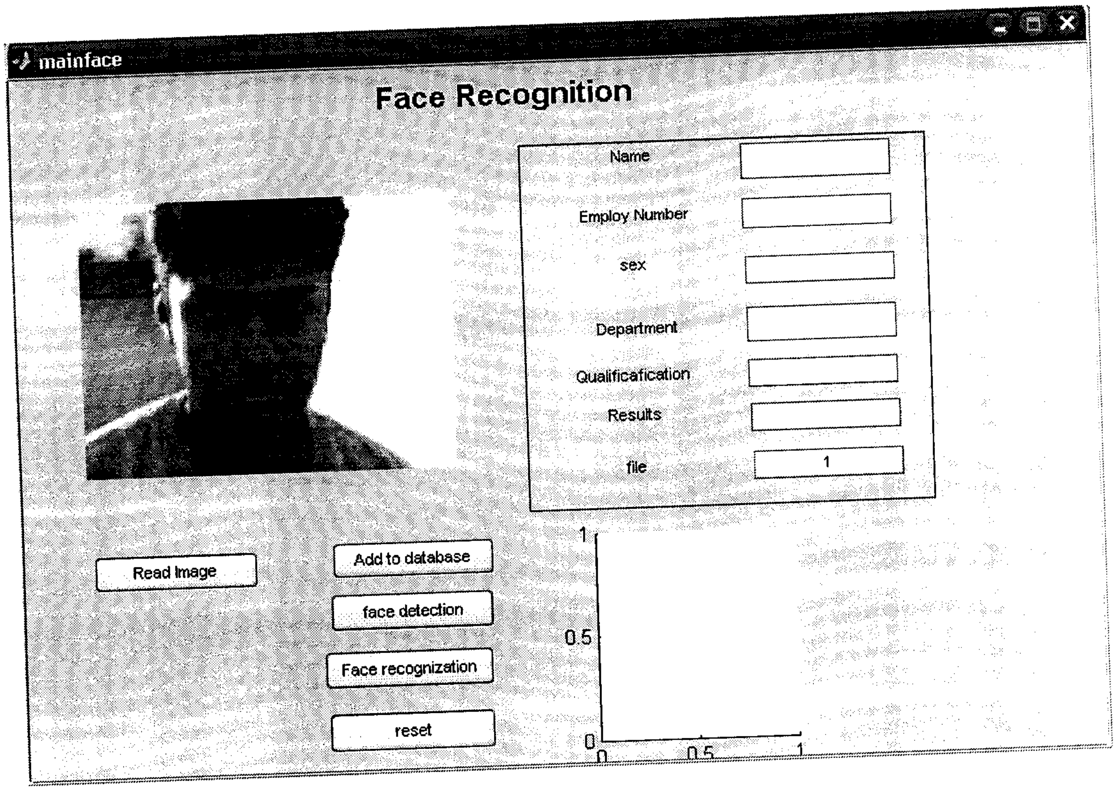
```
x = double(x)+start(2)-1;
test = double(test)/255;
[l,sc] = osuSVMclass(test,C.n, C.alpha, C.sv,C.b, C.params);

[val,linpos] = max(sc);
x = double(x(linpos));
y = double(y(linpos));
%search_pos = zeros(size(search_im));
%search_pos(y-1:1:y+1,x-1:1:x+1)=1;
% search_pos = uint8(search_pos);
%[test,y,x]=comphisteq(search_im,search_pos,comph,compw);
%y = double(y)+start(1)-1;
%x = double(x)+start(2)-1;
%test = double(test)/255;
%[l,sc] = osuSVMclass(test,C.n, C.alpha, C.sv,C.b, C.params);
%[val,linpos] = max(sc);
%x = double(x(linpos));
%y = double(y(linpos));
```

10.2 SAMPLE OUTPUT

FRONT SCREEN



CAPTURING IMAGE



Name

Employ Number

sex

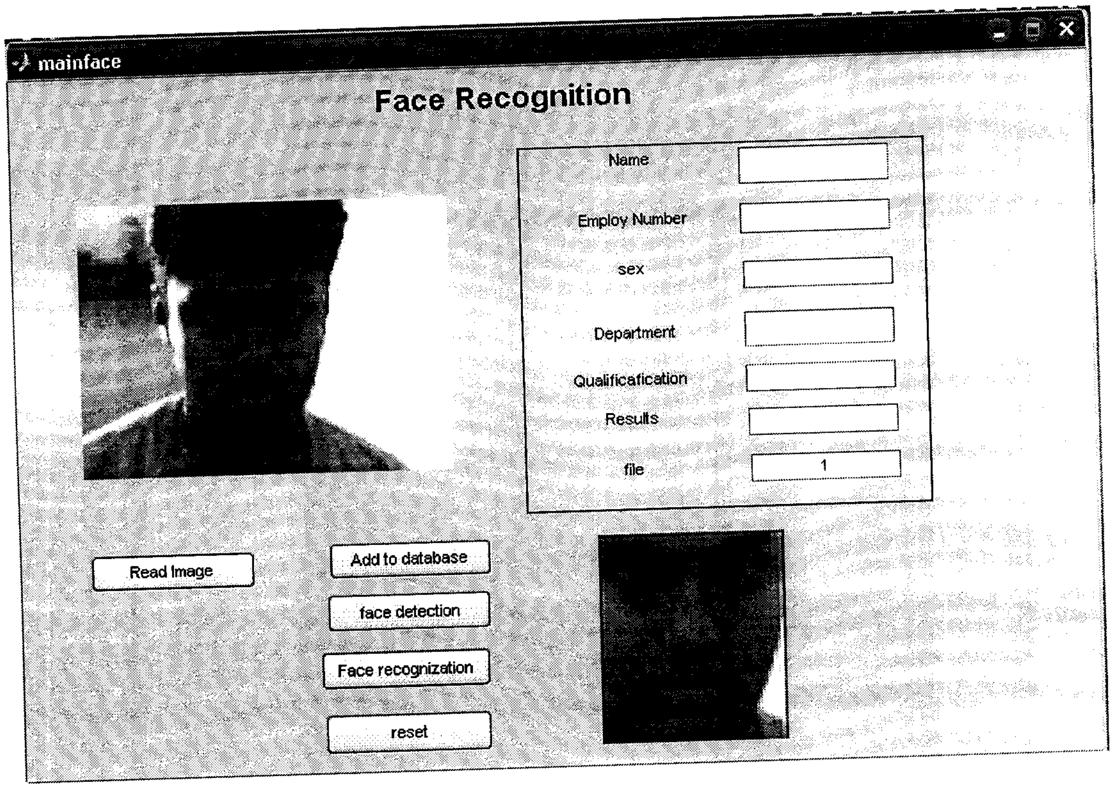
Department

Qualificafication

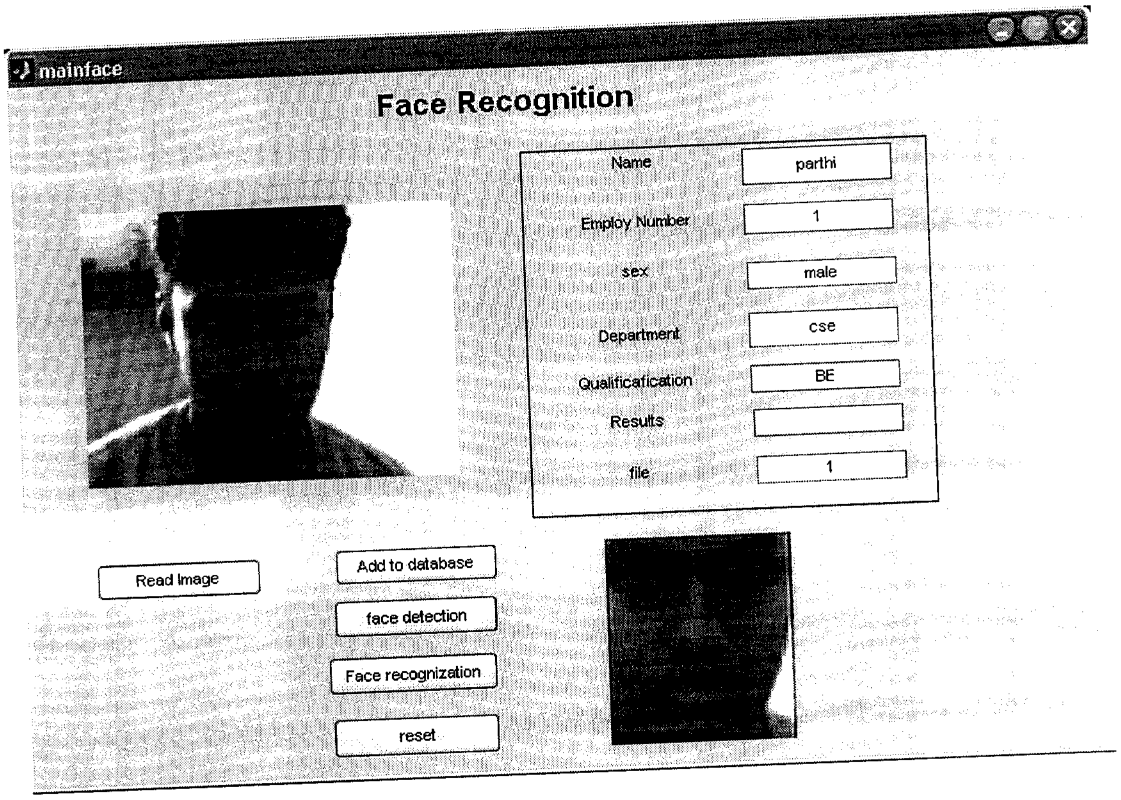
Results

file 1

FACE DETECTION

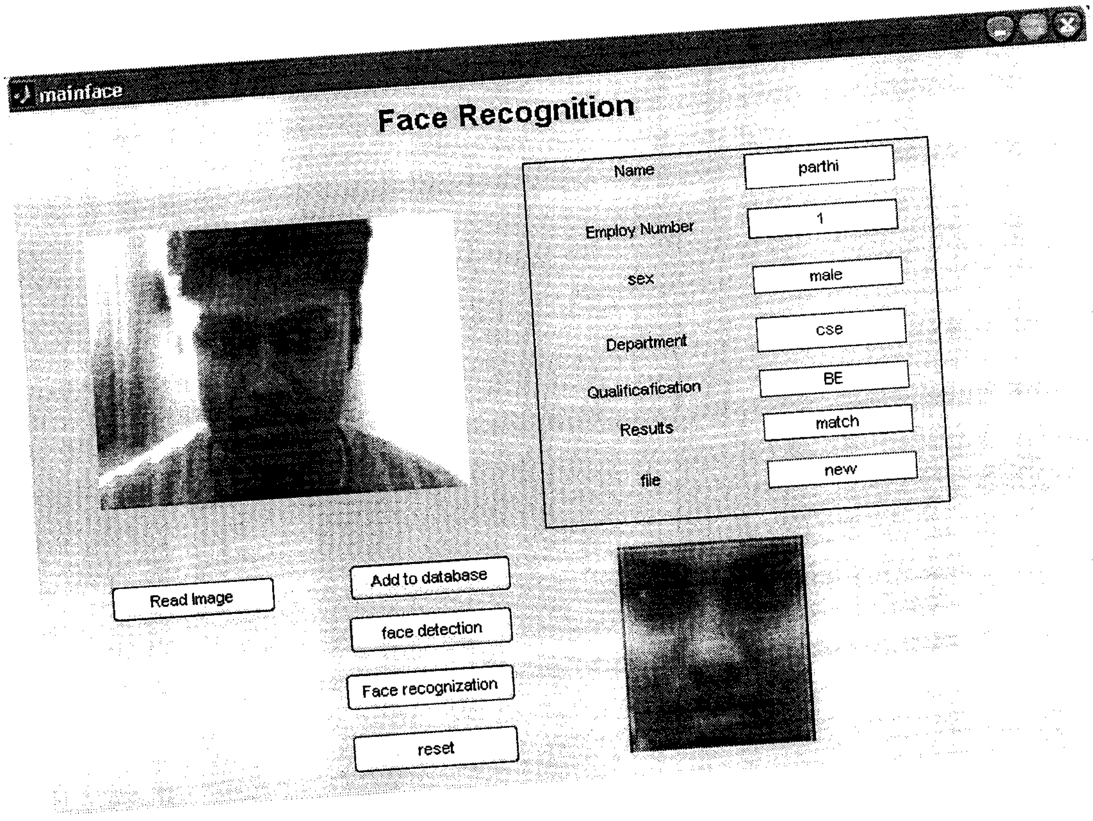


DATABASE ADDITION



Name	parthi
Employ Number	1
sex	male
Department	cse
Qualificafication	BE
Results	
file	1

FACE RECOGNITION



Name	parthi
Employ Number	1
sex	male
Department	cse
Qualificafication	BE
Results	match
file	new

P-1599

11. REFERENCES

* “ **Learning and Recognizing Faces : From still Images to Video Sequences** “,
Abdenour Dadid , Oulu University Press , 2005 Edition.

* “ **The Local Binary Pattern Approach To Texture Analysis –Extensions and Applications** “ , Topi Maenpaa, Oulu University Press, 2003 Edition.

* “ **Color Texture Classification with Color Histograms and Local Binary Patterns** “,
Matti Pietikainen , Topi Maenpaa , Jaakko Viertola, Oulu University Press, 2003 Edition.

* “ **SVM Kernel-Based Visualization and Classification** “, Zhou Nina , 2003 Edition.

* **SVM Classifier Matlab Toolbox** based on J.Ma ,Y.Zhao and S.Anhalt ‘s LiBSVM ,
2004.

* “ **Face Recognition with Support Vector Machines And 3D Head Models** “ ,
Jennifer Huang , Volker Blanz , Bernd Heisle, 2003.

* www.csie.ntu.edu.tw

* www.MathWorks.com

* www.osusvm.com