Abstract

Abstract :

We propose a system for "3D Face Recognition and Detection" that describes an systematic approach by taking 3D image as input than it is converted to an equivalent 2D image & then by applying an algorithm for 2D Face Detection & Recognition.

For 3D to 2D conversion, we read the 3D ".wrl" file format and then by extracting x & y axis & eliminating z-axis and then saving this image into it's equivalent 2d image, which then will be used for Face Detection and Recognition.

Face detection is a preprocessing step for face Recognition algorithms. It is the localization of face/faces in an image or image sequence. Face detection is a challenging computer vision problem because of lighting conditions, a high degree of variability in size, shape, background, color, etc. The existing methods for face detection can be divided into image based methods and feature based methods. We have used an intermediate system, using a boosting algorithm to train a classifier which is capable of processing image rapidly while having high detection rates. The family of simple classifiers contains simple rectangular wavelets which are reminiscent of the Haar basis. Their simplicity and a new image representation called Integral Image allow a very quick computing of these Haar-like features. For this, classifiers with an increasingly complexity are combined sequentially. This improves both, the detection speed and the detection efficiency. The detection of faces in input images is preceded using a scanning window at different scales which permits to detect faces of every size without re-sampling the original image. On the other hand, the structure of the final classifier allows a real time implementation of the detector.

The detected face varies in rotation, brightness, size, and etc. in different images even for the same person. Those features are independent of face features and will affect the recognition rate significantly. One method to solve the problem is to normalize the detected faces. After the face detection, face normalization process has been performed. It's main objective is to normalize the detected faces which helps in increasing the face recognition rate.

Face recognition (FR) is the preferred mode of identity recognition by humans: It is natural, robust and unintrusive. The Normalized images are then compared with images in Database. If Image is found then appropriate image is displayed or else error message is displayed.

About Project

1. INTRODUCTION:

1.1. Problem Definition :

Face Detection:

Face detection is an essential application of visual object detection and it is one of the main components of face analysis and understanding with face localization and face recognition.

Automatic face detection is a complex problem which consists in detecting one or many faces in an image. Faces are non rigid objects. Face appearance may vary between two different persons but also between two photographs of the same person, depending on the lightning conditions, the emotional state of the subject and pose. That is why so many methods have been developed during last few year's.

Face Recognition:

A facial recognition system is a computer application for automatically identifying or verifying a person from a digital image or a video frame from a video source. One of the ways to do this is by comparing selected facial features from the image and a facial database.

It is typically used in security systems and can be compared to other biometrics such as finger print or eye iris recognition system.

Face recognition is interesting to study because it is an application area where computer vision research is being utilized in both military and commercial products. Much effort has been spent on this problem, yet there is still plenty of work to be done.

Basic research related to this field is currently active. Often, practical applications can grow out of improvements in theoretical understanding and it seems that this problem will continue to demonstrate this growth.

Personally, we are interested in this project because it's a pattern recognition problem in which humans are very adept, whereas it can be quite challenging to teach a machine to do it. The intermediate and final visual results are interesting to observe in order to understand failures and successes of the various approaches.

Face recognition is challenging because it is a real world problem. The human face is a complex, natural object that tends not to have easily (automatically) identified edges and features. Because of this, it is difficult to develop a mathematical model of the face that can be used as prior knowledge when analyzing a particular image.

Applications of face recognition are widespread. Perhaps the most obvious is that of human computer interaction. One could make computers easier to use if when one simply sat down at a computer terminal, the computer could identify the user by name and automatically load personal preferences. This identification could even be useful in enhancing other technologies such as speech recognition, since if the computer can identify the individual who is speaking, the voice patterns being observed can be more accurately classified against the known individual's voice.

Human face recognition technology could also have uses in the security domain. Recognition of the face could be one of several mechanisms employed to identify an individual. Face recognition as a security measure has the advantage that it can be done quickly, perhaps even in real time, and does not require extensive equipment to implement. It also does not pose a particular inconvenience to the subject being identified, as is the case in retinal scans. It has the disadvantage, however, that it is not a foolproof method of authentication, since human face appearance is subject to various sporadic changes on a day-to-day basis (shaving, hair style, acne, etc…), as well gradual changes over time (aging). Because of this, face recognition is perhaps best used as an augmentation for other identification techniques.

A final domain in which face recognition techniques could be useful is search engine technologies. In combination with face detection systems, one could enable users to search for specific people in images. This could be done by either having the user provide an image of the person to be found, or simply providing the name of the person for well-known individuals. A specific application of this technology is criminal mug shot databases. This environment is perfectly suited for automated face recognition since all poses are standardized and lighting and scale are held constant. Clearly, this type of technology could extend online searches beyond the textual clues that are typically used when indexing information.

A newly emerging trend, claimed to achieve previously unseen accuracies, is 3D face recognition. This technique uses 3D sensors to capture information about the shape of a face. This information is then used to identify distinctive features on the surface of a face, such as the contour of the eye sockets, nose, and chin.

1.2. Significance:

Recent developments in computer technology and the call for better security applications have brought 2D Face Recognition & Detection into focus. A biometric is a physical property; it cannot be forgotten or mislaid like a password, and it has the potential to identify a person in very different settings: a criminal entering an airport, an unconscious patient without documents for identification, an authorized person accessing a highly-secured system. Be it for purposes of security or human–computer interaction, there is wide application to robust biometrics.

Two different scenarios are of primary importance. In the verification (Authentication) scenario, the person claims to be someone, and this claim is verified by ensuring the provided biometric is sufficiently close to the data stored for that person. In the more difficult recognition scenario, the person is searched in a database. The database can be small (e.g. criminals on the wanted list) or large
(E.g. photos on registered ID cards). The unobtrusive search for a number of people is called *screening*.

The signature and handwriting have been the oldest biometrics, used in the verification of authentication of documents. Face image and the fingerprint also have a long history, and are still kept by police departments all over the world. More recently, voice, gait, retina and iris scans, hand print, and 3D face information are considered for biometrics. Each of these have different merits, and applicability. When deploying a biometrics based system, we consider its accuracy, cost, ease of use, ease of development, whether it allows integration with other systems, and the ethical consequences of its use. Two other criteria are susceptibility to spoofing (faking an identity) in a verification setting, and susceptibility to evasion (hiding an identity) in a recognition setting.

The purpose of the present study is to discuss the merits and drawbacks of 2D face information as a biometric, and review the state of the art in 2D face recognition. Two things make face recognition especially attractive for our consideration. The acquisition of the face information is easy and non-intrusive, as opposed to iris and retina scans. This is important if the system is going to be used frequently, and by a large number of users. The second point is the relatively low privacy of the information; we expose our faces constantly, and if the stored information is compromised, it does not lend itself to improper use like signatures and fingerprints would. The drawbacks of 3D face recognition include high cost and decreased ease-of-use for laser sensors, low accuracy for other acquisition types, and the lack of sufficiently powerful algorithms.

Face recognition (FR) is the preferred mode of identity recognition by humans: It is natural, robust and unobtrusive. With the availability of cheaper acquisition methods, 2D face recognition can be a way out of these problems regarding conventional authentication processes. We review the relevant work on 2D face recognition here, and discuss merits of different representations and recognition algorithms.

2D Face recognition and Detection has a natural place in the present and the future environment because it's unobtrusive and passive in nature.

Our goal at the end of the project is to develop Human Face Recognition software which works efficiently on input images that are taken either by webcam (real time) or from harddisk and which is also simple to understand.

1.3. SCOPE :

- Take a 3d image file.
- Read the 3d image file as text.
- Extract x, y, values and convert it into a 2d image cloud.
- Thus now we get a 2d image.
- Apply face detection and get the detected face.
- If the face is tilted by some angle then normalized it and save it.
- Find the correlation coefficient between the normalized image and all the images in the database.
- If the correlation coefficient is above the threshold value than display the matched image of the database.
- If the correlation coefficient is below the threshold value then display not recognized.

1.4. Existing System Overview :

2D Face Recognition :

Every face or image that we can see on machine is in its 2 dimensional form. There are many different technologies available today to uniquely identify a person's identity. Many of which like Password/PIN known as Personal Identification Number systems are the most common in practice today. However these systems have their own intrinsic drawbacks. Passwords can be forgotten and worse if they are lost or stolen, person identity can be misused by somebody else. In order to overcome these problems there has been a considerable interest in "biometrics" identification systems, which use pattern recognition techniques to identify people using their unique characteristics. Some of those methods are fingerprints and retina and iris recognition. But these are obtrusive and expensive.

2D face recognition has a natural place in the present and the future environment because it's unobtrusive and passive in nature. It does not restrict the movements of an individual during recognition.

2D face recognition gives best result when image is frontal and effect of illumination is negligible. If image is in profile view i.e. Image is rotated more than 20(degrees) then the recognition rate falls considerably. 2D face recognition algorithm are also prone to illumination effect and recognition rate below threshold value. Depth information is one of the most important parameter which mostly concern about eye, nose etc detection and recognition. We cannot realize depth information and is simply ignored in 2D view.

Advance System Overview

Need for 3D Face Recognition :

A newly emerging trend, claimed to achieve previously unseen accuracies, is 3D. This technique uses cameras to capture an 3D image and provide information about the shape of a face. This information is then used to identify distinctive features on the surface of a face, such as the contour of the eye sockets, nose, and chin.

One advantage of 3D facial recognition is that it is not affected by changes in lighting like other techniques. It can also identify a face from a range of viewing angles, including a profile view.

Even a perfect 3D matching technique could be sensitive to expressions. For that goal a group at the Technion applied tools from metric geometry to treat expressions as isometrics.

One advantage of 3D facial recognition is that it is not affected by changes in lighting like other techniques. It can also identify a face from a range of viewing angles, including a profile view.

Our goal at the end of the project is to capture 3D Human Face, convert it to 2D, Apply Face Detection, Normalization & finally Recognition software which works efficiently and is simple to understand. In addition, we try to implement 3D Face Detection & Recognition for User Authentication Systems too.

1.5. Proposed system :

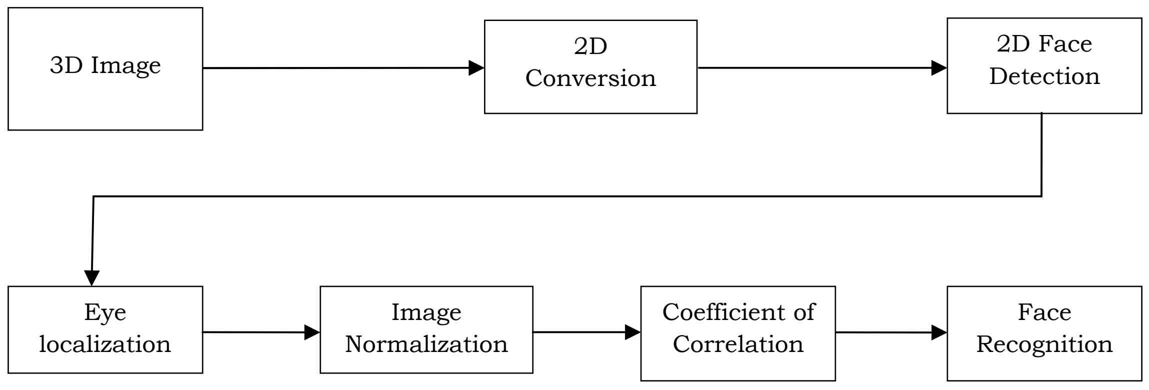


Figure : Block diagram for 3D Face Detection & Recognition

The entire process is divided into four major steps:

- Conversion from 3D to 2D.
- Face detection .
- Eye Localization and Normalization.
- Recognition using Coefficient of Correlation.

The facial image of the person is in 3D and can be of any format that can be used. To read the image in matlab, we access the image i.e. the ".wrl" file serially and retrieve the co-ordinate points of the 3D image which are further stored in an array. It needs to be converted into .mat file, for the same purpose we do wrl to mat conversion and store the image as .mat file.

The conversion into 2D using 'perspective projection method', hence we create 2D database from the available 3D database. Third step is the Face detection of 2D image. Now this 2D sample image is used for detection process. The detected face is then applied foe eye localization and finally Normalization is applied to rotate tilted faces.

The recognition method uses the Coefficient of Correlation. In the Coefficient of Correlation method, Coefficient of Correlation of images is found. This Coefficient of Correlation of image is then compared with the Coefficient of Correlation of images of each persons in the database. This comparison is done using Coefficient of Correlation technique. A threshold value is set and the value is compared with the threshold value. If the value exceeds the threshold limit then it is said to be matched and face is recognized.

Block diagram can explained as follow:

a) 3D image:

The 3D image is a ".wrl" file extension 3d image. It contains X,Y,Z co-Coordinates', Color indices & Vectors.

b) 2D Conversion:

The 3D image is read as a text file. We thus get X, Y, Z Co-ordinates, we truncate the Z co-ordinate and create a point cloud of X, Y. This point cloud is the frontal image of the person.

c) 2d Face detection:

Image pre-processing often takes the form of signal conditioning (such as noise removal, and normalization against the variation of brightness). Filtering of given image is required to achieve it.

In this application, we will employ the face detection method developed by Viola based on an adaboost and cascade algorithm which uses the rectangular Haar features. The selected haar-like features (masks) will effectively represent particular facial features. Using eyes as one of the facial feature for face detection has the advantages of being easy to detect and robust to image noises. After the detection of faces in a given image, extraction of these faces will be carried out.

d) Image Normalization:

The first step in normalization is detection of eyes. We find the co-ordinates of the eyes.

Once we have got the co-ordinates the next step is to find the slope of the line connecting the two eyes balls.

Slope can be given the the following equation

$$\text{Slope} \quad m=|Y2-Y1| \;/\; |X2-X1|;$$

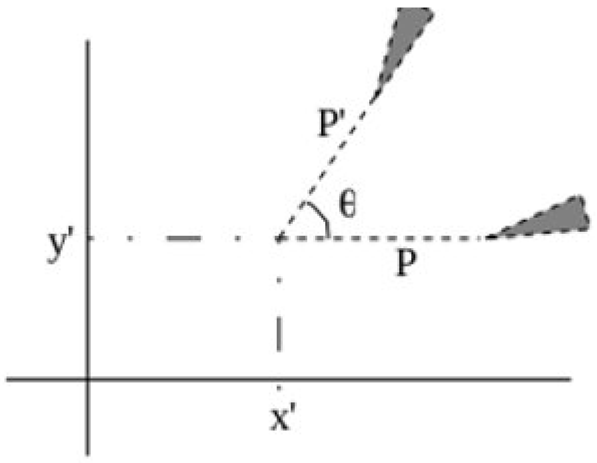
Once we have got the slope we can find the angle of inclination by the

$$\theta = \tan^{-1}(m);$$

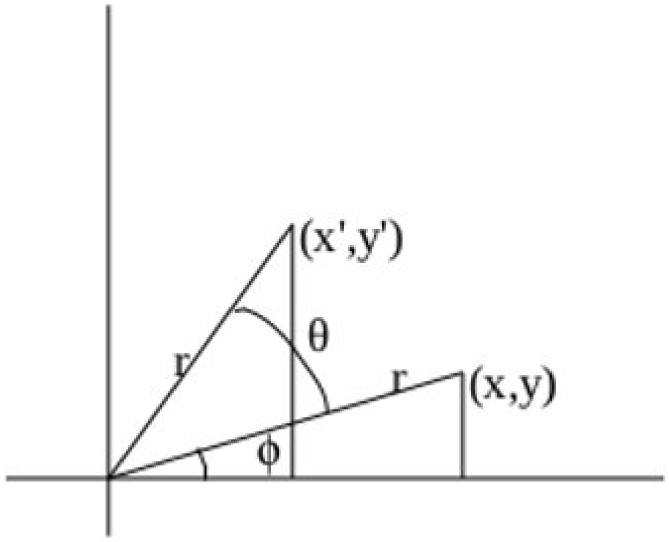
This Theta gives us the information about the angle by which the face is titled.

We then rotate the face by the given angle so that it becomes normalize. For eg: we rotate the image from P' plane to P plane. Due to this the image is normalized.

Figure below shows : Rotation of an object through angle about the pivot point (x’,y’).



Rotation can be performed in the following manner



We first determine the transformation equations for rotation of a point position P when the pivot point is at the coordinate origin. The angular and coordinate relationships of the original and transformed point positions are shown in Fig. above. In this figure, r is the constant distance of the point from the origin, angle θ is the original angular position of the point from the horizontal, and Φ is the rotation angle. Using standard trigonometric identities, we can express the transformed coordinates in terms of angles θ and Φ as

$$x' = r\cos(\theta + \Phi) = r\cos\theta\cos\Phi - r\sin\theta\sin\Phi$$

$$y' = r\sin(\theta + \Phi) = r\cos\Phi\sin\theta + r\sin\Phi\cos\theta$$

e) Coefficient of correlation and Face recognition:

We use the correlation coefficient method in order to perform face recognition. We compare the normalized image with all the images in the database by finding the correlation coefficient between them. Correlation coefficient can be calculated as follows:

$$\text{Corr-coeff} = \Sigma(x-\mu_x)(y-\mu_y) \;/\; (n-1)sx.sy$$

Where,

$$sx = \text{sqrt}\,[\,\Sigma(x-\mu_x)^2 \,/(n-1)\,]$$

$$sy = \text{sqrt}\,[\,\Sigma(y-\mu_y)^2 \,/(n-1)\,]$$

Where x is the value of all pixels of normalized image and y is value of all the pixels of the image to be compared.

If the correlation coefficient is above a threshold value than only it is recognized else it is not recognized. Thus face recognition is done with the help of correlation coefficient.

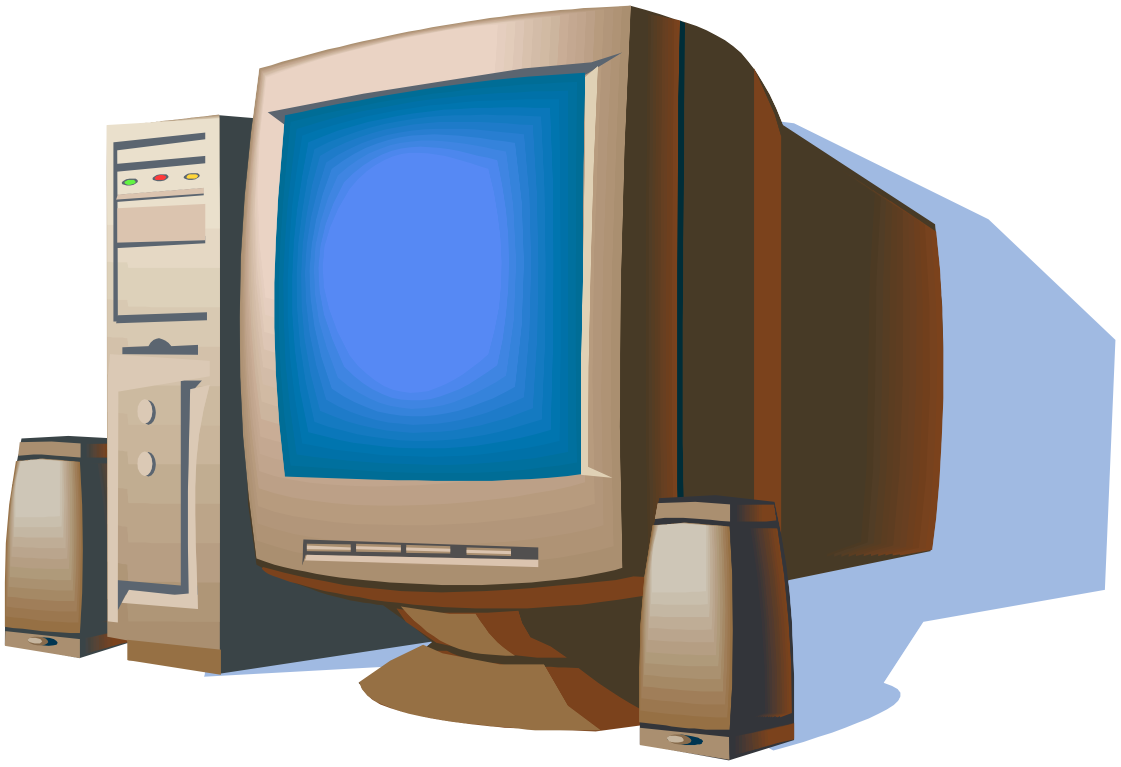
1.6. Assumptions :

- The facial expressions of the person remain constant.
- Database is complete and stores sufficient images
- Ageing characteristics of person is not considered.
- Surrounding characteristics not considered.
- The image is taken in such a way that contour features are detected.
- The physical appearance of the person remains the same eg : specks or bear are not taken into account while detecting contour parts like eyes or while recognizing person.

1.7. Constraints :

- We can only apply rotational movements.
- Accuracy is not 100 percent.
- False Recognition takes place.

System Requirements



1.8. SYSTEM APPLICATION REQUIREMENTS :

1. To make the application independent of folder location in system.
2. To insert file to application either from WebCam or from HardDisk.
3. To insert images to Database as required by user.
4. To design the application as user friendly as possible.
5. To provide extensive Help to the End-user for proper usage.
6. Perform quick processing of the user tasks and also provide high Efficiency.
7. To display error message for unauthorized user and to avoid False recognition.

HARDWARE & SOFTWARE REQUIREMENTS :

This Software and Hardware Requirements Specification provides a complete description of all the specifications of Face Recognition software.

Hardware Requirements :

The minimum required configuration:

- X-86 based Personal Computer (P-IV or above recommended)
- 512 MB RAM
- 80 GB Hard Disk
- SVGA Color Monitor (800 x 600 recommended)
- Mouse
- Multimedia System.

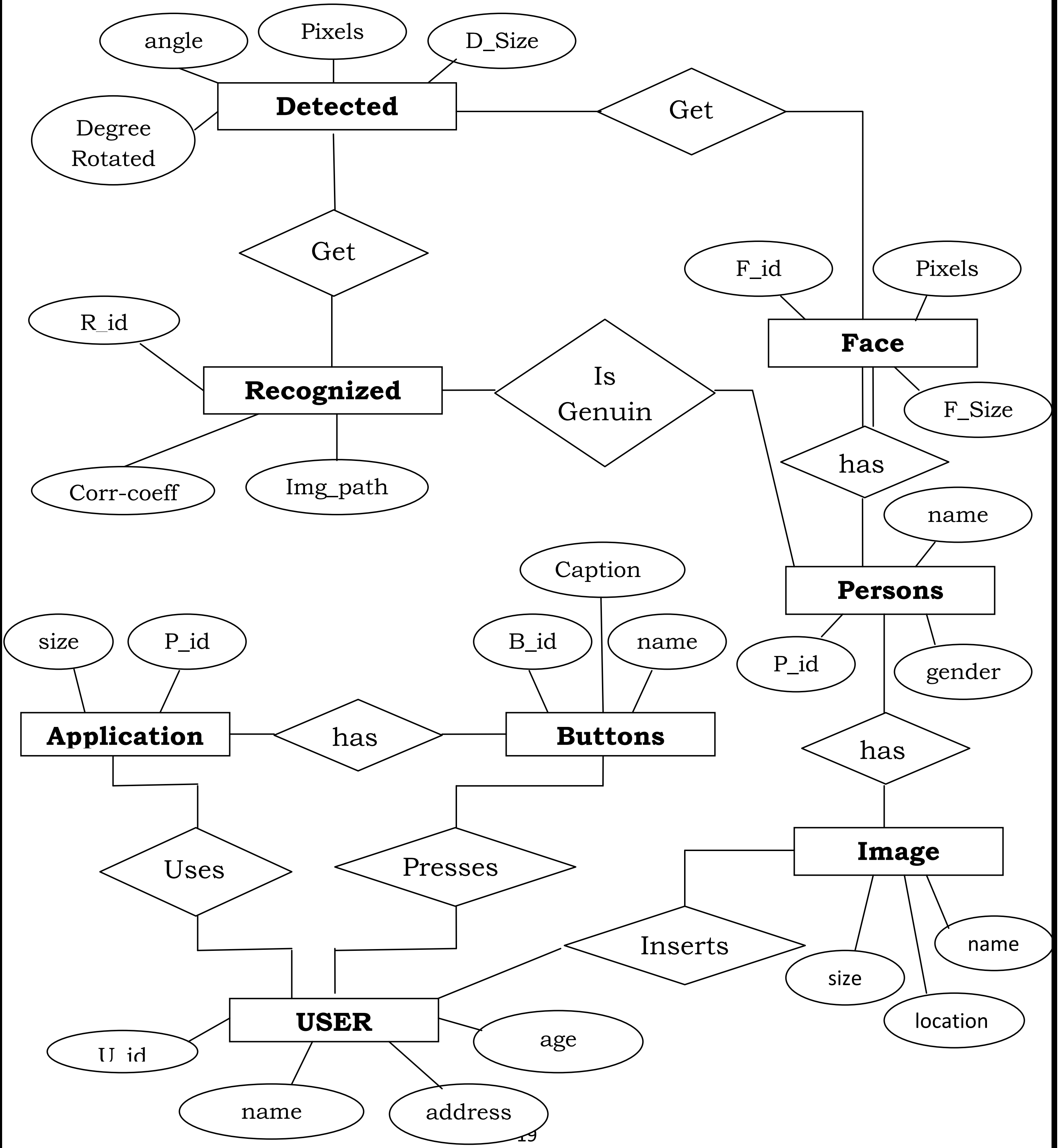
Software Requirements :

- ❖ Compatible Operating Systems:
  - ➢ Windows Vista Home Premium Edition
  - ➢ Windows XP Professional
  - ➢ Windows 2000 Professional
  - ➢ Windows NT/ME
  - ➢ Windows 98
- ❖ MATLAB version 6.0 or 7.0 installed with IP toolbox
- ❖ OpenCV version 1.0 or later.
- ❖ Microsoft Visual C++ 2006 or higher.

System Design

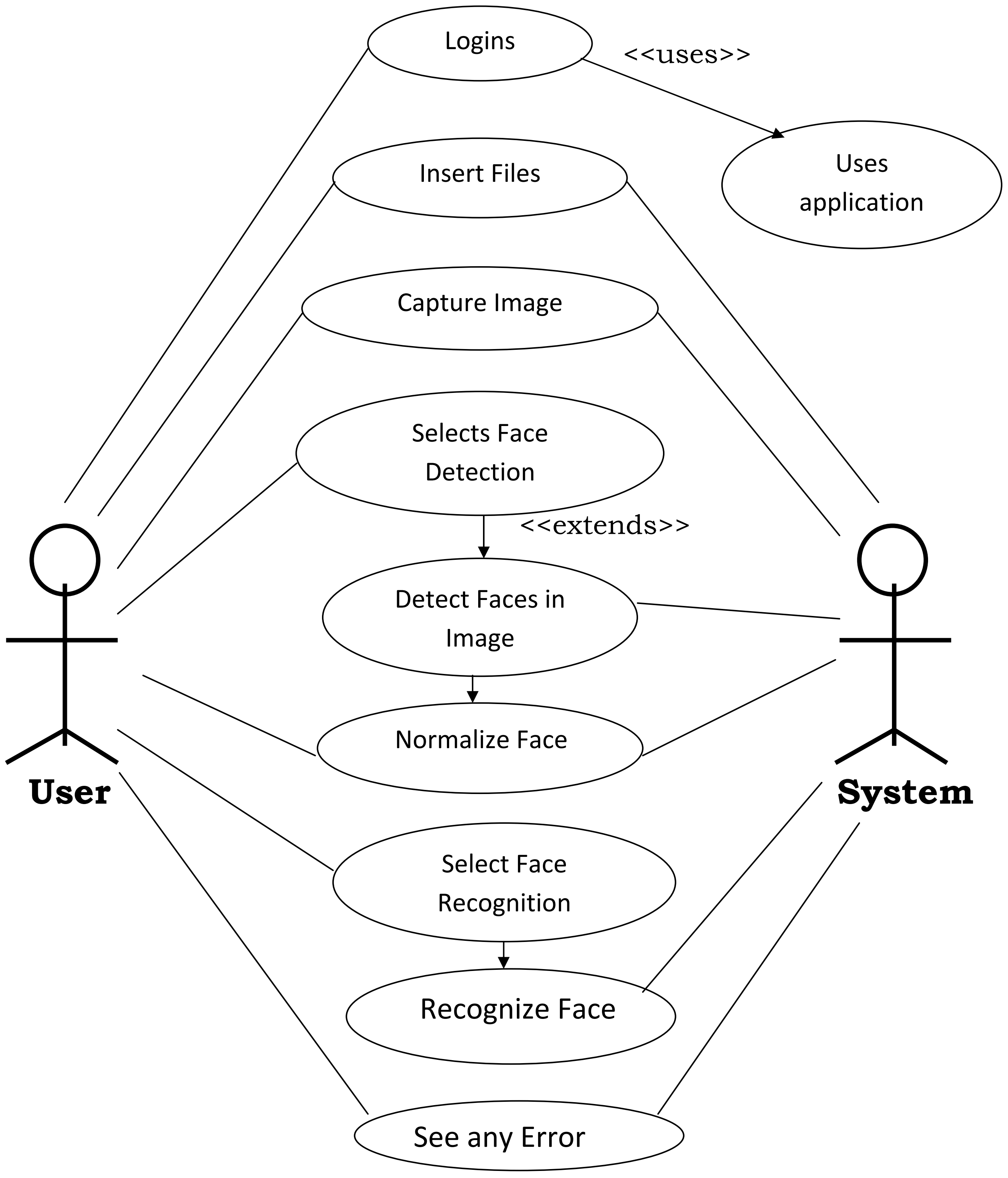
2. SOFTWARE REQUIREMENT SPECIFICATION AND DESIGN :

2.1. ER DIAGRAM :

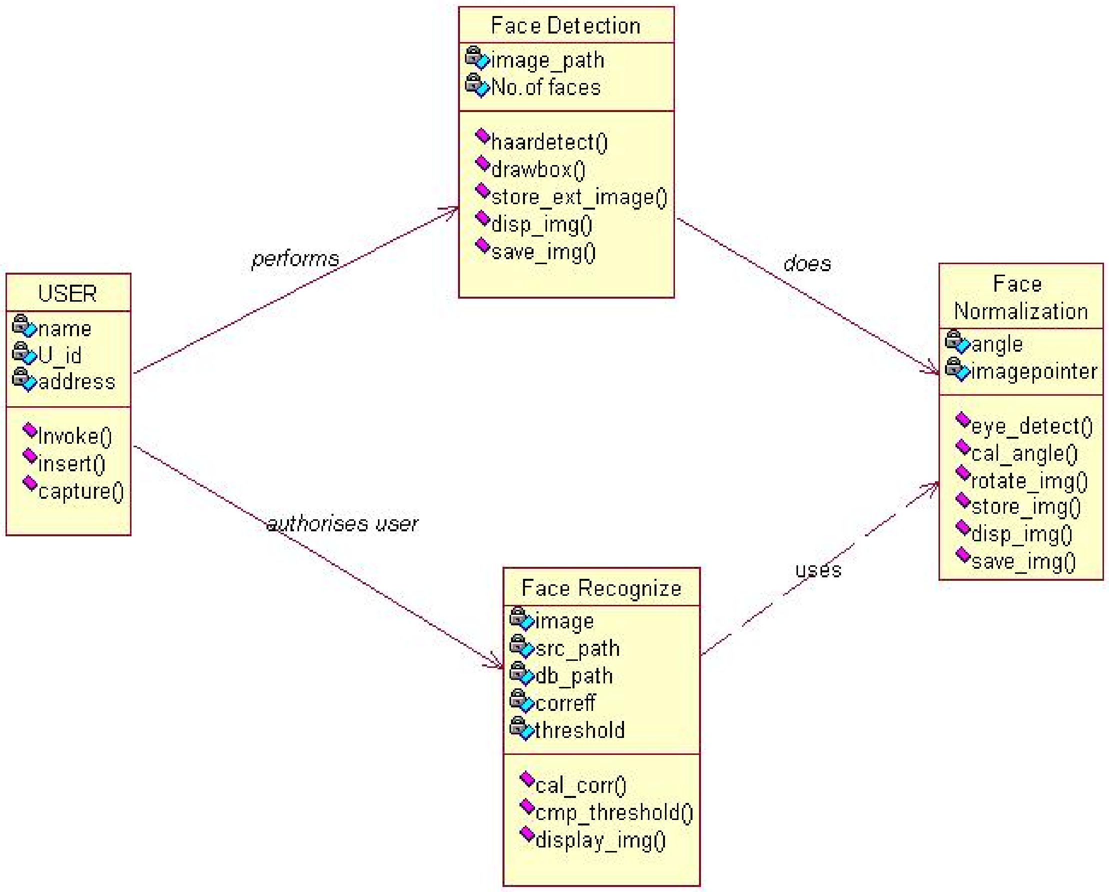


2.2. UML DIAGRAM :

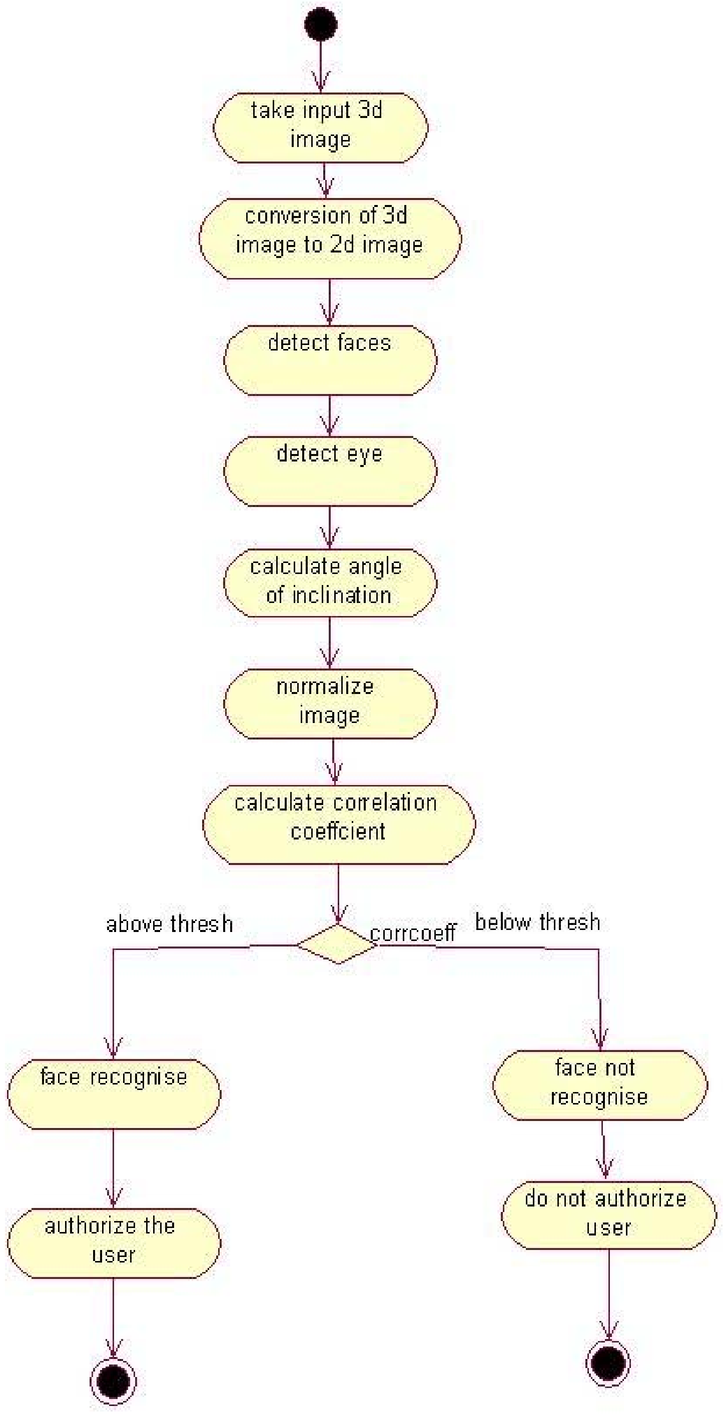
2.2.1. USE-CASE DIAGRAM :



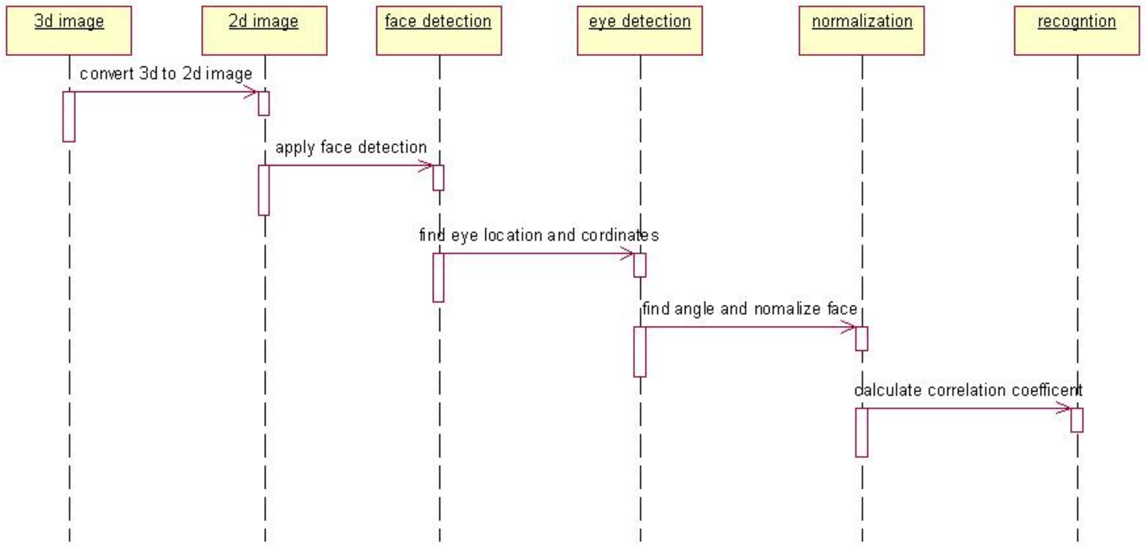
2.2.2. CLASS DIAGRAM :



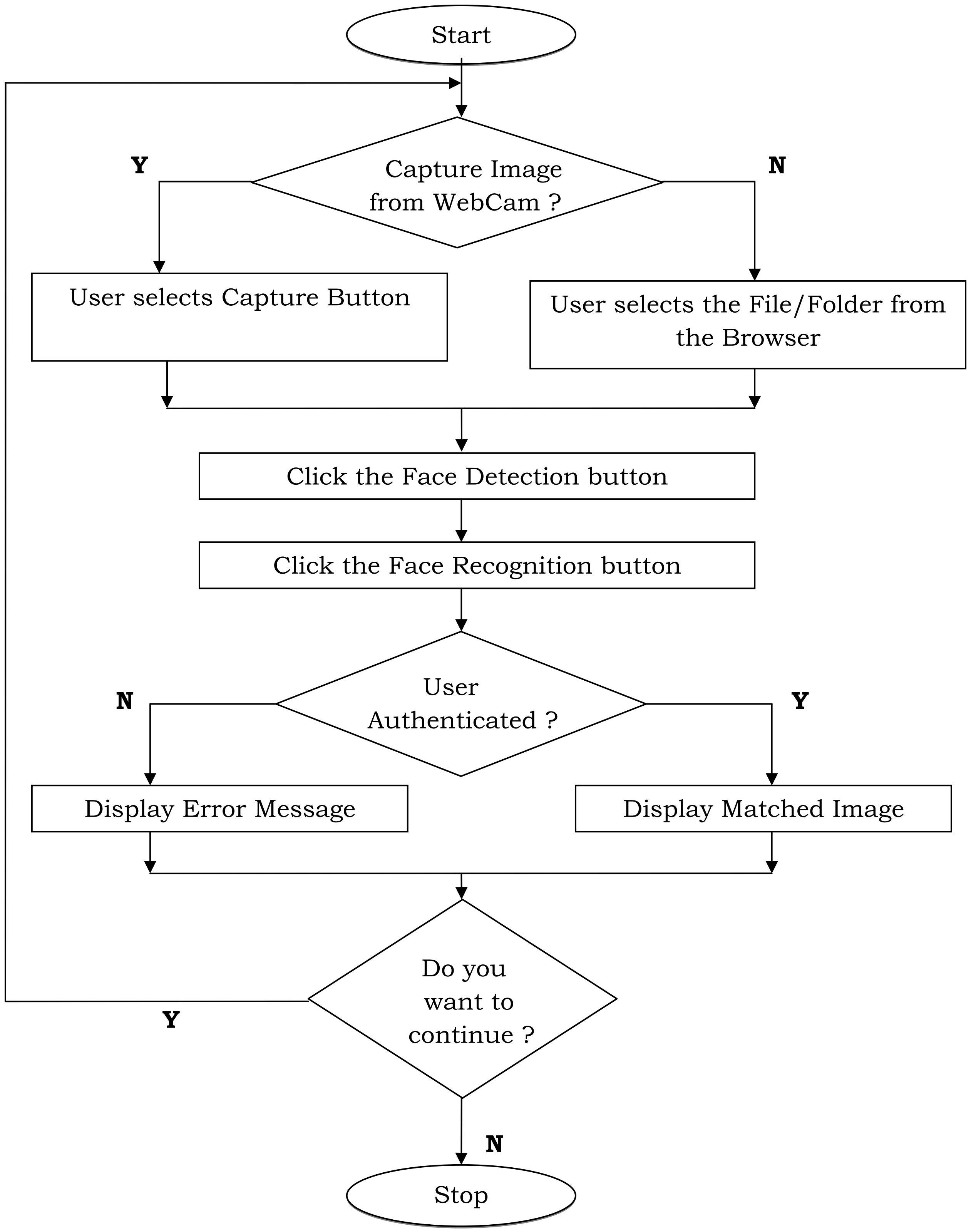
2.2.3. ACTIVITY DIAGRAM:



2.2.4. SEQUENCE DIAGRAM :



2.3. FLOWCHART :


Start
Capture Image from WebCam ?
Y
N
User selects Capture Button
User selects the File/Folder from the Browser
Click the Face Detection button
Click the Face Recognition button
User Authenticated ?
N
Y
Display Error Message
Display Matched Image
Do you want to continue ?
Y
N
Stop

Project Implementation

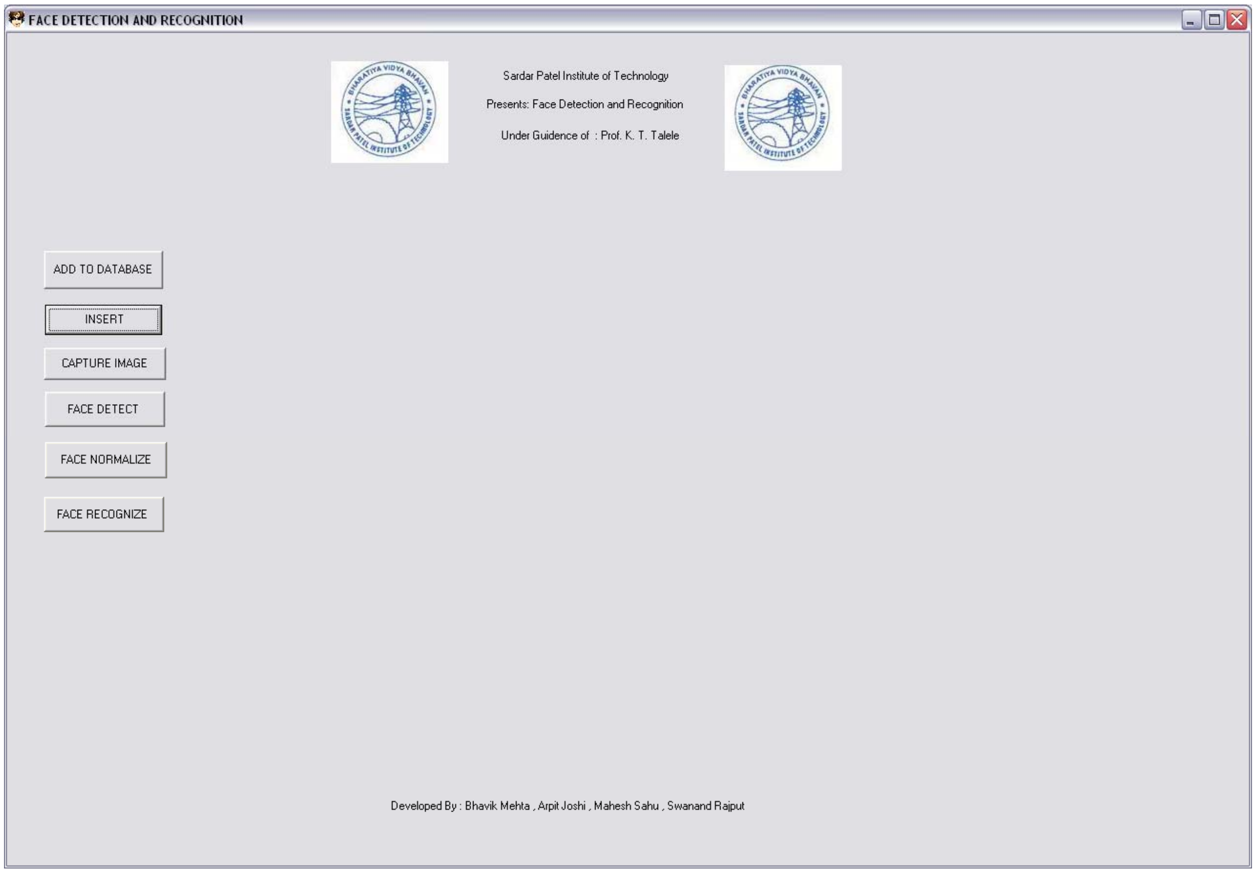
3. CODE WITH REFERENCE TO DESIGN :

3.1. Interface Design :

- **Main Window :**

To start the process of Face Recognition on images

- Install Microsoft Visual C++ on your PC.
- Install OpenCV packages and drivers for webcam.
- Open the FaceDNR.exe in the folder 'FaceDNR'.
- This will show the 'Face Detection AND Recognition' GUI on your screen



Sardar Patel Institute of Technology

Presents: Face Detection and Recognition

Under Guidence of : Prof. K. T. Talele

Developed By : Bhavik Mehta , Arpit Joshi , Mahesh Sahu , Swanand Rajput

Code with Reference to Design :

```
#include "cv.h"
#include "highgui.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>
#include <math.h>
#include <float.h>
#include <limits.h>
#include <time.h>
#include <ctype.h>
#include <fstream.h>
#include "stdafx.h"
#include "FaceDNR.h"
#include "FaceDNRDlg.h"
#include <conio.h>
```

//Code for capturing image from webcam:

```
int CFaceDNRDlg::OnCapture()
{
         int key;
         CvCapture* capture = NULL;

     if(NULL==(capture = cvCaptureFromCAM(-1)))
       {
        printf("\nError on cvCaptureFromCAM");
        return -1;
       }

     cvNamedWindow("Capture", CV_WINDOW_AUTOSIZE);
     cvMoveWindow("Capture", 50, 50);

     for(;;)
       {
        if(NULL==(src=cvQueryFrame(capture))){
        printf("\nError on cvQueryFrame");
        break;
       }

     cvShowImage("Capture", src);
         cvSaveImage("C:/FaceDNR/INPUT/CAM.jpg",src);
         strcpy(input_name, "CAM.jpg");


     key = cvWaitKey(0);

     if(key==0x1b)
     break;
```

```
  }

      cvReleaseCapture(&capture);
      cvDestroyWindow("Capture");

      return 0;


}
```

//Code for loading an input image:

```
void CFaceDNRDlg::OnInsert()
{
      CFile f;
           CString input;
           static char input_path[80];
           char strFilter[] = { "JPG Files (*.jpg)|*.jpg|All Files (*.*)|*.*||" };

           CFileDialog FileDlg(TRUE, ".jpg", NULL, 0, strFilter);

           if( FileDlg.DoModal() == IDOK )
             {
                     if( f.Open(FileDlg.GetFileName(), CFile::modeRead) == FALSE )
                     return;
                     input=FileDlg.GetPathName( );
                     strcpy(input_path, input);
            input=FileDlg.GetFileName( );
                     strcpy(input_name, input);
             }
           else
                     return;

           f.Close();
           char dummy[100] = "C:/FaceDNR/INPUT/";
           strcat(dummy,input_name);

           IplImage* load = cvLoadImage( input_path, 0 );
           cvSaveImage(dummy,load);
           cvNamedWindow( "IMAGE", 1 );
           cvShowImage( "IMAGE", load );
           cvWaitKey( 0 );
           return;

}
```

//**Code for adding an image to database**:

```
void CFaceDNRDlg::Onaddtodatabase()
{

        CFile f;
        CString input;
        char input_addr[80];
        char input_file[80];

        FILE* point;
        char dum[]="C:/FaceDNR/OUTPUT/db.txt";
        point=fopen(dum,"a");
        char strFilter[] = { "JPG Files (*.jpg)|*.jpg|All Files (*.*)|*.*||" };

        CFileDialog FileDlg(TRUE, ".jpg", NULL, 0, strFilter);

        if( FileDlg.DoModal() == IDOK )
          {
                if( f.Open(FileDlg.GetFileName(), CFile::modeRead) == FALSE )
                return;
                input=FileDlg.GetPathName( );
                strcpy(input_addr, input);
                input=FileDlg.GetFileName( );
                strcpy(input_file, input);
          }
        else
                return;

        f.Close();

        char dummy[100] = "C:/FaceDNR/db/";
        strcat(dummy,input_file);
        char ins[100]="\n.\\..\\db\\";
        strcat(ins,input_file);
        IplImage* load = cvLoadImage( input_addr, 0 );
        cvSaveImage(dummy,load);
        fprintf(point,ins);
    fclose(point);
        return;
}
```

//Code for face detection:

```
void CFaceDNRDlg::OnDetect()
{
        CvCapture* capture = 0;
         IplImage *frame, *frame_copy = 0;
         int optlen = strlen("--cascade=");
        cvDestroyWindow("IMAGE");
        cascade_name = "C:/FaceDNR/haarcascade/haarcascade_frontalface_alt2.xml";
        cascade = (CvHaarClassifierCascade*)cvLoad( cascade_name, 0, 0, 0 );
        storage = cvCreateMemStorage(0);
         cvNamedWindow( "INPUT IMAGE", 1 );
        cvNamedWindow( " DETECTED FACES ", 1 );
        CvSize imgSize;
        IplImage* image1;
        char letter='.';
         char temp[]="C:/FaceDNR/INPUT/";

        filename = input_name ? input_name : "lena.jpg";
        len=strlen(filename);
        filename2=(char*)malloc(len+1);
        filename1=(char*)malloc(len+1);
        x=0;
        LOOP:          //subroutine to extract letters which are before dot(.) from filename
        if(filename[x]!='.')
            {
              filename2[x]=filename[x];
              x++;
              goto LOOP;
            }
        ptr=(char *)memcpy(filename1,filename2,x);
        filename1[x]='\0';
         filename2[x]='\0';
        char bhavar[100] = "C:/FaceDNR/OUTPUT/NORM_";
        strcat(bhavar,filename2);
        strcat(bhavar,".txt");
        mid = fopen( bhavar, "w+" );
        strcat(temp,filename);
        filename=temp;
         IplImage* image = cvLoadImage( filename, 1 );
        cvShowImage( "INPUT IMAGE", image );

        if( image)
              {
                 if(image->width < 325 || image->height < 256)  //variable window  for
different size of images
                    {
                          imgSize.width =256; // visualisation image is
                          imgSize.height =256; // 256x256 pixels
                    }
                else if(image->width < 600 || image->height < 500)
                   {
                          imgSize.width =680; // visualisation image is
                          imgSize.height =480; // 680x480 pixels
                   }
                else
                   {
                          imgSize.width =1280; // visualisation image is
                          imgSize.height =960; // 1280x960 pixels
                  }

                 image1=cvCreateImage(imgSize,image->depth,image->nChannels);
                 cvResize(image,image1,CV_INTER_AREA);
                 detect_and_draw( image1 );
                 cvReleaseImage( &image );
                cvReleaseImage( &image1 );
                function();
            }

      else
            {
      /* assume it is a text file containing the list of the image filenames to be processed -
one per line */
                 FILE* f = fopen( filename, "r+" );

                 if( f )
                   {
                     char buf[1000+1];

                     while( fgets( buf, 1000, f ) )
                        {
                          printf("face detection starts\n");
                           int len = (int)strlen(buf);
                         printf("len=%d\n",len);
                           while(len<2)                          //loop to end whole program
when last image_name gets read from file
                              {
                                   fclose(f);
                                   fclose(fp);
                                   goto yyy;
                               }

                          while(len > 0 && isspace(buf[len-1]))
                           len--;
                           buf[len] = '\0';
                           filename = buf;
                           filename2="";
                           filename1="";
                           filename2=(char*)malloc(len-4);
                           filename1=(char*)malloc(len-4);
                            x=0;
                              LOOP1:                   //subroutine to extract letters which are
                                                       //before dot(.) from filename
                            if(filename[x]!='.')
                               {
                                    filename2[x]=filename[x];
                        x++;
                                    goto LOOP1;
                                 }
                            ptr=(char *)memcpy(filename1,filename2,x);
                            image = cvLoadImage( buf, 1 );

                              if( image )
                                 {
                                      if(image->width < 325 || image->height < 256)    //variable
window  for different size of images
```

```
                                        {
                                            imgSize.width =256; // visualisation image is
                                                            imgSize.height =256; // 256x256 pixels
                                         }
                                  else if(image->width < 600 || image->height < 500)
                                        {
                                            imgSize.width =680; // visualisation image is
                                            imgSize.height =480; // 680x480 pixels
                                         }
                                  else
                                        {
                                            imgSize.width =1280; // visualisation image is
                                            imgSize.height =960; // 1280x960 pixels
                                         }

                                   image1=cvCreateImage(imgSize,image>depth,image>nChannels);
                                 cvResize(image,image1,CV_INTER_AREA);
                                   cascade_name =
                             C:/FaceDNR/haarcascade/haarcascade_frontalface_alt2.xml";
                                 cascade = (CvHaarClassifierCascade*)cvLoad( cascade_name, 0, 0, 0
);


                                   detect_and_draw( image1 );
                                   cvReleaseImage( &image );
                                 cvReleaseImage( &image1 );
                                 function();

                          }

                      }                 //close of while loop
        fclose(f);
                            }   // close of if
          }                     // close of else
        cvDestroyWindow(" DETECTED FACES ");
        fclose(mid);
        yyy:
        return;                  // End of function
}
```

//********** This function is used to detect faces in an image and save these detected faces in OUTPUT folder ************************//**

```
void detect_and_draw( IplImage* img )
{
   const CvArr* mask;
   double scale = 1;
   IplImage* gray = cvCreateImage( cvSize(img->width,img->height), 8, 1 );
   IplImage* small_img = cvCreateImage( cvSize( cvRound (img->width/scale),cvRound
(img->height/scale)), 8, 1 );

   cvCvtColor( img, gray, CV_BGR2GRAY );
   cvResize( gray, small_img, CV_INTER_LINEAR );
   cvEqualizeHist( small_img, small_img );
```

```
    cvClearMemStorage( storage );

    if( cascade )
        {
            double t = (double)cvGetTickCount();
            faces = cvHaarDetectObjects( small_img, cascade, storage,1.1, 3, 0,cvSize(20,20) );
            t = (double)cvGetTickCount() - t;
            printf( "detection time = %gms\n", t/((double)cvGetTickFrequency()*1000.) );

            for( i = 0; i < (faces ? faces->total : 0); i++ )
              {
                  CvRect* r = (CvRect*)cvGetSeqElem( faces, i);
                 int line_type;
                  int thickness;
                  int shift;
                 char other_string[100];
                 int integer;
                 char integer_string[100];
                 integer=i;
                 cvRectangle( img,cvPoint(r->x*scale,r->y*scale),cvPoint((r->x+r->width)*scale,(r-
>y+r->height)*scale),CV_RGB(255,0,0),1,4,0 );
                 cvSetImageROI(img,*r);
                 sprintf(integer_string,"%d",integer);
                 strcpy(other_string,"C:/FaceDNR/OUTPUT/");
                 strcat(other_string,filename2);
                 strcat(other_string,integer_string);
                 strcat(other_string,".jpg");
                 cvSaveImage(other_string,img);
                 cvResetImageROI(img);
              }
        }
    char dummy[80] = "C:/FaceDNR/OUTPUT/";
    strcat(dummy,input_name);

    cvShowImage( " DETECTED FACES ", img );
    cvWaitKey( 0 );
    cvSaveImage(dummy,img);
    char menu[100] = "C:/FaceDNR/OUTPUT/DET_";
    strcat(menu,input_name);
    strcat(menu,".txt");
    FILE* detface = fopen( menu, "w+" );
    char write[80] = "C:/FaceDNR/OUTPUT/";
    strcat(write,input_name);
    fprintf(detface,write);




    cvReleaseImage( &gray );
    cvReleaseImage( &small_img );

}
```

//Code for face normalization:

```
void CFaceDNRDlg::OnNorm()
{
  FILE* fp = fopen( "angle.txt", "a+" ); //file is created to save angles of rotated images

        //start of eye detection and face normalization
  cascade_name = "C:/FaceDNR/haarcascade/haarcascade eye.xml";
  cascade = (CvHaarClassifierCascade*)cvLoad( cascade_name, 0, 0, 0 );
  storage = cvCreateMemStorage(0);
  CvSize imgSize;
  static int t;
  IplImage* image1;

/*      if(faces->total <= 0)
        {
                MessageBox("Please Select Face Detect Tab First",NULL,NULL);

        }   */

        for( t = 0; t < (faces ? faces->total : 0); t++ )
        {

            char other_string[100];
            static int integer;
            char integer_string[100];
            integer=t;
            sprintf(integer_string,"%d",integer);
            strcpy(other_string,"C:/FaceDNR/OUTPUT/");
            strcat(other_string,filename2);
            strcat(other_string,integer_string);
            strcat(other_string,".jpg");

                filename = other_string;
                //printf("%s\n",filename);
                IplImage* image22 = cvLoadImage( filename, 1 );
                fprintf(fp,"%s",filename);
                fprintf(fp,"=");
                imgSize.width =256; // visualisation image is
                imgSize.height =256; // 256x256 pixels
                image1=cvCreateImage(imgSize,image22->depth,image22->nChannels);
                cvResize(image22,image1,CV_INTER_AREA);

                        if( image1)
                        {
```

```
                    k=t;                    //it is used to give exact input face no. to
corresponding normalized output face
                    eyedetectanddraw( image1 );
                    fprintf(fp,"%f",angledeg);
                    fprintf(fp,"\n");
                    cvReleaseImage( &image22 );
                    cvReleaseImage( &image1 );
                    }

        }//end of for loop
  fclose(fp);
}
```

//Code for face normalisation:

```
void CFaceDNRDlg::OnNorm()
{
        FILE* fp = fopen( "angle.txt", "a+" );// file is created to save angles of rotated images
        cascade_name = "C:/FaceDNR/haarcascade/haarcascade eye.xml";
    cascade = (CvHaarClassifierCascade*)cvLoad( cascade_name, 0, 0, 0 );
        storage = cvCreateMemStorage(0);
        CvSize imgSize;
        static int t;
        IplImage* image1;
        for( t = 0; t < (faces ? faces->total : 0); t++ )
          {
       char other_string[100];
          static int integer;
          char integer_string[100];
          integer=t;
          sprintf(integer_string,"%d",integer);
          strcpy(other_string,"C:/FaceDNR/OUTPUT/");
          strcat(other_string,filename2);
          strcat(other_string,integer_string);
          strcat(other_string,".jpg");
       filename = other_string;
          IplImage* image22 = cvLoadImage( filename, 1 );
          fprintf(fp,"%s",filename);
          fprintf(fp,"=");
          imgSize.width =256;  // visualisation image is
          imgSize.height =256; // 256x256 pixels
          image1=cvCreateImage(imgSize,image22->depth,image22->nChannels);
          cvResize(image22,image1,CV_INTER_AREA);
       if( image1)
                    {
                     k=t;//it is used to give exact input face no. to corresponding normalized
output face
                     eyedetectanddraw( image1 );
                     fprintf(fp,"%f",angledeg);
                     fprintf(fp,"\n");
                     cvReleaseImage( &image22 );
                     cvReleaseImage( &image1 );
                    }
```

```
            }                  //end of for loop
    fclose(fp);
}
```

//****************** This is the sub-function of subroutine "eyedetectanddraw" **************************//**

```
void function()
{
          FILE* fp = fopen( "angle.txt", "a+" );// file is created to save angles of rotated images

          //start of eye detection and face normalization
          cascade_name = "C:/FaceDNR/haarcascade/haarcascade eye.xml";
     cascade = (CvHaarClassifierCascade*)cvLoad( cascade_name, 0, 0, 0 );
          storage = cvCreateMemStorage(0);
          CvSize imgSize;
          static int t;
          IplImage* image1;
     for( t = 0; t < (faces ? faces->total : 0); t++ )
           {

             char other_string[100];
             static int integer;
             char integer_string[100];
             integer=t;
             sprintf(integer_string,"%d",integer);
             strcpy(other_string,"C:/FaceDNR/OUTPUT/");
             strcat(other_string,filename2);
             strcat(other_string,integer_string);
             strcat(other_string,".jpg");
        filename = other_string;
             printf("%s\n",filename);
             IplImage* image22 = cvLoadImage( filename, 1 );
             fprintf(fp,"%s",filename);
             fprintf(fp,"=");
             imgSize.width =256; // visualisation image is
             imgSize.height =256; // 256x256 pixels
             image1=cvCreateImage(imgSize,image22->depth,image22->nChannels);
             cvResize(image22,image1,CV_INTER_AREA);
        cvNamedWindow( "SEPERATED FACES", CV_WINDOW_AUTOSIZE );
             cvShowImage( "SEPERATED FACES", image1 );
             cvWaitKey( 0 );
        if( image1)
                  {
                    k=t;   //it is used to give exact input face no. to corresponding normalized
output face
                    eyedetectanddraw( image1 );
                    fprintf(fp,"%f",angledeg);
                    fprintf(fp,"\n");
                    cvReleaseImage( &image22 );
                    cvReleaseImage( &image1 );
                  }

              }          //end of for loop
          fclose(fp);

}                 //end of function()
```

//******************** This subroutine is used to rotate tilted faces and save resultant faces in OUTPUT folder *********************//**
//eye detection and face normalization

```
int eyedetectanddraw( IplImage* img11)
{

   static CvScalar colors[] =
     {
       {{0,0,255}},
       {{0,128,255}},
       {{0,255,255}},
       {{0,255,0}},
       {{255,128,0}},
       {{255,255,0}},
       {{255,0,0}},
       {{255,0,255}}
     };
   int xval;
   int yval;
   int width1;
   int height1;
   IplImage *dst8,*dst7,*dst9;
   const CvArr* mask;
   double scale = 1.2;
   IplImage* gray = cvCreateImage( cvSize(img11->width,img11->height), 8, 1 );
IplImage* small_img = cvCreateImage( cvSize( cvRound (img11->width/scale),cvRound (img11-
>height/scale)), 8, 1 );
int i,j=0;
float m;
float angle=0,angle1=0,angle2=0,angle3=0,angle11=0,angle22=0,angle33=0;
float angledeg1=0,angledeg2=0,angledeg3=0;
CvMat *dst1,*dst2;
int rows,cols,type;

dst7 = cvCreateImage( cvSize(img11->width,img11->height), 8, 3 );          //it is used to remove
borders of face image
cvCopy( img11, dst7, NULL );
xval=5;
yval=5;
width1=dst7->width-10;
height1=dst7->height-10;

dst8=cvCreateImage(cvSize(width1,height1),IPL_DEPTH_8U,3);
CvRect rect={xval,yval,width1,height1};
cvSetImageROI(dst7,rect);
cvCopy(dst7,dst8,NULL);
```

```
cvResetImageROI(dst7);
CvSize imgSize;
imgSize.width =256; // visualisation image is
imgSize.height =256; // 256x256 pixels

dst9=cvCreateImage(imgSize,dst8->depth,dst8->nChannels);
cvResize(dst8,dst9,CV_INTER_AREA);
dst1= cvCreateMat(dst9->height,dst9->width,CV_8UC3);
dst2= cvCreateMat(dst9->height,dst9->width,CV_8UC3);
cvCopy( dst9, dst1, NULL );    //resized(256x256)image copied to CvMat dst1
cvCopy( dst9, dst2, NULL );

cvCvtColor( img11, gray, CV_BGR2GRAY );
cvResize( gray, small_img, CV_INTER_LINEAR );
cvEqualizeHist( small_img, small_img );
cvClearMemStorage( storage );

if( cascade )
        {
                double t = (double)cvGetTickCount();
                CvPoint center1,center2,center3,center4,center5;
                CvPoint point1,point2,point3, point4;
                CvSeq* faces = cvHaarDetectObjects( small_img, cascade,     storage,1.1, 3,
0,cvSize(30,30) );
                t = (double)cvGetTickCount() - t;
                printf( "eye detection time = %gms\n",     t/((double)cvGetTickFrequency()*1000.)
);

                for( i = 0; i < (faces ? faces->total : 0); i++ )
                  {
                    CvRect* r = (CvRect*)cvGetSeqElem( faces, i );
                    CvPoint center;
                        int radius;
                        center.x = cvRound((r->x + r->width*0.5)*scale);
                        center.y = cvRound((r->y + r->height*0.5)*scale);
                        radius = cvRound((r->width + r->height)*0.25*scale);
                        cvCircle(img11,center,radius,colors[i%8],2,CV_AA,0);
                        point1.x=(center.x)+5;
                        point2.x=(center.x)-5;
                        point3.y=(center.y)+5;
                        point4.y=(center.y)-5;
                        point1.y=(center.y);
                        point2.y=(center.y);
                        point3.x=(center.x);
                        point4.x=(center.x);

                        if(j<1)
                          {
                                center1=center;
                                j++;
                          }
                        else if(j==1)
                          {
                                center2=center;
                                j++;
```

```
          }
         else if(j==2)
          {
                 center3=center;
                 j++;
          }
         else if(j==3)
         {
                 center4=center;
                 j++;
         }
         else if(j==4)
         {
                 center5=center;
         }

   }             //end of for loop



 if(i<=0 || i==1)
  {
         angledeg=0;
         goto HERE;
  }

 point1.y=(center1.y+center2.y)/2;
 point2.y=(center1.y+center2.y)/2;
 point1.x=0;
 point2.x=256;
 cvLine(img11, point1, point2, colors[i%8], 1, CV_AA, 0);
 point1.y=(center1.y+center2.y)/2;
 point2.y=center2.y;
 point1.x=(center1.x+center2.x)/2;
 point2.x=center2.x;
 m=((float)(point2.y-point1.y)/(point2.x-point1.x));           //slope of line
 angle1=atan(m);
 //angle of inclined head
 angledeg1=(angle1*180)/3.14159265;
 point1.y=(center1.y+center3.y)/2;
 point2.y=(center1.y+center3.y)/2;
 point1.x=0;
 point2.x=256;
 cvLine(img11, point1, point2, colors[i%8], 1, CV_AA, 0);   //draws horizontal line
 point1.y=(center1.y+center3.y)/2;
 point2.y=center1.y;
 point1.x=(center1.x+center3.x)/2;
 point2.x=center1.x;
 m=((float)(point2.y-point1.y)/(point2.x-point1.x));          //slope of line
 angle2=atan(m);
 //angle of inclined head
 angledeg2=(angle2*180)/3.14159265;

 point1.y=(center2.y+center3.y)/2;
 point2.y=(center2.y+center3.y)/2;
```

```
point1.x=0;
point2.x=256;
cvLine(img11, point1, point2, colors[i%8], 1, CV_AA, 0);  //draws horizontal line
point1.y=(center2.y+center3.y)/2;
point2.y=center3.y;
point1.x=(center2.x+center3.x)/2;
point2.x=center3.x;
m=((float)(point2.y-point1.y)/(point2.x-point1.x));
angle3=atan(m);
//angle of inclined head
angledeg3=(angle3*180)/3.14159265;

angle11=abs(angledeg1);
angle22=abs(angledeg2);
angle33=abs(angledeg3);

if((angle11<=angle22) && (angle11<=angle33))
 {
        angle=angle1;
  }

if((angle22<angle11) && (angle22<angle33))
  {
        angle=angle2;
  }
if((angle33<angle11) && (angle33<angle22))
  {
        angle=angle3;
  }

angledeg=(angle*180)/3.14159265;

while(abs(angledeg)>40)
  {
        angledeg=0;
        goto HERE;
  }


double T[]= { 1, 0, 128, 0, 1, 128, 0, 0, 1 };
double R[]= {cos(angle),-sin(angle),0,sin(angle,cos(angle),0, 0, 0, 1 };
double T1[]={ 1, 0, -128, 0, 1, -128, 0, 0, 1 };
double c[9],Tm[9];
int scale,x,y,m,n;
CvMat MT,MR,MT1,Mc;
CvMat *MTm;
cvInitMatHeader( &MT, 3, 3, CV_64FC1, T, CV_AUTOSTEP);
cvInitMatHeader( &MR, 3, 3, CV_64FC1, R, CV_AUTOSTEP);
cvInitMatHeader( &Mc, 3, 3, CV_64FC1, c, CV_AUTOSTEP);
cvInitMatHeader( &MT1, 3, 3, CV_64FC1, T1, CV_AUTOSTEP);
MTm=cvCreateMat(3,3,CV_64FC1);
cvMatMulAdd( &MT, &MR, 0, &Mc);
cvMatMulAdd( &Mc, &MT1, 0,MTm);
double t1,t2,t3;
CvScalar scal;
```

```
                CvMat *MP1;
                MP1=cvCreateMat(3, 1, CV_64FC1);
                for(x=8;x<dst1->rows-5;x++)
                {
                        for(y=8;y<dst1->cols-5;y++)
                         {
                                double P[] = { x, y, 1 };
                                CvMat MP;
                                cvInitMatHeader(&MP,3,1,CV_64FC1,P, CV_AUTOSTEP);
                                scal=cvGet2D(dst1,x,y);
                                cvMatMulAdd( MTm, &MP, 0, MP1);
                                m=0;n=0;
                                t1=cvmGet(MP1,m,n);
                                m=1;n=0;
                                t2=cvmGet(MP1,m,n);
                                m=2;n=0;
                                t3=cvmGet(MP1,m,n);
                                if(t1<256 && t1>=0 && t2<256 && t2>=0)
                                 {
                                        cvSet2D(dst2,t1,t2,scal);
                                 }
                         }
                }
        }



        char other_string1[100];
        int integer1;
        char integer_string1[100];
        integer1=k;
        sprintf(integer_string1,"%d",integer1);                    //rotated faces are saved
        strcpy(other_string1,"C:/FaceDNR/OUTPUT/");
        strcat(other_string1,filename2);
        strcat(other_string1,integer_string1);
        strcat(other_string1,".jpg");
        cvSaveImage(other_string1,dst2);
    strcat(other_string1,"\n");
    fprintf(mid,other_string1);
    cvNamedWindow( " EYE LOCALIZATION ", CV_WINDOW_AUTOSIZE );
    cvShowImage( " EYE LOCALIZATION ", img11 );
    cvWaitKey( 0 );

    cvNamedWindow( "NORMALIZED IMAGE", CV_WINDOW_AUTOSIZE );
    cvShowImage( "NORMALIZED IMAGE", dst2 );
    cvWaitKey( 0 );
    cvDestroyWindow(" EYE LOCALIZATION ");
    cvDestroyWindow( "NORMALIZED IMAGE" );
    cvDestroyWindow("SEPERATED FACES" );

    cvReleaseImage( &gray );
    cvReleaseImage( &small_img );
    cvReleaseMat(&dst1);
    cvReleaseMat(&dst2);
    cvReleaseImage( &dst7 );
    cvReleaseImage( &dst8 );
    cvReleaseImage( &dst9 );

    HERE: return 0;
}
```

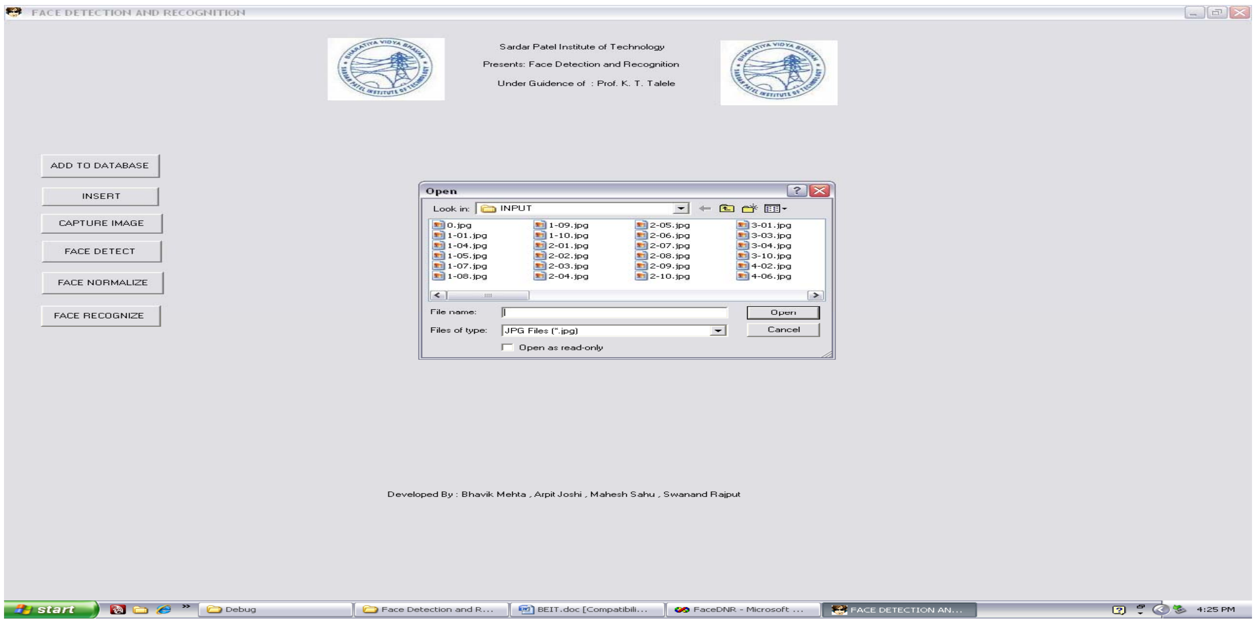
//Code for face recognition:

```
void CFaceDNRDlg::OnRecog()
 {
        IplImage* out;
        IplImage* in;
        IplImage* image1;
        IplImage* image2;
        char ch[30],tmp[100],input[30],para[100];
        float coreff=0.0;
        int i;
        CvSize refSize;
        CvSize tarSize;
        FILE* InputFILE;
    char dummy[100] = "C:/FaceDNR/OUTPUT/NORM_";
        strcat( dummy , filename2);
        strcat(dummy,".txt");
    InputFILE = fopen(dummy,"r+");
    while(!feof(InputFILE))
          {
                coreff=0.0;
                fscanf(InputFILE,"%s",input);
        in=cvLoadImage(input,0);

                IplImage* ref=cvCreateImage( cvGetSize(in), IPL_DEPTH_8U, 1 );
                cvEqualizeHist( in, ref );

                char dumb[100] = "C:/FaceDNR/OUTPUT/";
                strcat(dumb,"db.txt");


                FILE* hFile ;
                hFile = fopen( dumb, "r+" );

                if(ref->width >= 92 || ref->height >= 112)          //variable window  for
                                                                    //different size of images
                 {
                        refSize.width =92; // visualisation image is
                        refSize.height =112; // 256x256 pixels
                  }

                image1 = cvCreateImage(refSize,ref->depth,ref->nChannels);
                cvResize(ref,image1,CV_INTER_AREA);
                 uchar *ref_data = ( uchar* ) image1->imageData;
        //pointer to first image
                int width1= image1->width;
int height1 = image1->height;
int step1= image1->widthStep;
int fft_size1 = width1* height1; //first image size


 float sum=0;
int i;
for(i=0;i<fft_size1;i++)
sum=sum+(ref_data[i]/256.0);


float avg=sum/fft_size1;
float std1[10304];

for(i=0;i<fft_size1;i++)
        std1[i]=(ref_data[i]/256.0);


for(i=0;i<fft_size1;i++)
        std1[i]=std1[i]-avg;

for(i=0;i<fft_size1;i++)
        std1[i]=(std1[i]*std1[i]);

float  sum1=0;

for(i=0;i<fft_size1;i++)
        sum1=sum1+std1[i];

float  sx=sqrt(sum1/fft_size1);//standard deviation of image 1

while(!feof(hFile))
   {

        fscanf(hFile,"%s",ch);
        out=cvLoadImage(ch,0);
        IplImage* target=cvCreateImage(cvGetSize(out), IPL_DEPTH_8U,1);

        cvEqualizeHist( out, target );




         if(target->width >= 92 || target->height >= 112)    //variable window
                                                              //for different size of
                                                              //images
        {
                tarSize.width =92; // visualisation image is
                tarSize.height =112; // 256x256 pixels
          }
         image2=cvCreateImage(tarSize,target->depth,target->nChannels);
         cvResize(target,image2,CV_INTER_AREA);
        uchar *ref_data1 =(uchar*) image2->imageData;
         int width2 = image2->width;
 int height2 = image2->height;
 int step2 = image2->widthStep;
 int fft_size2= width2 * height2;//second image size


 //calculation of standard deviation of second image
 float  sum4=0;

 for(i=0;i<fft_size2;i++)
 sum4=sum4+(ref_data1[i]/256.0);

 float  avg2=sum4/fft_size2;
 float std2[10304];

 for(i=0;i<fft_size2;i++)
         std2[i]=(ref_data1[i]/256.0);

 for(i=0;i<fft_size2;i++)
         std2[i]=std2[i]-avg2;

 for(i=0;i<fft_size2;i++)
         std2[i]=std2[i]*std2[i];

 float sum5=0;
 for(i=0;i<fft_size2;i++)
         sum5=sum5+std2[i];

 float sy=sqrt(sum5/fft_size2);       //standard deviation of image 2


 //calcualtion of correlation coefficient
 float   add=0;
 float rx1[10304];

 for(i=0;i<fft_size2;i++)
rx1[i]=((ref_data[i]/256.0)- avg)*((ref_data1[i]/256.0)-avg2);
float sum6=0;

for(i=0;i<fft_size2;i++)
         sum6=sum6+rx1[i];

float rxy = sum6/(((fft_size1-1)*sx)*sy);




if(rxy>coreff)
   {
         coreff=rxy;
         strcpy(tmp,ch);
         strcpy(para,input);
    }
if(coreff>0.99)
         coreff=1.0;
```

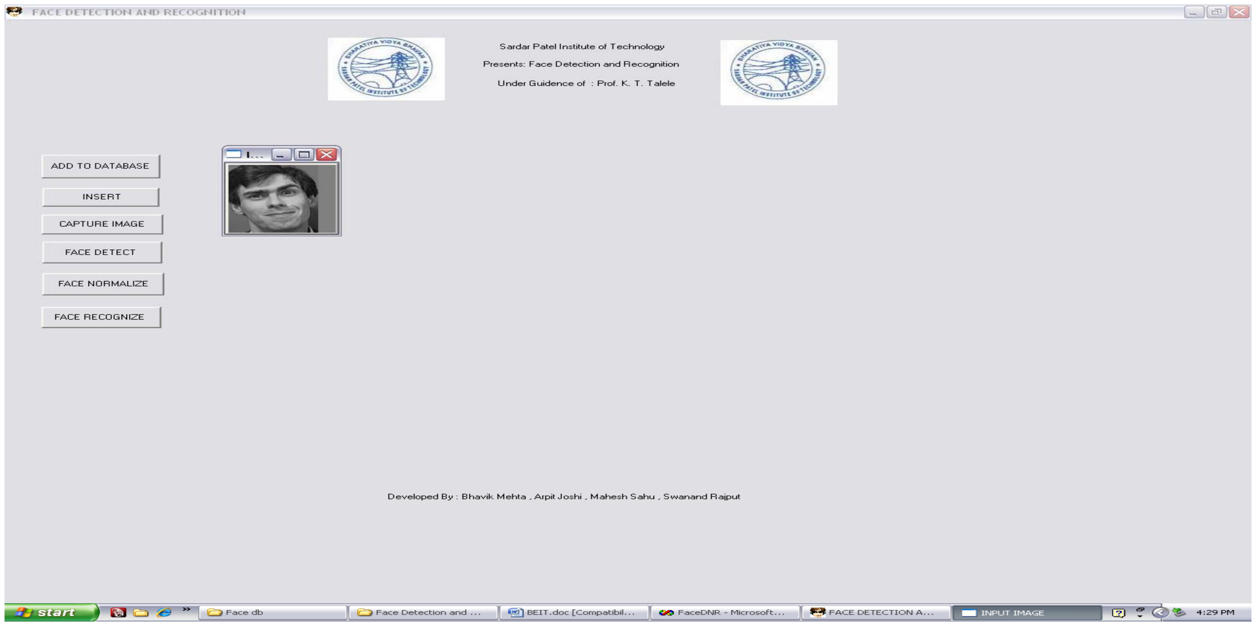
```
                }   // End of Internal While loop

                char rec[] = "Recognized";
                strcat(rec,tmp);
                char nrec[] = "Not Recognized";
                float threshold=0.4;
                if(coreff>=threshold)
                {
                        //MessageBox(input,NULL,NULL);

                        IplImage* recog = cvLoadImage( tmp, 0 );
                        cvNamedWindow( "RECOGNIZED IMAGE", 1 );
                        cvShowImage( "RECOGNIZED IMAGE", recog );
                        cvWaitKey( 0 );
                        cvDestroyWindow("RECOGNIZED IMAGE");
                        cvDestroyWindow("INPUT IMAGE");
                }
                else
                {
                        MessageBox(nrec,NULL,NULL);

                 }


                fclose(hFile);
        }

        fclose(InputFILE);

}
```

3.2. Snapshots of GUI & Report :

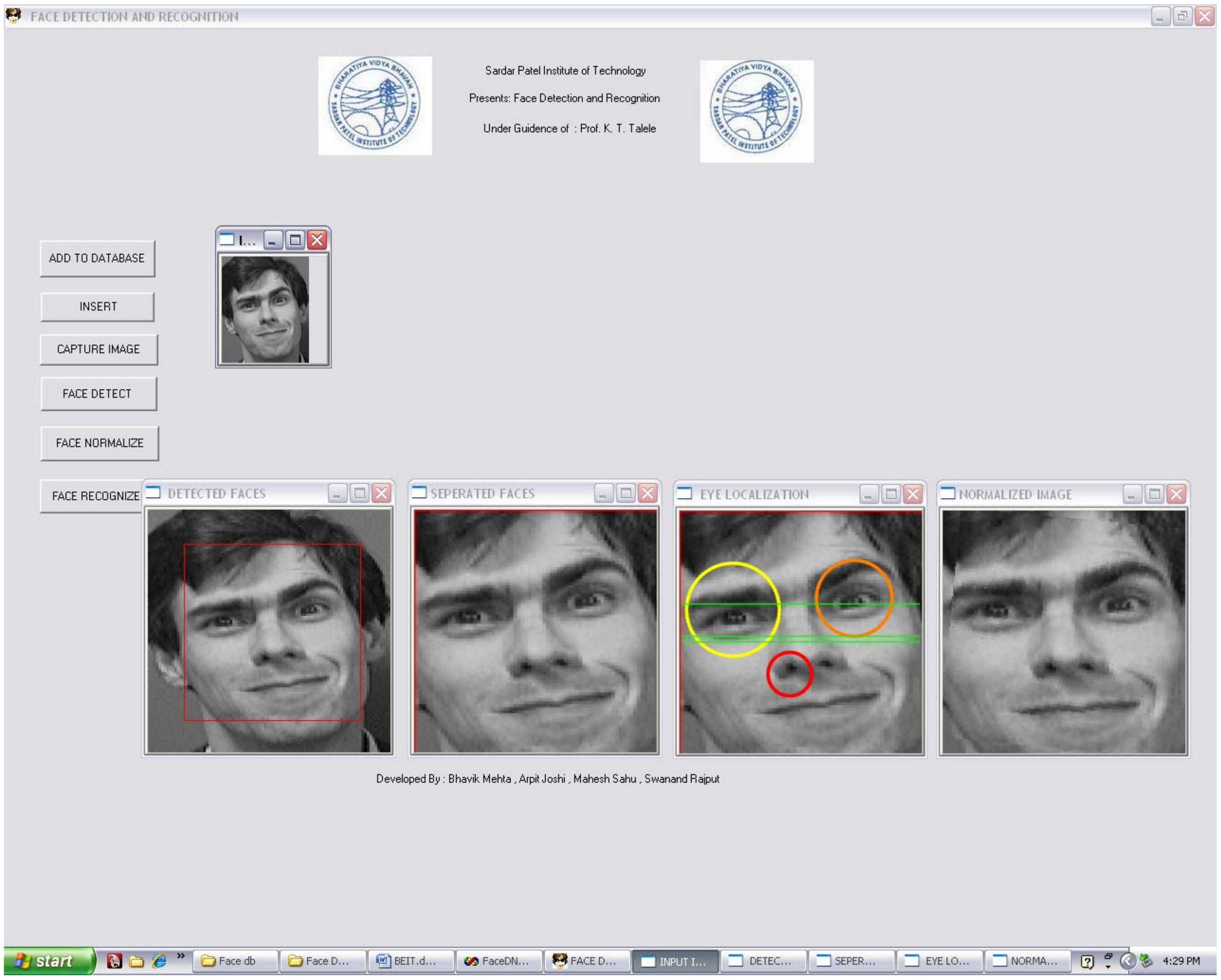
- You can select an image (for face detection or to include it in the database) by clicking on the pushbutton **Insert**
- A browser window appears on the screen to select an image with the extension .jpg and size 90x112



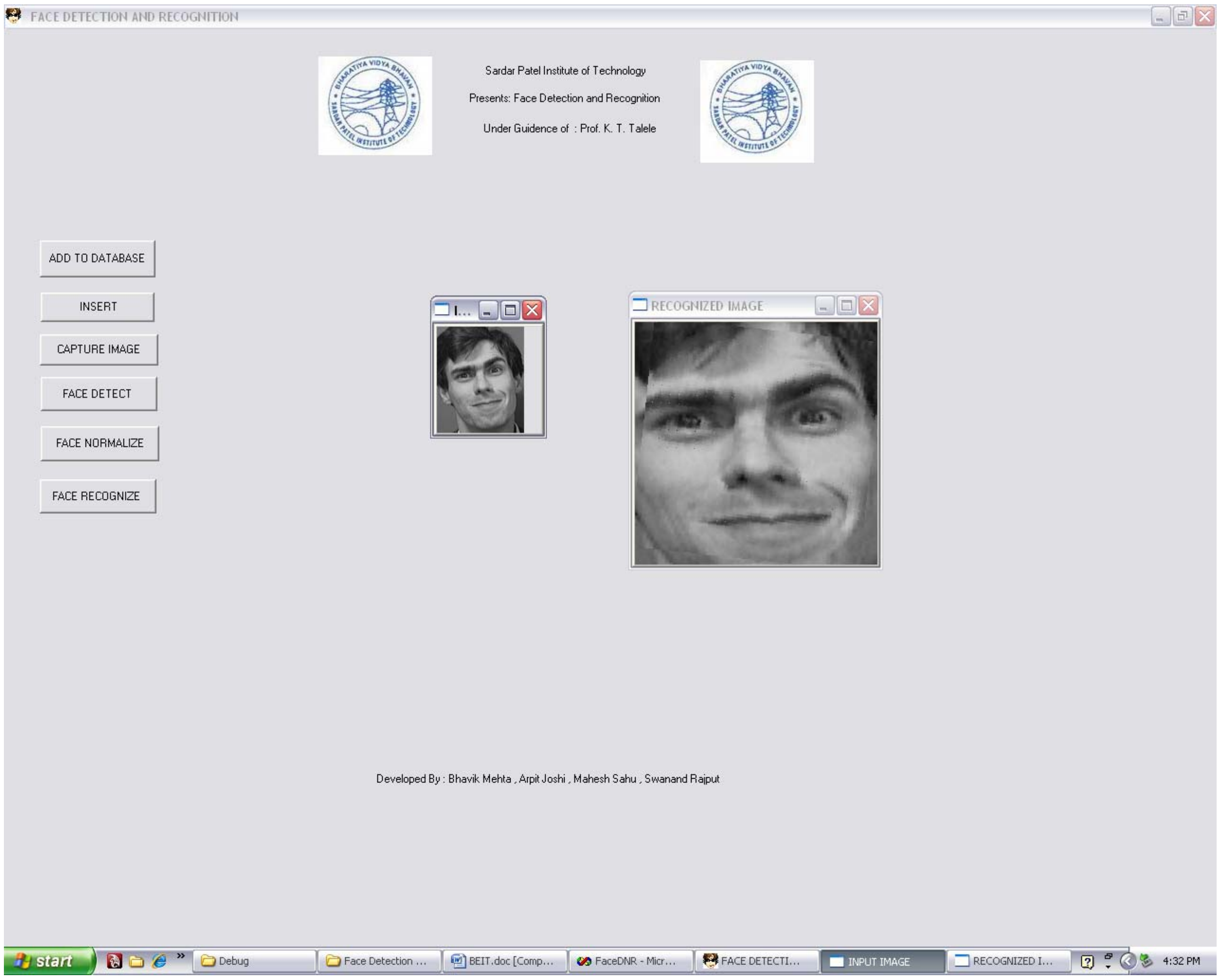
- Select the required image. The image appears on the GUI as shown below


FACE DETECTION AND RECOGNITION
Sardar Patel Institute of Technology
Presents: Face Detection and Recognition
Under Guidence of : Prof. K. T. Talele
ADD TO DATABASE
INSERT
CAPTURE IMAGE
FACE DETECT
FACE NORMALIZE
FACE RECOGNIZE
Developed By : Bhavik Mehta , Arpit Joshi , Mahesh Sahu , Swanand Rajput
start
Face db
Face Detection and ...
BEIT.doc [Compatibil...
FaceDNR - Microsoft...
FACE DETECTION A...
INPUT IMAGE
4:29 PM

- First select an image as described previously
- To do so, click on the **FACE DETECTION** push button
- It displays the detected face and normalizes it


FACE DETECTION AND RECOGNITION
Sardar Patel Institute of Technology
Presents: Face Detection and Recognition
Under Guidence of : Prof. K. T. Talele
ADD TO DATABASE
INSERT
CAPTURE IMAGE
FACE DETECT
FACE NORMALIZE
FACE RECOGNIZE
DETECTED FACES
SEPERATED FACES
EYE LOCALIZATION
NORMALIZED IMAGE
Developed By : Bhavik Mehta , Arpit Joshi , Mahesh Sahu , Swanand Rajput
start
4:29 PM

- First select an image as described previously
- Now you can perform face recognition of this image against the images loaded in the database
- To do so, click on the **FACE RECOGNITION** push button
- Face Recognition software checks the database, matches the isodensity lines map
- If matches are found, the correspoding matched image is displayed
- Else , ‘*Not recognised* ’ message appears.


FACE DETECTION AND RECOGNITION
Sardar Patel Institute of Technology
Presents: Face Detection and Recognition
Under Guidence of : Prof. K. T. Talele
ADD TO DATABASE
INSERT
CAPTURE IMAGE
FACE DETECT
FACE NORMALIZE
FACE RECOGNIZE
RECOGNIZED IMAGE
Developed By : Bhavik Mehta , Arpit Joshi , Mahesh Sahu , Swanand Rajput
start
Debug
Face Detection ...
BEIT.doc [Comp...
FaceDNR - Micr...
FACE DETECTI...
INPUT IMAGE
RECOGNIZED I...
4:32 PM

3.3. Test Cases:

Here we had performed testing in three phases

1. Taking ten Images of each person in database.
2. Taking five Images of each person in database.
3. Taking three images of the same person in the database.

In this testing we have 230 images in the database and we compare each image with all the images in the database. We actually find the correlation coefficient between all the images and classify the result as

a) Not recognized: Corrleation coefficient below threshold value.
b) Recognised: Correlation coefficient above threshold value.
c) False Recognition: Image of a person matched with image of another person in database.

Result of first test case:

Total No. of Images -- 207
Threshold----0.7

False Recognition -- 0
Not Recognition -- 6
Recognized -- 201

Efficiency - 97.1% (201/207)

Result of second test case:

Total No. of Images -- 230
Threshold----0.70

False Recognition -- 3
Not Recognition -- 8
Recognized -- 219

Efficiency - (219/230= 95.21%)

Result of third test case:

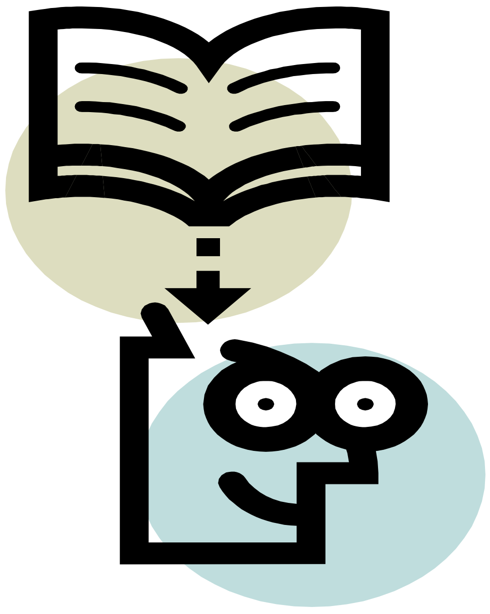
Total No. of Images -- 230

Threshold----0.70

False Recognition	--	5
Not Recognition	--	29
Recognized	--	196

Efficiency - (196/230= 85.21%)

Documentation and CD



4. DOCUMENTATION & CD :

4.1. System Manual :

HOW TO RUN:

1. Copy the install folder
2. Run the FaceDNR.exe file which is located in FaceDNR\debug
3. Browse the image and click on detect button.
4. To capture Image from WebCam click on CAPTURE button.
5. Click on FACE DETECT in order to detect the face in the image you have selected.
6. Click on FACE NORMALIZE in order to normalize the face.
7. Click on FACE RECOGNIZE in order to authorize the person.

INSTALLATION:

1] INSTALLATION FOR VISUAL STUDIO:

To install Help for Standard and Professional Editions

1. Insert the CD or DVD you installed Visual Studio from, or browse to the network share where you installed from.
2. Double-click setup.exe.
3. In the wizard, click **Install Product Documentation**.

—or—

- Insert your MSDN CD and run MSDN setup.

To install Help for Express Editions

1. From the **Start** menu, choose **Control Panel** and then choose **Add or Remove Programs**.
2. Select the Express Edition you installed and then click **Change/Remove**.
3. In the setup wizard, choose **Add Optional Products** and then click **Next**.
4. Select **Microsoft MSDN Express Library 2005** and then click **Next**.
5. Specify whether you have the installation media or intend to download the documentation from the Web and then click **Install**.

2] INSTALLATION FOR OPENCV:

1. Insert the CD or DVD you installed OpenCV from, or browse to the network share where you installed from.
2. Double-click setup.exe.

TOOLS:

1. VISUAL STUDIO 2005
2. OPEN CV 1.0

4.2. User Manual with Instruction :

Our project is easy to use. By following the below steps the code can be setup with full functionality.

PRE-CONDITIONS:

- ✓ For implementation of our project, the system must be installed with Webcam Drivers and Matlab at the system where our code will run.
- ✓ The database, with which code is going to check the input for recognition of person, should be equipped with all images that are required for recognition or else error message is displayed.

Let us look at the project implementation in a step-by-step manner

STEP 1: Insert Images to Input Folder inside FaceDNR.

- ✓ Select INSERT Button on GUI, an dialog box will be displayed for selecting Image at any location inside Hard disk.
- ✓ Choose appropriate path where the file is residing using dialog box.
- ✓ You can select any type of file format like “jpeg, bmp, png , etc.” .
- ✓ See that your selected file has been copied and placed to INPUT Folder.

STEP 2: Capturing Image through Webcam.

- ✓ To capture Image from WebCam, select CAPTURE button on GUI screen.
- ✓ Make sure that WebCam is set Capture mode.
- ✓ Press any key on keyboard to capture another Image, and continue doing this, until you are done.
- ✓ Finally press ESC key on keyboard to finish capturing.
- ✓ Note that the Image is saved into INPUT folder of FaceDNR folder with name “CAM.jpg”.

STEP 3: To Add Image to Database.

- ✓ Select ADD TO DATABASE Button on GUI, an dialog box will be displayed for selecting Image at any location inside Hard disk.
- ✓ Choose appropriate path where the file is residing using dialog box.
- ✓ You can select any type of file format like "jpeg, bmp, png , etc." .
- ✓ See that your selected file has been copied and placed to "db" Folder inside FaceDNR.
- ✓ Also, see "db.txt" file inside OUTPUT folder of FaceDNR folder, and note that path to this has been added inside this text file.

STEP 4: Face Detection for an Image.

- ✓ Select FACE DETECT Button on GUI, for detecting faces inside Image that you supplied to the application either by using INSERT or CAPTURE Button.
- ✓ For Face Detection, application read image from INPUT folder, apply algorithm on it and store the result inside OUTPUT folder with filename same as that of filename of input file.

STEP 5: For Face Normalization.

- ✓ Select FACE NORMALIZE Button on GUI, for Normalizing faces inside Image that are stored inside OUTPUT folder.
- ✓ For Face Normalization, application read image from OUTPUT folder, apply algorithm on it and store the normalized result inside OUTPUT folder with filename same as that of filename of input file along with system generated integers appended to it..
- ✓ Note that a file is created inside OUTPUT folder with file name "NORM_" and appended with original file name and finally ".txt".
- ✓ This file contains path to image files that are stored by Normalization code after Normalization of faces.

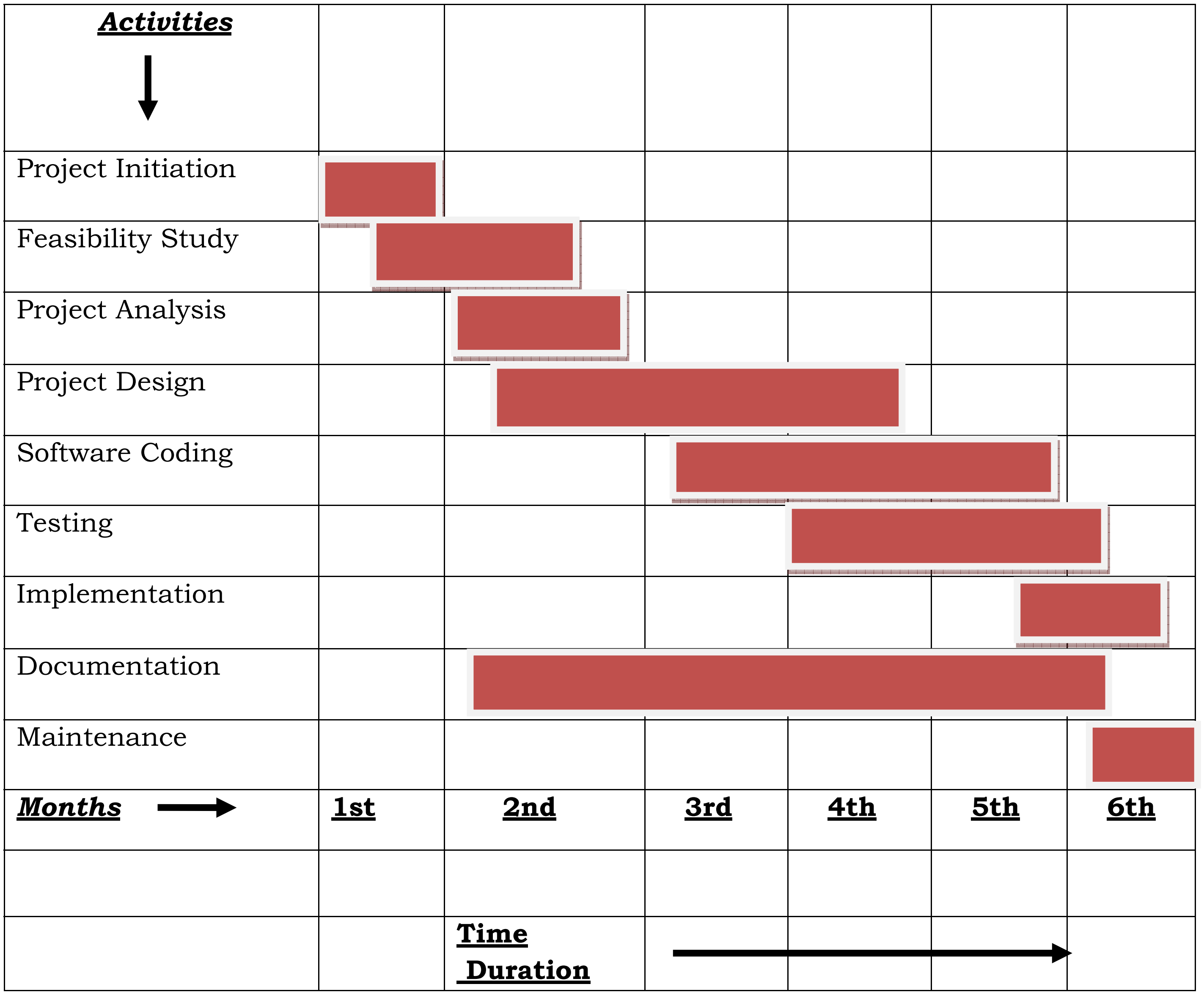
STEP 6: For Face Recognition.

- ✓ Select FACE RECOGNIZE Button on GUI, for Recognizing faces inside Image that are stored inside OUTPUT folder.
- ✓ It uses "NORM_*.txt" file as an Input and compute Corr-eff for each image with all images from databases and finally displaya images with highesh threshold value.

Estimation & Analysis

5. ESTIMATION AND ANALYSIS :

5.1. GANTT CHART



The various activities in the Gantt chart are shown along with the tentative time-period required to complete that activity. The horizontal bar one below the other indicates that the activities are overlapping.

5.2. COST ESTIMATION :

COnstructive COst MOdel (COCOMO) :

Like all estimation models for software, the COCOMO II models require sizing information. Three different sizing options are available as part of the model hierarchy: **Object points**, **function points**, and **lines of source code**.

Complexity Weighting Factor:

Objects/Screens	Simple	Medium	Complex
Input	1	1	-
Output	2	2	1
3G/4G Components	-	1(C)	1(VC++)

Value for Simple = 2
Value for Medium = 4
Value for Complex = 6

Total	**6**	**16**	**12**

Total= 6+16+12 = 34 (No. of Object Points)
NOP = (object points) x [(100 - %reuse)/100]
NOP = 34 x [(100-1/7) / 100]
NOP= 33.95 approx. 34.

Productivity Rates for Object Points:

Development	**V.Low**	**Low**	**Medium**	**High**	**V.High**
Environment	**V.Low**	**Low**	**Medium**	**High**	**V.High**
Productivity Rate	2	4	6	8	10

Productivity Rate for Development: NOP/Medium = 34/6
= **5.66 approx. 6**
Productivity Rate for Environment: NOP/Easy 34/4
= **8.5 approx. 9**

Estimated Efforts: 34/ (6+9) = 34/15 = 2.26
<u>2 person/month</u> i.e. <u>15 days/person.</u>

Thus it is estimated by COCOMO Model to have 2 persons per month to complete the Project.

<u>Cost estimation using COCOMO MODEL</u>

<u>BASIC COCOMO Equation: -</u>

$E= ai\ (KLOC)^{bi} * EAF$

$D= cb * E^{db}$

$N= E/D$

Where;

N→No. of people
D→Duration
E→Effort equation
EAF→Effort adjustment factor

Now;

LOC= 700
KLOC= 0.70
$E= 3.0\ (0.70)^{1.12} *1.2$
$= 3.0* 0.67 * 1.2$
= **<u>2.41 persons / month</u>**

$D = (2.5)\ (2.41)^{0.32}$
$= 2.5* 1.32$
=3.3 months ~= **<u>3 months</u>**
N= E/D
= 0.66 ~= **<u>1 person</u>**
Cost per person month= 3500
Estimated Cost= 2.41 * 3500
= **<u>8435 Rs.</u>**
Approx. <u>8500 Rs.</u>

This is the overall Cost of the Project development as estimated using the Basic COCOMO model.

5.3. RISK MANAGEMENT

Risk analysis and management are a series of steps that help a software team to understand and manage risks. Understanding the risks and taking proactive measures to avoid or manage them—is a key element of good Software project management. There are various types of risks viz; *Project risks, Business risks, Technical risks* etc.

Risk identification is a systematic attempt to specify threats to the project plan (estimates, schedule, resource loading, etc.). By identifying known and predictable risks, we can take a first step towards avoiding them when possible and controlling them when necessary.

One method for identifying risks is to create a *risk item checklist*. The checklist can be used for risk identification and focuses on some subset of known and predictable risks in the following generic subcategories:

1. Product Size
2. Business impact
3. Customer characteristics
4. Process definition
5. Development Environment
6. Technology to be built
7. Staff size & experience.

Risk projection also called *risk estimation,* attempts to rate each risk in two ways—the likelihood or probability that the risk is real and the consequences of the problems associated with the risk.

Risk Table:

Risk	**Category**	**Probability**	**Impact**
Size estimated may be significantly low	PS	40%	3
Larger number of users than planned	PS	30%	2
Delivery deadline will be tightened	BU	20%	3
Customer will change requirements	PS	30%	4
Technology will not meet expectations	TE	50%	2
Funding will be lost	BU	10%	4
End-user resists the System	MR	10%	1
		∑ 190%	**∑ 19**

Impact values:
1—Catastrophic
2 – Critical
3 – Marginal
4 – Negligible

Risk identification: 85% Reusable components.

Risk Probability: **(P)** 190/7 = 27.1%

20 Reusable Components are there.
Due to only 85% of components are Reusable, out of 20, **3** components are not reusable.

Avg. Loc per Component = 600/20
= **30**

Price per LOC = **10 Rs**

Cost **(C)** = 3 × 30 × 10
= **900 Rs.**

The overall *risk exposure,* RE*,* is determined using the following relationship.

RE = *P* x *C*

Risk Exposure = P × C
= 27/100 × 900
= **240 Rs.**

Risk Mitigation, Monitoring and Management

An effective strategy for Risk Analysis is best suited if the three major issues are considered:

- Risk avoidance
- Risk monitoring
- Risk management and contingency planning.

The ***Risk Information Sheet*** is shown below.

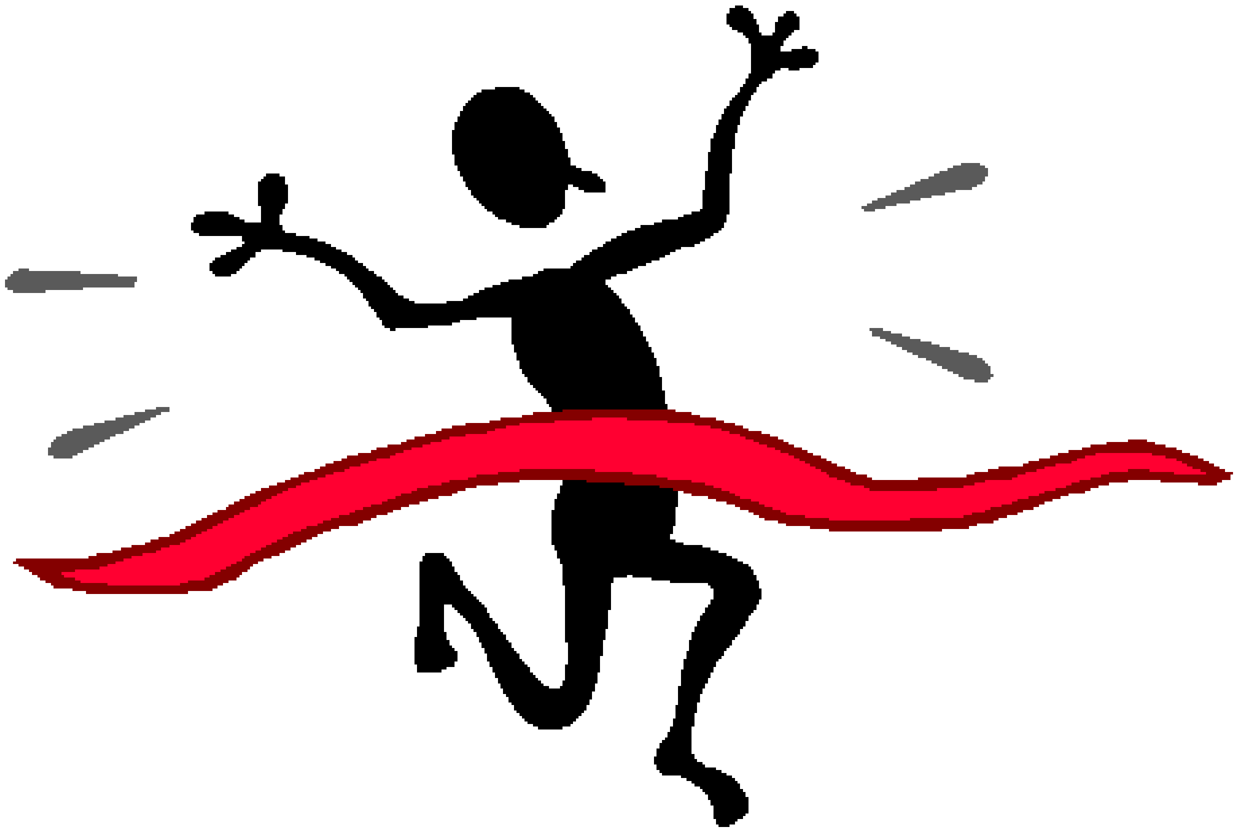
Risk Id: cfr1	Date 20-3-10	Prob.: 40%	Impact: 3
Description: Estimated code size may be low. More code may be required.			
Refinement/Context: A new function needs to be coded to add extra specification given by user.			
Mitigation/Monitoring: Contact the user to get full requirements for the project			
Management/Contingency Plan/Trigger: Check if functionality is already provided by any other feature in the application.			
Current Status: 25-3-10 mitigation steps initiated.			
Originator: Bhavik		Assigned: Arpit	

Risk Id: cfr2	Date 24-3-10	Prob.: 20%	Impact: 3
Description: Delivery deadlines will be tightened			
Refinement/Context: Deadlines are tightened due to pressure from customer or project being delayed			
Mitigation/Monitoring: Tight monitoring of the project schedules and milestones being completed.			
Management/Contingency Plan/Trigger: Important functionality can be delivered before the project is build.			
Current Status: 31-3-10 mitigation steps initiated.			
Originator: Arpit		Assigned: Bhavik	

Risk Id: cfr3	Date 26-3-10	Prob.: 30%	Impact: 4
Description: Customer will change requirements			
Refinement/Context: Customer requirements change because of lack of understanding of project			
Mitigation/Monitoring: Communicate with users to clearly define user requirements			
Management/Contingency Plan/Trigger: Clearly define requirements using a prototype model.			
Current Status: 1-4-10 mitigation steps initiated.			
Originator: Arpit		Assigned: Bhavik	

So, these are some of the Risk Information Sheets produced to cover the risks that were identified in the Project. These risks are assigned by one person to the other according to the skills of the assigned person.

Conclusion



6. CONCLUSIONS :

6.1. APPLICATIONS :

- We have made a user friendly Browser which displays the files stored in the drives of a particular system from which a user can select a file for insertion, then using our application software for face detection in selected file.

- Used in Industries for attendance system for an Employee using the Database.

- Our software can be used at airports or any public places for detecting any suspicious person or terrorist.

- This application can be used in Network (LAN or Internet) where a user can upload an image file to our system which will be running at server end and then it will perform intended function and will return result to client who had inserted image to the application.

- This application can be used in Laptops or PC’s by using Webcam for user authentication as an secondary part if user doesn’t want to enter password.

6.2. FUTURE ENHANCEMENTS :

Like any worldly commodity, every application-software needs to be maintained and enhanced to be with the current times. It needs to be modified to add more and more features in it.

Since, we have made this application-software to only with still images, it can also be integrated to work with frames of images in videos or capture directly through cameras (real-time). Also, this software is platform dependent and could work only under windows operating system, it should be made platform independent using any programming language that is platform independent like Java.

This would prove to be the additional usage of this application-software apart from just running under a single operating system.

Also this application-software can be enhanced and extended to be used in car's for making sure that person in driver's seat won't take a nap while driving. This can be done by extacting eye from image and to see if eye's are closed or not, also by calculating angle of tilted face of driver face. If both conditions are occurred then appropriate action need to be taken.

References :

Papers:

[1] Rafael C. Gonzalez, Digital Image Processing.

[2] V.V. Starovoitov, D. I. Samal and D. V. Briliuk, Three Approaches for Face Recognition.

[3] John D. Woodward Jr., Christopher Horn, Julius Gatune and Aryn Thomas, Biometrics a Look at Facial Recognition.

[4] S. T. Gandhe and K. T. Talele, Face Recognition Using Soft Computing Tools.

[5] S. T. Gandhe, K. T. Talele and A. G. Keskar, Intelligent Face Recognition Techniques: A Comparative Study.

[6] J. Canny, 'A Computational Approach to Edge Detection,' IEEE Transactions on pattern analysis and machine intelligence, vol.8, no. 6, pp.679-698, 1986.

[7] T.K. Leung, M.C. Burl, and P. Perona, 'Finding Faces in Cluttered Scenes Using Random Labeled Graph Matching,' Proc. Fifth IEEE International Conference Computer Vision, pp. 637-644, 1995.

Books :

1. O"Reilly : learning opencv.
2. Maths five – khumbhojkar.
3. Let Us C : By yashwant kanetkar.

Website visited:

1. www.msdn.mircosoft.com
2. www.cognotics.com/opencv
3. www.software.intel.com/en-us/forums/
4. www.facedetectgroup.com
5. www.opencv.com