HUMAN FACIAL DETECTION SYSTEM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology in Computer Science and Engineering

By

PRATEEK PAWAN
108CS019
&
KUSHAGRA PANDEY
108CS012

Department of Computer Science and Engineering
National Institute of Technology Rourkela-769008
2012

HUMAN FACIAL DETECTION SYSTEM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Technology in Computer Science and Engineering

By

PRATEEK PAWAN
108CS019
&
KUSHAGRA PANDEY
108CS012

Under the guidance of

Prof. Rameswar Baliarsingh

Department of Computer Science and Engineering
National Institute of Technology Rourkela-769008
2012

NATIONAL INSTITUTE OF TECHNOLOGY
ROURKELA

CERTIFICATE

This is to certify that the thesis entitled, "**HUMAN FACIAL DETECTION SYSTEM**" submitted by **Mr. PRATEEK PAWAN, Roll No. 108CS019 AND KUSHAGRA PANDEY, Roll No. 108CS012** in partial fulfillment of the requirement for the award of Bachelor of Technology Degree in Computer Science and Engineering at the National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any University/Institute for the award of any Degree or Diploma.

Date: May 14, 2012

Prof. Rameswar Baliarsingh
Department Of Computer Science and Engineering
National Institute of Technology
Rourkela – 769008

TABLE OF CONTENTS

ABSTRACT

Augmenting human computer interaction with automated analysis and synthesis of facial expressions is the goal towards which much research effort has been devoted to in the last few years. Face recognition and detection is one of the important aspects of natural human machine interfaces; this technology has great applications such as in security systems, capturing image, authentication and in clinical practice. Although humans recognize facial parts virtually without effort or delay, reliable face detection and recognition by a computer system is still a challenging task. The facial part recognition and detection problem is challenging because different individuals have different structure for their nose, eye and ears differently. In this project we are trying to design a face detection and recognition system in real time using the concepts of Haar wavelets, Eigen faces and template matching. In this project we will be using our system for detecting face area, eyes, mouth ears and nose.

ACKNOWLEDGEMENT

We would like to articulate our profound gratitude and indebtedness to those persons who helped us in the project.

First of all, we would like to express our obligation to our project guide Prof. Rameswar Baliarsingh, (Associate Professor, Department of Computer Science and Engineering, National Institute of Technology, Rourkela), for his motivation, help and supportiveness. We are sincerely thankful to him for his guidance and helping effort in improving our knowledge on the subject. He has been always helpful to us in all aspects and we thank him from the deepest of our heart.

An assemblage of this nature could never have been attempted without reference to and inspiration from the works of others whose details are mentioned in reference section. I acknowledge my indebtedness to all of them.

At the last, my sincere thanks to all my friends who have patiently extended all sorts of helps for accomplishing this assignment.

Date: May 14, 2012

PRATEEK PAWAN(108CS019)

KUSHAGRA PANDEY(108CS012)

INTRODUCTION

Face detection and recognition is a very substantial and natural way of human-machine interaction. But developing this project which is face detection system is very challenging which work as effectively as humans.

Face detection and recognition is based on facial features recognition and face detection. So in order to start face detection and recognition we have to first completely and accurately detect the face features as like nose, mouth, eyes and the distance between the each other.

We are designing our project, face detection and recognition in real time system. In order to do the same we have studied various techniques of face detection and recognition. So, here first we will going through all the methods to construct such system suitable for finding or detecting face and then will use anyone of them to proceed for the construction of our system.

We will be using various face detection techniques which are:-

- **Detecting the Face**
- **Detecting Facial Features**

VARIOUS METHODS USED FOR FACE DETECTION

The various methods used for face detection may be classified in the following categories:-

- **Knowledge based method**- It is rule-based method which encode our knowledge of human faces.

- **Feature-invariant method**- It is a method of algorithms that find invariant features of face despite its angle or position.

.

1. Knowledge based methods-

These are the rule-based methods which encode our knowledge of human faces. In it the system tries to capture our knowledge of faces, and translates them into a set of rules defined by the computer. However it is easy to guess some simple rules. For instance, a face usually has two symmetric eyes, and the eye area is a bit darker than the cheeks. Facial features can be the distance between the two eyes or the color intensity difference between the two eye area and the lower zone. The major problem which we face in this method is the difficulty in building an appropriate set of rule for the system to work on. If we use the set of rules which are too general then there can be many false positive results. On the other hand, it can also happen that there will be many false positive results if the rules are too detailed. This is the reason due to which we will not be using this method for the development of our project. A solution to overcome this problem is to build hierarchical knowledge-based methods. However, this approach alone is very limited.

2. Feature-invariant methods-

In feature-invariant method we try to find some invariant features for face detection. The idea behind this method is to overcome the limitations of our instinctive knowledge of faces. In this method we first try to find eye-analogue pixels, so using this we can remove unwanted pixels from the images. After performing the segmentation process, they consider each eye-analogue segment as a candidate of one of the eyes. Then, a set of rule is executed to determine the potential pair of eyes. Then once the eyes are selected, the algorithm tries to calculate the face area and captures them inside a rectangle. Then the four vertexes of the face are determined using the set of functions fed in the system. So, the faces are normalized to a fixed size and orientation. Then, the face regions are verified using a methodology based on neural network back propagation.

FEATURE DETECTION USING HAAR LIKE FEATURES-

Haar features are used in object recognition as they feature digital image. They owe their name to the similarity with Haar wavelets. Viola and Jones joined their hand and found out the idea of using Haar wavelets and developed the method named Haar-like features. In this method we considers adjacent rectangular regions at a specific location in a detection window, and then sums up the pixel moderations in these regions and calculates the difference between these variations. And the we use this difference to categorize subsections of the image to be detected. For instance, let us take an image database with human faces. It is a commonly found that among all faces the region of the eyes is a bit darker than the region of the cheeks. So a common Haar feature for face detection is a set of two neighboring rectangles which lie above the eye and the cheek area. In it we define the position of all these rectangles relative to a detection window which will act like a bounding box to the target object. In the detection method of the Viola and Jones object detection, a proper window of the target size is moved over the input original image, and then for each and every part of the image the Haar-like feature is calculated respectively. This difference is then noted and is compared to a learned threshold that separates non-objects from objects. As we know such a Haar-like feature is a weak classifier so in order to avoid errors a large number of Haar-like features are required to describe an object with accuracy. In the Viola and Jones object detection framework, the Haar-like features are therefore prepared in something known as classifier cascade to form a strong apprentice or classifier. Therefore we will be using cascade classifiers as it will improve performance and correctness of our work.

A recognition process is much more efficient, if it is based on the detection of features that encode some information about the class to be discovered. This is the case of Haar like features that, encode the existence of leaning contrasts between regions in the image. A set of these features can be used to encode the contrasts exhibited by human faces and their spacial interactions. Haar like features are so known because they are calculated similar to the coefficients in Haar wavelet transforms.

The object detector of the OpenCV has been initially suggested by Paul Viola and amended by Rainer Lienhart. First, a classifier namely a cascade of boosted classifiers working with haar-like features is trained with a few hundreds of sample views of a particular object i.e., a face or a car, called positive examples, that are scaled to the same size (say, 20x20), and the negative examples are arbitrary images of the same size as that of image.

After a classifier is trained, it can be applied to a region of interest (of the same size as used during the training) in an input image. The classifier outputs a "1" if the region is likely to show the object (i.e., face/car), and "0" in the other case. To search for the object in the whole image, we will have to move the search window across the image and check every possible location using the classifier. The classifier is basically designed in such a manner so that it can be easily "resized" in order to be able to locate the objects of interest of different sizes, which is much more efficient than resizing the image. So, to find an object of an undetermined size in the image the scan procedure should be done many times at different scales.

The word "cascade" in the classifier name means that the resultant classifier consists of several simpler classifiers, that are applied subsequently to a region of interest, until at some stage the candidate is rejected or all the stages are to be passed. The word "boosted" means that the classifiers at every stage of the cascade are complex themselves and they are built out of basic classifiers using, one of four different boosting techniques i.e weighted voting. Currently Discrete Adaboost, Real Adaboost, Gentle Adaboost and Logitboost are supported in it. The basic classifiers are decision-tree classifiers, with at least 2 leaves. Haar-like features are the input to the basic classifiers. The feature used in a particular classifier is specified by its shape , position within the region of interest and the scale.

Rectangular-Haar-like features

We can define rectangular Haar-like feature as the variance of the sum of pixels of areas inside the rectangle, that can be at any position and scale inside the image we are working on. And we call this modified feature set as *2-rectangle feature*. Viola and Jones also did a lot of work on 3-rectangle features and 4-rectangle features. These values indicate certain individualities of a particular area of the image or the picture. All these feature types can indicate the existence or absence of certain characteristics in the image, such as edges or changes in texture. For instance, a 2-rectangle feature can easily tell us where the border lies between a dark region and a light region.

Fast-Computation of Haar-like features

The idea used by Viola and Jones for this technique was the use summed area tables, which they named as integral images. These Integral images as two-dimensional lookup tables which are in the form of a matrix and are of the same size as the original image are. Every element of this integral image have the sum of all pixels situated on the up-left region of the original image. This allows us in computing the sum of rectangular areas in the original image, at any position or scale, by using only four lookups:

$$\text{sum} = pt_4 - pt_3 - pt_2 + pt_1.$$

where points pt_n belong to the integral image.

Each Haar-like feature needs further 4 lookups for this purpose, which completely depends on how it is defined. Viola and Jones's 2-rectangle features need 6 lookups, 3-rectangle features need 8 lookups, and 4-rectangle features 9 nine lookups.

Tilted Haar-like features

Lienhart and Maydt introduced the concept of the tilted 45° Haar-like feature. This feature is used to increase the dimensionality of the set of features in an attempt to improve the recognition of the objects in the original images. This feature is quite useful, as these features are able to describe the object in a better way in the original image. For instance, a 2-rectangle tilted Haar-like feature can indicate the existence of an edge at 45°.

Messom and Barczak revised this idea to a generic rotated Haar like feature. Although this idea sounds mathematically good, but some of the practical problems prevented the use of Haar like features at any angle. In order to be fast in detecting the face, detection algorithms use low resolution images, causing rounding errors. For this purpose only, rotated Haar like features are not commonly used these days and are also not helpful for us.

FEATURE TYPES AND EVALUATION:-

The features which Viola and Jones used are based on Haar wavelets. Haar wavelets are basically single wavelength square waves (one having high interval and another low interval).In two dimension ,a square wave is a pair of adjacent rectangle of which one is light and other is dark.

However, since the features used by Viola and Jones rely on more than one rectangular area, which are generally more complicated. There are basically four types of feature types used

- Two 2-rectangle feature (one horizontal and other vertical).
- One 3-rectangle feature.
- One 4-rectangle feature

These features are shown in figure 1.

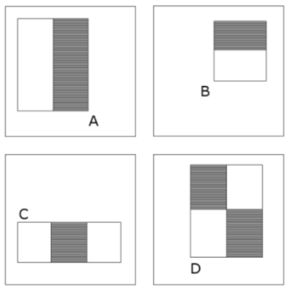


Figure 1.Examples of Haar features

The features used for the detection framework involve the sum of pixel values within rectangular areas. The value of any feature is always simply the sum of pixels within clear rectangles subtracted from the sum of pixel within shaded rectangles. Nowadays rectangle features of this kind are rather primitive to use as many alternatives such as steerable filters has been developed. The presence of a Haar feature is generally determined by removing the average dark region pixel value from that of the average light region pixel value. If the difference is more than a threshold value, then that feature is said to be present. So by using this typical kind of image depiction called as integral image with the rectangular features it is seen that the feature evaluation is a considerably much faster as compared. To determine the presence or absence of hundreds of Haar features at every image position and at several scales efficiently, Viola and Jones used Integral Image. Generally "integrating" means summing up all the small units together. Here the small units are the pixel values and the integral value of each pixel is the sum of all the pixels above it and to its left.

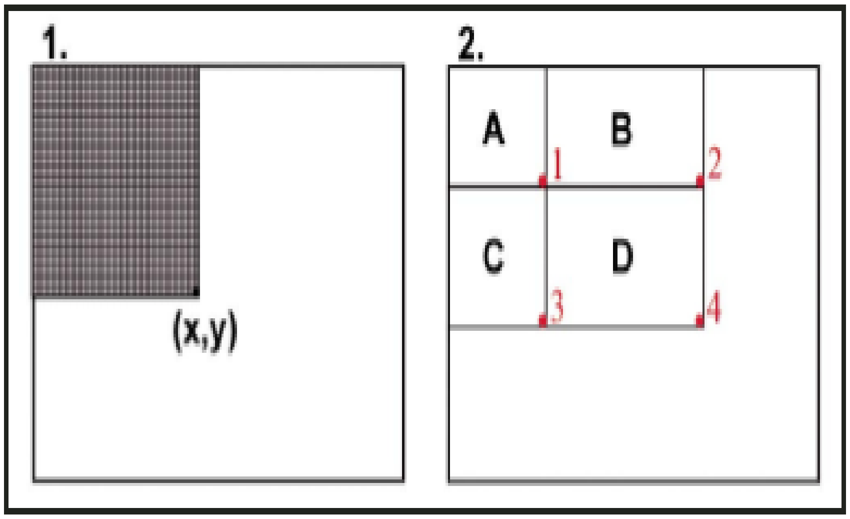


Figure 2.Above shows that the pixel value at (x,y),contains the sum of all pixel values within rectangular region that has one corner on the top left of the original image and the other at location (x,y).

And similarly the pixel values of rectangular part D is (x4, y4)-(x2, y2)-(x3, y3) +(x1, y1)

<u>METHODOLOGY</u>

For creating a face detection recognition system we need to develop an application which works in real-time with good accuracy as well as good speed. We are approaching this project using

OpenCV (Open Computer Vision) library with a compiler named CodeBlocks. Our entire work is in C++ language.

Implementing Haar concept-

Creating the database of a particular feature-

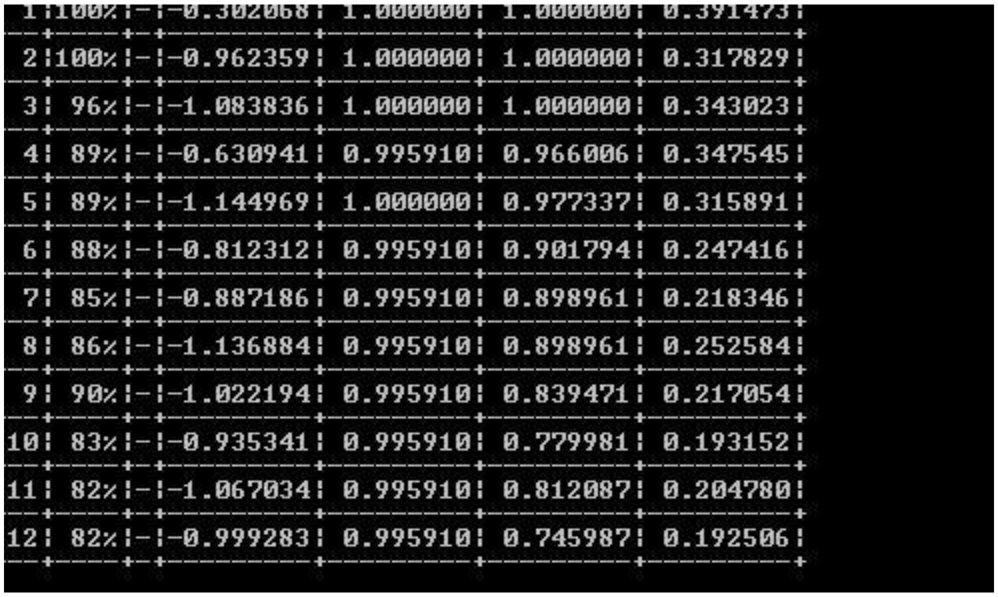
The database is generated using software where at first we have to crop the photos accordingly and automatically it stores cropped one in a folder predefined. Then after cropping the images we write a small code in the RUN command along with the directory involved. The code looks somewhat like as shown below:-

```
C:\Users\Prateek\createsamples.exe -info positive/info.txt -vec data/vector.vec -num 527 -w 24 -h 24
```

This command starts the process of creating samples. And after the samples are created one is requires to again run another code as shown below:--

```
C:\Users\Prateek\haartraining.exe -data data/cascade -vec data/vector.vec -bg negative/infofile.txt -npos 527 -nneg 1142 -nstages 30 -mem 1000 -mode ALL -w 24 -h 24 –nonsym
```

When this command is executed a window is created that resembles as the one shown below:-



```
 1¦100%¦-¦-0.302068¦ 1.000000¦ 1.000000¦ 0.391473¦
---+-----+-+---------+---------+---------+---------+
 2¦100%¦-¦-0.962359¦ 1.000000¦ 1.000000¦ 0.317829¦
---+-----+-+---------+---------+---------+---------+
 3¦ 96%¦-¦-1.083836¦ 1.000000¦ 1.000000¦ 0.343023¦
---+-----+-+---------+---------+---------+---------+
 4¦ 89%¦-¦-0.630941¦ 0.995910¦ 0.966006¦ 0.347545¦
---+-----+-+---------+---------+---------+---------+
 5¦ 89%¦-¦-1.144969¦ 1.000000¦ 0.977337¦ 0.315891¦
---+-----+-+---------+---------+---------+---------+
 6¦ 88%¦-¦-0.812312¦ 0.995910¦ 0.901794¦ 0.247416¦
---+-----+-+---------+---------+---------+---------+
 7¦ 85%¦-¦-0.887186¦ 0.995910¦ 0.898961¦ 0.218346¦
---+-----+-+---------+---------+---------+---------+
 8¦ 86%¦-¦-1.136884¦ 0.995910¦ 0.898961¦ 0.252584¦
---+-----+-+---------+---------+---------+---------+
 9¦ 90%¦-¦-1.022194¦ 0.995910¦ 0.839471¦ 0.217054¦
---+-----+-+---------+---------+---------+---------+
10¦ 83%¦-¦-0.935341¦ 0.995910¦ 0.779981¦ 0.193152¦
---+-----+-+---------+---------+---------+---------+
11¦ 82%¦-¦-1.067034¦ 0.995910¦ 0.812087¦ 0.204780¦
---+-----+-+---------+---------+---------+---------+
12¦ 82%¦-¦-0.999283¦ 0.995910¦ 0.745987¦ 0.192506¦
---+-----+-+---------+---------+---------+---------+
```

Fig.4- Haar Training

This starts Haar training of images. The database generated is first of a .vec file. On accomplishment of training process, replace the "data" in "*Haar\cascade2xml*" by the "data" folder in "Haar\temp", and copy the "vector.vec" to address "*Haar\cascade2xml*".

Now execute the "convert.bat" file. It ultimately gives the “output.xml” file.

Now the .xml file generated contains the information about the images. So now we use this very file to compare with a real time video and wherever it finds the similar image it creates a rectangle around it. Below there are some theory and that is followed by a code snippet that uses this database created and displays the required features inside the video.

Face feature detection using Haar concept-

The following are the functions we used in writing the program for implementing Haar concepts for facial feature detection. Some are user defined and some are inbuilt. The inbuilt function's arguments have been explained below-

- *CvHaarClassifierCascade*:- The word cascade means that the resulting classifier comprises of, several simpler classifiers which are applied subsequently to a region of interest until at some stage the portion is rejected and it will pass. This is also a structure that stores the flags, the number of stages, current object size.

- *CvMemStorage*:-It is basically a structure used to store the dynamically increasing data structures for example sequences, contours, graphs, subdivisions etc.

- *haarcascade_righteye_2splits.xml*:- previously trained file that is being accessed by the program. It defines the classification principles used in the recognition process.

- Capture is a pointer of type *CvCapture* that initiates the camera to be made on.

- *cvNamedWindow*:- creates window for showing the outputs as like images or videos.

- *cvQueryFrame*(*capture*)– this command invokes and initiates the frame acquisition.

- *cvSmooth*:-This functions smooths an image using cvblur or cv_gaussian and other methods.

- *cascade1* = (*CvHaarClassifierCascade**)*cvLoad*(*filename*, 0, 0, 0):-

  filename:-name of the directory containing the details of a trained cascade classifier.

- *cvHaarDetectObjects*(img,cascade2,storage2,1.1,3,0,CvSize(40, 40))

  img– Image to detect objects in

  cascade2 – Haar classifier cascade in internal representation

  storage2– Memory storage to store the resultant arrangement of original object candidate rectangles

  1.1 - The factor by which the search window is scaled between the subsequent scans, 1.1 means increasing window by 10%

  3 - Minimum number (minus 1) of neighbor rectangles that marks an object. We reject all the groups of a smaller number of rectangles than min_neighbors-1. If min_neighbors is 0, then the function does not do any grouping at all and returns all the detected candidate rectangles, which is useful when the user wants to put on a customized grouping procedure to the image.

- *cvRectangle* :- this functions draws a rectangle in the first argument which is the destined image and the further arguments explain about its location where rectangle is to be drawn.

CODE USED

The C++ code for Haar based detection is-

```
#include <stdio.h>

#include "cv.h"

#include "highgui.h"

#include <windows.h>


CvHaarClassifierCascade *cascade;

CvMemStorage                          *storage;

CvFont font;

void detectFaces(IplImage * img );

CvVideoWriter *writer = 0;

int isColor = 1;

int fps     = 5;  // or 30

int frameW  = 640; // 744 for firewire cameras

int frameH  = 480; // 480 for firewire cameras

int main (int argc, char** argv)

{  //writer=cvCreateVideoWriter("out.avi",CV_FOURCC('D','I','V','X'),

   //                    fps,cvSize(frameW,frameH),isColor);


   CvCapture *capture;

         IplImage  *frame;
```

```
        int       key;

        char      *filename = "haarcascade_frontalface_alt.xml";

        cascade = ( CvHaarClassifierCascade* )cvLoad( filename, 0, 0, 0 );

        storage = cvCreateMemStorage( 0 );

        capture = cvCaptureFromCAM( 0 );

        cvNamedWindow( "video", CV_WINDOW_AUTOSIZE );

        while( key != 'q' ) {

                frame = cvQueryFrame( capture );

                if( !frame ) {

                        fprintf( stderr, "Cannot query frame!\n" );

                        break;

                }

                //cvFlip( frame, frame, -1 );

                //frame->origin = 0;


                detectFaces( frame );

      // cvPutText(frame, "eye", cvPoint(56,78 ), &font, cvScalar(23, 12, 255, 0));

cvShowImage( "video", frame );

                key = cvWaitKey(10);

  int nFrames = 1;

/*for(int i=0;i<nFrames;i++){

  cvWriteFrame(writer,frame);      // add the frame to the file

}*/
```

```
        }
        //PlaySound(TEXT("recycle.wav"), NULL, SND_FILENAME);
        cvReleaseCapture( &capture );
        cvDestroyWindow( "video" );
        cvReleaseHaarClassifierCascade( &cascade );
        cvReleaseMemStorage( &storage );
        return 0;
}

void detectFaces( IplImage *img )
{
        int i;
        CvSeq *faces = cvHaarDetectObjects(
                        img,
                        cascade,
                        storage,
                        1.1,
                        3,
                        0 ,
                        cvSize( 40, 40 ) );
        for( i = 0 ; i < ( faces->total ) ; i++ ) {  //( faces ? faces->total : 0 )
                CvRect *r = ( CvRect* )cvGetSeqElem( faces, i );
```

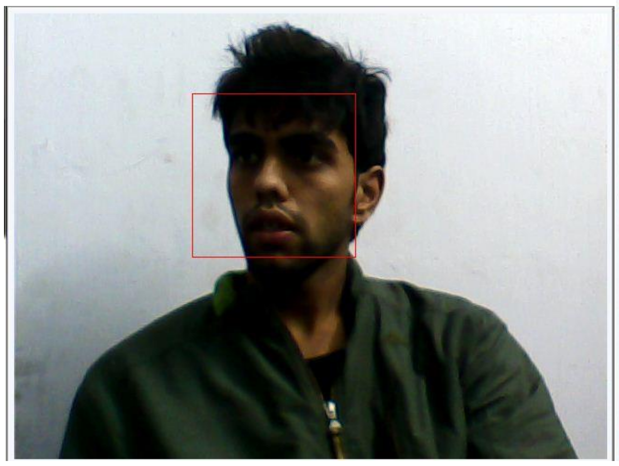
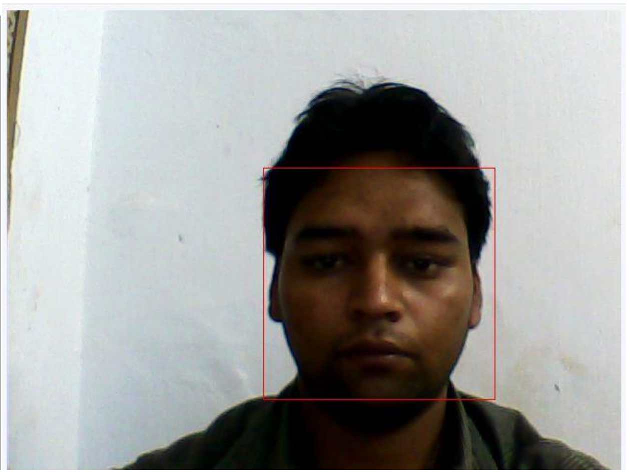
```
          cvRectangle( img,
                              cvPoint( r->x, r->y ),
                              cvPoint( r->x + r->width, r->y + r->height ),
                              CV_RGB( 255, 0, 0 ), 1, 8, 0 );


}
//IplImage* img ;//= 0;
 }
```

RESULTS

We wrote the C++ code for different face detecting technique and tried for extracting feature first using template matching method. (Complete code is in Appendix)

Now, we implement the Haar concept to locate different features like eyes, mouth and face region in a video. (Complete code is in Appendix) In this we tried to use the database file of eye and face to detect the region of face from a video. The output is as follow-

CONCLUSION

From our project work done till now, we conclude that-

- There are various techniques for facial detection and feature detection. We have learned about some of them.

- On the basis of experiments done with template matching method and Haar approach we conclude that, Haar based approach is more robust and can be used.

REFERENCES

- Ming-Hsuan Yang, David J. Kriegman, and Narendra Ahuja, “Detecting Faces in Images: A Survey”, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 24, NO. 1, JANUARY 2002.

- Takeshi Mita, Toshimitsu Kaneko and Osamu Hori, “JOINT HAAR-LIKE FEATURES FOR FACE DETECTION”, Multimedia Laboratory, Corporate Research & Development Center, Toshiba Corporation, Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV’05).

- Matthew A. Turk and Alex P. Pentland, “FACE RECOGNITION USING EIGENFACES”. Vision and Modelling Group, The Media Laboratory, MIT, ©1991 IEEE.

- Paul Viola and Michael Jones, “Robust Real-time Object Detection”,. Second International Workshop On Statistical And Computational Theories Of Vision – Modeling, Learning, Computing, And Sampling, 2001.

- R. J. Baron, “Mechanisms of human facial recognition,” Int. J. Man Machine Studies, vol. 15, pp. 137-178, 1981.
- Canny, J., A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.

SITES

- http://www710.univ-lyon1.fr
- http://opencv.willowgarage.com
- http://opencv.willowgarage.com
- http://www.isa.umh.es
- http://opencv.jp/opencv-2svn_org
- http://programing-tutorial.blogspot.com
- http://wenku.baidu.com
- http://www.comp.leeds.ac.uk
- http://www.seas.upenn.edu
- http://en.wikipedia.org/wiki/Haar-like_features
- http://students.iitk.ac.in/projects
- http://opencv.willowgarage.com
- http://www.cs.iit.edu