PROJECT REPORT

ON

“MATLAB Based Human Face Detection System”

Submitted to

Rashtrasant Tukadoji Maharaj Nagpur University,

Nagpur

In partial fulfillment of the requirement of

M.Sc. Semester - IV (Computer Science) Examination

Submitted by

Vaibhav R. R. Chichmalkar

Under the guidance of

Dr. Mahendra P. Dhore

Associate Professor

DEPARTMENT OF ELECTRONICS & COMPUTER SCIENCE,

Rashtrasant Tukadoji Maharaj Nagpur University Campus, Nagpur.

2018-2019

CERTIFICATE

This is to certify that th**e project report entitled "MATLAB Based Human Face Detection System"** is carried out and developed by **Vaibhav R. R. Chichmalkar** in partial fulfillment of the **M.Sc. (Computer Science)** and submitted to **Rashtrasant Tukdoji Maharaj Nagpur University, Nagpur** under my guidance and supervision.

To the best of my knowledge the matter presented in this project has not been presented earlier for similar degree/diploma.

Project Guide
Dr. M. P. Dhore

Place: Nagpur
Date:

Head
Dr. S. J. Sharma

Internal Examiner

External Examiner

DECLARATION

I, the undersigned, hereby declare that the project work entitled **“MATLAB Based Human Face Detection System”** submitted to **Rashtrasant Tukadoji Maharaj Nagpur University, Nagpur** is my independent work . This is my original work and has not been submitted anywhere for any degree/diploma. The system presented herein has not been duplicated from any other source.

Place: Nagpur **Yours Sincerely,**

Date:

Vaibhav R. R. Chichmalkar

ACKNOWLEDGEMENT

I wish to express my sincere thanks to all who helped me to develop this project work.

My foremost thanks to **Dr. M. P. Dhore** who has guided me in completing this project report. I take the opportunity to express my deep sense of gratitude and whole hearted thanks for his inspiration and guidance throughout the course of this project.

I am also grateful to, **Dr. S. J. Sharma**, Head of Department of Electronics & Computer science who has aided us in completing this project report.

I would also like to thank **Dr. M. T. Wanjari** and **Dr. J. K. Keche** for their precious time and constant help and guidance during the project work.

I am thankful to all members of teaching and non- teaching staff of the Department of Electronics & Computer Science for their co-operation during the course. I would also like to thank my friends and specially those who helped me in my endeavors.

Last but not the least I extend my gratitude to my family for their support and encouragement throughout the work.

Place: Nagpur **Vaibhav R. R. Chichmalkar**

Date:

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Face Detection

From last few years, the face detection has received important consideration. In few areas the face detection gains extraordinary position due to various reasons such as verification of identities, large range of commercial and law enforcement available for feasible technologies. Face detection is one of many applications in digital image processing. It is concerned with the automatic identification of an individual in a digital image. Face detection is used to identifying and locating the human face irrespective of its size, position and situations. Face detection is an easy task for human brain but it is a very difficult task for computer system. To detect the face easily and accurately computer system needs some training factors so that the computer system can easily identify whether it is face or non-face. For detecting the face some thresholds values are sets based on these values a system can detect the human face. If the image specifies the desired threshold value then the image is a face otherwise it is a non-face[3].

1.2 Types of Face Detection Algorithm

Face detection algorithms are divided into two parts: (i) feature based (ii) learning based. The feature-based algorithms are based on the statement that the face is detected based on some simple features, independent of light, face variation and posture[2]. The learning-based algorithm usage an amount of training models, benefit from statistical models and machine learning algorithms.

Various complexities are linked with face detection algorithm that can be:

i. **Quality of image:** Face detection system requires good quality images. A good quality images are collected under predictable circumstances. An image quality is necessary to extract the features from an image. When the calculations of features are not good then the robustness of the system will be lost.

ii. **Variations in illumination:** Due to the lighting changes same faces images will be shown differently. The presence of an object can be changed due to variations in illumination.

iii. **Facial expression:** The person’s facial expressions are affected by the presence of faces.

iv. **Visual angle:** For different angles the face images directly vary about the camera optical axis. There are many algorithms through which the face detection process is carried out but, in this project, viola-jones algorithm is used for detecting the face from images which is one of the most popular algorithms among all the face detection algorithms[4]. Face detection components detects or separates out human faces from the non-face objects present in an image. The image may be captured either in a controlled environment or in an uncontrolled environment.

We now give a definition of face detection: Given an arbitrary image, the goal of face detection is to determine whether or not there are any faces in image and if present, return the image location and location of each face.

1.3 **Factors affecting Detection**

The challenges associated with face detection can be attributed to the following factors:

i. **Pose**: The images of a face may vary due to the relative camera-face pose (frontal, 45-degree, profile, upside down) and some facial features such as an eye or the nose may become partially or wholly occluded.
ii. **Presence or absence of structural components**: Facial features such as beards, moustaches and glasses may or may not be present and there is a great deal of variability among these components including shape, colour and size.
iii. **Facial expression**: The appearance of faces is directly affected by a person’s facial expression.
iv. **Occlusion**: Faces may be partially occluded by other objects[6]. In an image with a group of people, some faces may partially occlude other faces.
v. **Image orientation**: Face images directly vary for different rotations about the camera’s optical axis.
vi. **Imaging conditions**: When the image is formed, factors such as lighting (spectra, source distribution and intensity) and camera characteristics (sensor response, lenses) affect the appearance of the face.

1.4 Steps for Face Detection

The face detection system can be divided into the following steps: -

i. **Pre-Processing**: To reduce the variability in the faces, the images are processed before they are fed into the network. All positive examples that is the face images are obtained by cropping images with frontal faces to include only the front view. All the cropped images are then corrected for lighting through standard algorithms.
ii. **Classification**: Neural networks are implemented to classify the images as faces or nonfaces by training on these examples[7]. We use both our implementation of the neural network and the MATLAB neural network toolbox for this task. Different network configurations are experimented with to optimize the results.
iii. **Localization**: The trained neural network is then used to search for faces in an image and if present localize them in a bounding box. Various Feature of Face on which the work has done on: - Position Scale Orientation Illumination[8].

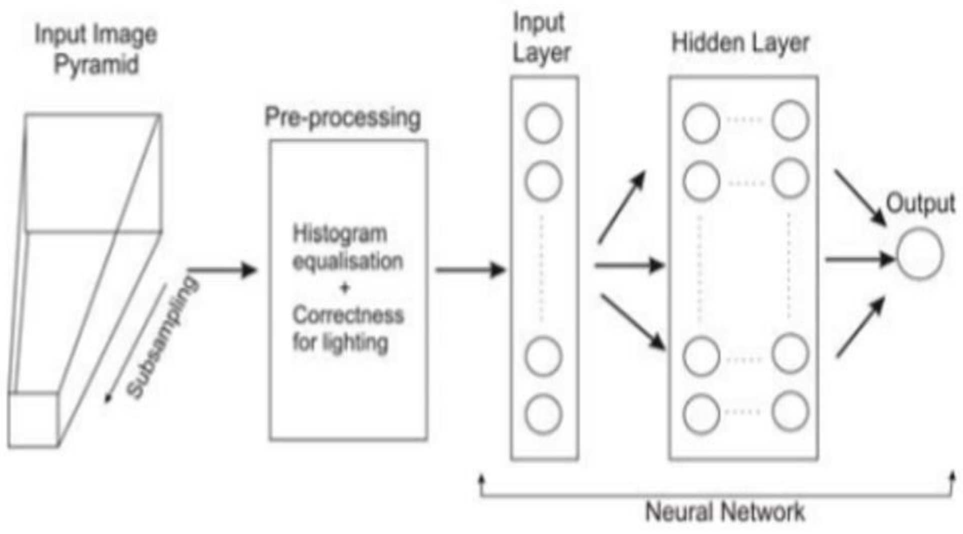


Fig. 1.1 Face Detection

CHAPTER 2

LITERATURE SURVEY

Face detection is a computer technology that determines the location and size of human face in arbitrary (digital) image. The facial features are detected and any other objects like trees, buildings and bodies etc are ignored from the digital image. It can be regarded as a specific case of object-class detection, where the task is finding the location and sizes of all objects in an image that belong to a given class. Face detection, can be regarded as a more general case of face localization[10]. In face localization, the task is to find the locations and sizes of a known number of faces (usually one). Basically, there are two types of approaches to detect facial part in the given image i.e. feature base and image base approach. Feature base approach tries to extract features of the image and match it against the knowledge of the face features. While image base approach tries to get best match between training and testing images[11].

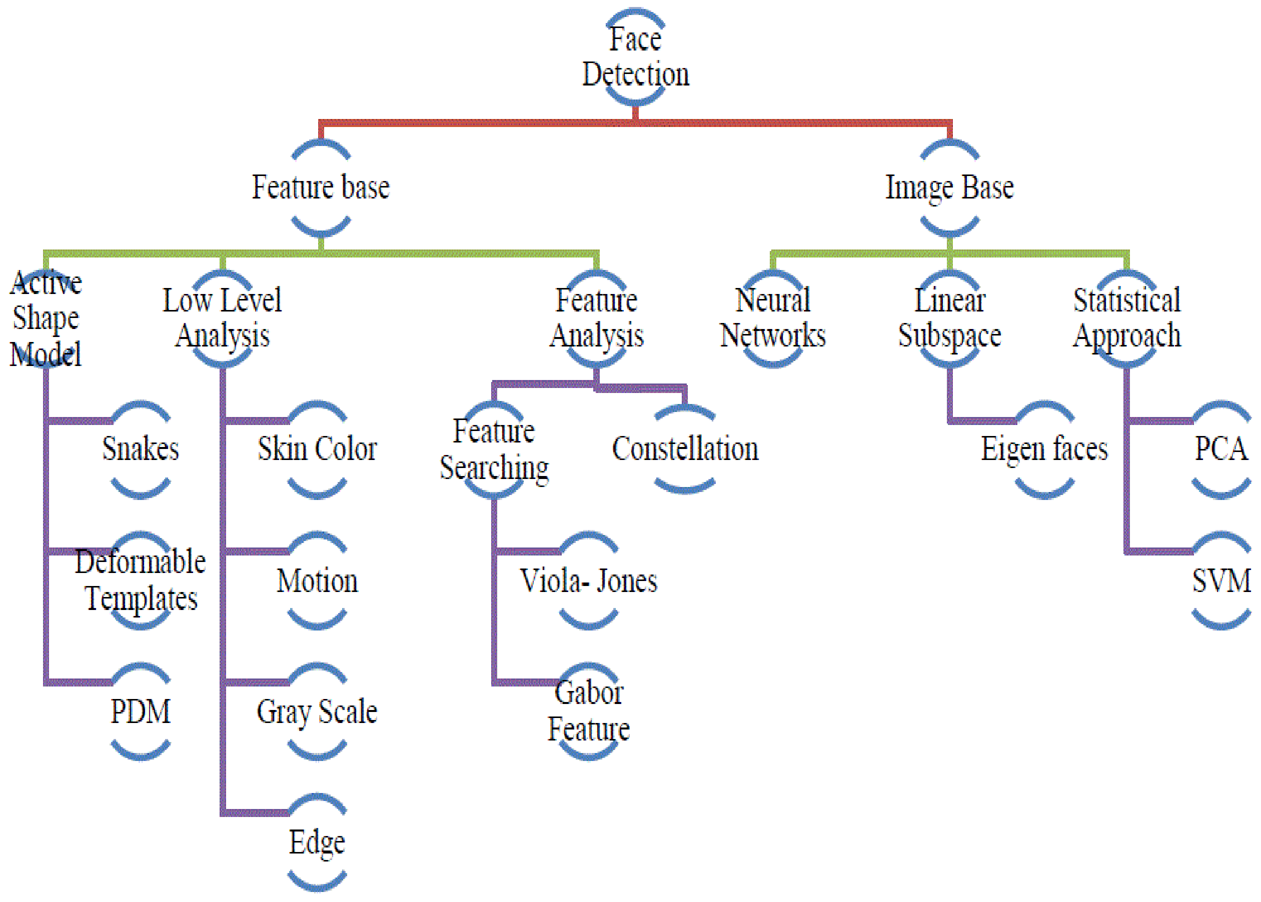


Fig. 2.1 Detection Methods

2.1 FEATURE BASED APPROCH:

Active Shape Model Active shape models focus on complex non-rigid features like actual physical and higher-level appearance of features Means that Active Shape Models (ASMs) are aimed at automatically locating landmark points that define the shape of any statistically modelled object in an image. When of facial features such as the eyes, lips, nose, mouth and eyebrows. The training stage of an ASM involves the building of a statistical

i. Facial model from a training set containing images with manually annotated landmarks. ASMs is classified into three groups i.e. snakes, PDM, Deformable templates

ii. Snakes: The first type uses a generic active contour called snakes, first introduced by Kass et al. in 1987 Snakes are used to identify head boundaries [8,9,10,11,12]. In order to achieve the task, a snake is first initialized at the proximity around a head boundary. It then locks onto nearby edges and subsequently assume the shape of the head. The evolution of a snake is achieved by minimizing an energy function, Esnake (analogy with physical systems), denoted as Esnake = Einternal + EExternal, where Einternal and EExternal are internal and external energy functions. Internal energy is the part that depends on the intrinsic properties of the snake and defines its natural evolution. The typical natural evolution in snakes is shrinking or expanding[14]. The external energy counteracts the internal energy and enables the contours to deviate from the natural evolution and eventually assume the shape of nearby features—the head boundary at a state of equilibria. Two main consideration for forming snakes i.e. selection of energy terms and energy minimization. Elastic energy is used commonly as internal energy. Internal energy varies with the distance between control points on the snake, through which we get contour an elastic-band characteristic that causes it to shrink or expand. On other side external energy relay on image features. Energy minimization process is done by optimization techniques such as the steepest gradient descent. Which needs highest computations. Huang and Chen and Lam and Yan both employ fast iteration methods by greedy algorithms. Snakes have some demerits like contour often becomes trapped onto false image features and another one is that snakes are not suitable in extracting non convex features.

2.1.1 Deformable Templates:

Deformable templates were then introduced by Yuille et al. to take into account the a priori of facial features and to better the performance of snakes. Locating a facial feature boundary is not an easy task because the local evidence of facial edges is difficult to organize into a sensible global entity using generic contours.

The low brightness contrast around some of these features also makes the edge detection process. Yuille et al. took the concept of snakes a step further by incorporating global information of the eye to improve the reliability of the extraction process.

Deformable templates approaches are developed to solve this problem. Deformation is based on local valley, edge, peak, and brightness. Other than face boundary, salient feature (eyes, nose, mouth and eyebrows) extraction is a great challenge of face recognition. E = Ev + Ee + Ep + Ei + Einternal; where Ev, Ee, Ep, Ei, Einternal are external energy due to valley, edges, peak and image brightness and internal energy.

2.1.2 PDM (Point Distribution Model):

Independently of computerized image analysis, and before ASMs were developed, researchers developed statistical models of shape. The idea is that once you represent shapes as vectors, you can apply standard statistical methods to them just like any other multivariate object. These models learn allowable constellations of shape points from training examples and use principal components to build what is called a Point Distribution Model. These have been used in diverse ways, for example for categorizing Iron Age broaches[16]. Ideal Point Distribution Models can only deform in ways that are characteristic of the object. Cootes and his colleagues were seeking models which do exactly that so if a beard, say, covers the chin, the shape model can \override the image" to approximate the position of the chin under the beard. It was therefore natural (but perhaps only in retrospect) to adopt Point Distribution Models. This synthesis of ideas from image processing and statistical shape modelling led to the Active Shape Model. The first parametric statistical shape model for image analysis based on principal components of inter-landmark distances was presented by Cootes and Taylor in. On this approach, Cootes, Taylor, and their colleagues, then released a series of papers that cumulated in what we call the classical Active Shape Model[20].

2.2 LOW LEVEL ANALYSIS:

Based on low level visual features like colour, intensity, edges, motion etc. Skin Colour, Base Colour is a vital feature of human faces. Using skin-color as a feature for tracking a face has several advantages. colour processing is much faster than processing other facial features. Under certain lighting conditions, Colour is orientation invariant. This property makes motion estimation much easier because only a translation model is needed for motion estimation[20]. Tracking human faces using colour as a feature has several problems like the colour representation of a face obtained by a camera is influenced by many factors (ambient light, object movement, etc.

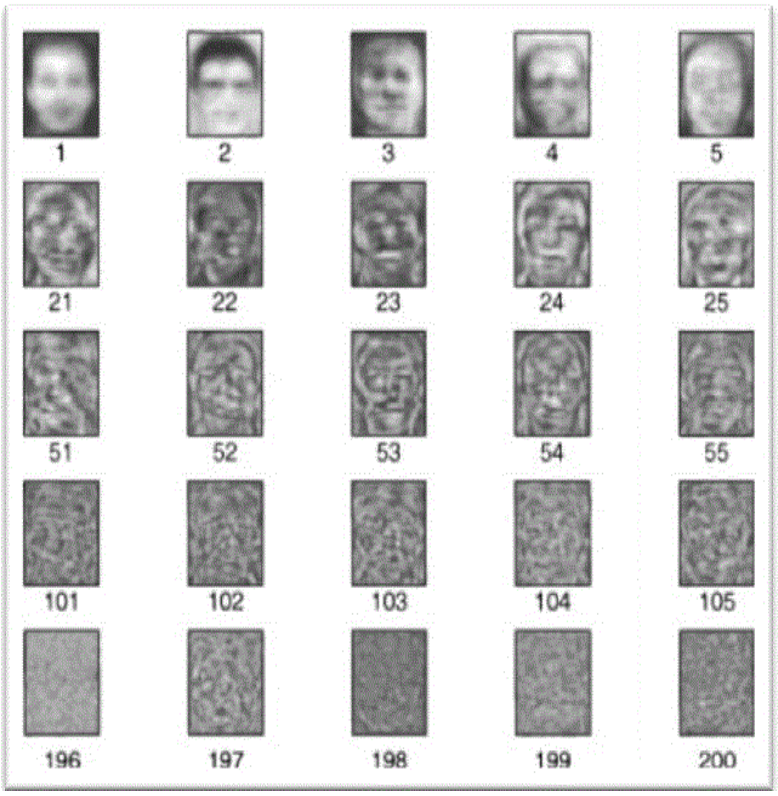


Fig 2.2. Face detection

Majorly three different face detection algorithms are available based on RGB, YCbCr, and HIS colour space models. In the implementation of the algorithms there are three main steps viz.

(1) Classify the skin region in the colour space,

(2) Apply threshold to mask the skin region and

(3) Draw bounding box to extract the face image.

Crowley and Coutaz suggested simplest skin colour algorithms for detecting skin pixels. The perceived human colour varies as a function of the relative direction to the illumination.

The pixels for skin region can be detected using a normalized colour histogram, and can be normalized for changes in intensity on dividing by luminance. Converted an [R, G, B] vector is converted into an [r, g] vector of normalized colour which provides a fast means of skin detection. This algorithm fails when there is some more skin region like legs, arms, etc. Cahi and Ngan suggested skin colour classification algorithm with YCbCr colour space. Research found that pixels belonging to skin region having similar Cb and Cr values. So that the thresholds be chosen as [Cr1, Cr2] and [Cb1, Cb2], a pixel is classified to have skin tone if the values [Cr, Cb] fall within the thresholds. The skin colour distribution gives the face portion in the colour image. This algorithm is also having the constraint that the image should be having only face as the skin region. Kjeldson and Kender defined a colour predicate in HSV colour space to separate skin regions from background. Skin colour classification in HSI colour space is the same as YCbCr colour space but here the responsible values are hue (H) and saturation (S). Similar to above the threshold be chosen as [H1, S1] and [H2, S2], and a pixel is classified to have skin tone if the values [H,S] fall within the threshold and this distribution gives the localized face image. Similar to above two algorithm this algorithm is also having the same constraint[23].

2.3 **MOTION BASED**:

When use of video sequence is available, motion information can be used to locate moving objects. Moving silhouettes like face and body parts can be extracted by simply thresholding accumulated frame difference. Besides face regions, facial features can be located by frame differences.

2.3.1 **Gray Scale Based**:

Gray information within a face can also be treat as important features. Facial features such as eyebrows, pupils, and lips appear generally darker than their surrounding facial regions. Various recent feature extraction algorithms search for local gray minima within segmented facial regions. In these algorithms, the input images are first enhanced by contrast-stretching and gray-scale morphological routines to improve the quality of local dark patches and thereby make detection easier. The extraction of dark patches is achieved by low-level gray-scale thresholding. Based method and consist three levels. Yang and Huang presented new approach i.e. faces gray scale behaviour in pyramid (mosaic) images. This system utilizes hierarchical Face location consist three levels. Higher two level based on mosaic images at different resolution. In the lower level, edge detection method is proposed. Moreover, this algorithm gives fine response in complex background where size of the face is unknown.

2.3.2 **Edge Based**:

Face detection based on edges was introduced by Sakai et al. This work was based on analysing line drawings of the faces from photographs, aiming to locate facial features. Than later Craw et al. proposed a hierarchical framework based on Sakai et al. worked to trace a human head outline. Then after remarkable works were carried out by many researchers in this specific area. Method suggested by Anila and Devarajan was very simple and fast. They proposed frame work which consist three steps i.e. initially the images are enhanced by applying median filter for noise removal and histogram equalization for contrast adjustment. In the second step the edge image is constructed from the enhanced image by applying sobel operator. Then a novel edge tracking algorithm is applied to extract the sub windows from the enhanced image based on edges[23]. Further they used Back propagation Neural Network (BPN) algorithm to classify the sub-window as either face or non-face.

2.4 FEATURE ANALYSIS

These algorithms aim to find structural features that exist even when the pose, viewpoint, or lighting conditions vary, and then use these to locate faces. These methods are designed mainly for face localization

2.4.1 Feature Searching

Viola Jones Method:

Paul Viola and Michael Jones presented an approach for object detection which minimizes computation time while achieving high detection accuracy. Paul Viola and Michael Jones proposed a fast and robust method for face detection which is 15 times quicker than any technique at the time of release with 95% accuracy at around 17 fps. The technique relies on the use of simple Haar-like features that are evaluated quickly through the use of a new image representation. Based on the concept of an ―Integral Image‖ it generates a large set of features and uses the boosting algorithm AdaBoost to reduce the overcomplete set and the introduction of a degenerative tree of the boosted classifiers provides for robust and fast interferences. The detector is applied in a scanning fashion and used on gray-scale images, the scanned window that is applied can also be scaled, as well as the features evaluated[1].

Gabor Feature Method:

Sharif et al proposed an Elastic Bunch Graph Map (EBGM) algorithm that successfully implements face detection using Gabor filters. The proposed system applies 40 different Gabor filters on an image. As a result of which 40 images with different angles and orientation are received. Next, maximum intensity points in each filtered image are calculated and mark them as fiducial points. The system reduces these points in accordance to distance between them[25]. The next step is calculating the distances between the reduced points

using distance formula. At last, the distances are compared with database. If match occurs, it means that the faces in the image are detected. Equation of Gabor filter is shown below:

$$\psi_{u,v}(z) = \frac{\| k_{u,v} \|^2}{\sigma^2} e^{\left(\frac{\|k_{u,v}\|^2 \|z\|^2}{2\sigma^2}\right)} \left[e^{i\vec{k}_{u,v}z} - e^{-\frac{\sigma^2}{2}} \right]$$

Where

$\phi_u = \frac{u\pi}{8}, \quad \phi_u \in [0, \pi)$ gives the frequency,

2.5 CONSTELLATION METHOD

All methods discussed so far are able to track faces but still some issue like locating faces of various poses in complex background is truly difficult. To reduce this difficulty investigator form a group of facial features in face-like constellations using more robust modelling approaches such as statistical analysis. Various types of face constellations have been proposed by Burl et al. They establish use of statistical shape theory on the features detected from a multiscale Gaussian derivative filter. Huang et al. also apply a Gaussian filter for pre-processing in a framework based on image feature analysis. Image Base Approach.

2.5.1 Neural Network

Neural networks gaining much more attention in many pattern recognition problems, such as OCR, object recognition, and autonomous robot driving. Since face detection can be treated as a two-class pattern recognition problem, various neural network algorithms have been proposed.

The advantage of using neural networks for face detection is the feasibility of training a system to capture the complex class conditional density of face patterns. However, one demerit is that the network architecture has to be extensively tuned (number of layers, number of nodes, learning rates, etc.) to get exceptional performance. In early days most hierarchical neural network was proposed by Agui et al. The first stage having two parallel subnetworks in which the inputs are filtered intensity values from an original image. The inputs to the second stage network consist of the outputs from the sub networks and extracted feature values. An output at the second stage shows the presence of a face in the input region. Propp and Samal developed one of the earliest neural networks for face detection [44]. Their network consists of four layers with 1,024 input units, 256 units in the first hidden layer, eight units in the second hidden layer, and two output units. Feraud and Bernier presented a detection method using auto associative neural networks. The idea is based on which shows an auto associative network with five layers is able to perform a nonlinear principal component analysis. One auto associative network is used to detect frontal- view faces and another one is used to detect faces turned up to 60 degrees to the left and right of the frontal view. After that Lin et al. presented a face detection system using probabilistic decision-based neural network (PDBNN). The architecture of PDBNN is similar to a radial basis function (RBF) network with modified learning rules and probabilistic interpretation[20].

2.6 LINEAR SUB SPACE METHOD

Eigen faces Method:

An early example of employing eigen vectors in face recognition was done by Kohonen in which a simple neural network is demonstrated to perform face recognition for aligned and normalized face images. Kirby and Sirovich suggested that images of faces can be linearly encoded using a modest number of basis images. The idea is arguably proposed first by Pearson in 1901 and then by HOTELLING in 1933. Given a collection of n by m pixel training.

Images represented as a vector of size m X n, basis vectors spanning an optimal subspace are determined such that the mean square error between the projection of the training images onto this subspace and the original images is minimized[22]. They call the set of optimal basis vectors Eigen pictures since these are simply the eigen vectors of the covariance matrix computed from the vectorized face images in the training set. Experiments with a set of 100 images show that a face image of 91 X 50 pixels can be effectively encoded using only50 Eigen pictures. A reasonable likeness (i.e., capturing 95 percent of the variance)

2.7 STATISTICAL APPROCH

Support Vector Machine (SVM):

SVMs were first introduced Osuna et al. for face detection. SVMs work as a new paradigm to train polynomial function, neural networks, or radial basis function (RBF) classifiers. SVMs works on induction principle, called structural risk minimization, which targets to minimize an upper bound on the expected generalization error. An SVM classifier is a linear classifier where the separating hyper plane is chosen to minimize the expected classification error of the unseen test patterns[14]. In Osunaet al. developed an efficient method to train an SVM for large scale problems and applied it to face detection. Based on two test sets of 10,000,000 test patterns of 19 X 19 pixels, their system has slightly lower error rates and runs approximately30 times faster than the system by Sung and Poggio. SVMs have also been used to detect faces and pedestrians in the wavelet domain.

CHAPTER 3

METHODOLOGY

3.1 VIOLA-JONES FACE DETECTION ALGORITHM

The viola-jones algorithm is used to detect the human face from an image. The system takes some face images or non-face images as an input. After taking the input images the training phase will be start in which the system detects the face. In training phase two types of sets are included that is positive image set or negative image set. In positive image set all the images are face images and in the negative image set all the images are non-faces images. In training phase, all the features are collected that is related to the face images and all these features are stored in a file. After the training phase the next phase is testing phase. In the testing phase all the stored features are applied on an input image and classified whether it is face or not. If the image passes all the threshold then it is classified as face otherwise the image is classified as non-face.

3.2 STAGES OF V-J ALGORITHM

The viola jones algorithm has four stages.

These are:

(i) Haar feature selection
(ii) An integral image
(iii) Adaboost training
(iv) Cascading classifiers.

i. Haar feature selection

All human faces share some similar properties. These regularities may be matched using Haar Features. A few properties common to human faces are:

1. The eye region is darker than the upper-cheeks.
2. The nose bridge region is brighter than the eyes.

Composition of properties forming match able facial features:

- Location and size: eyes, mouth, bridge of nose
- Value: oriented gradients of pixel intensities

The four features matched by this algorithm are then required in the image of a face. Rectangle features:

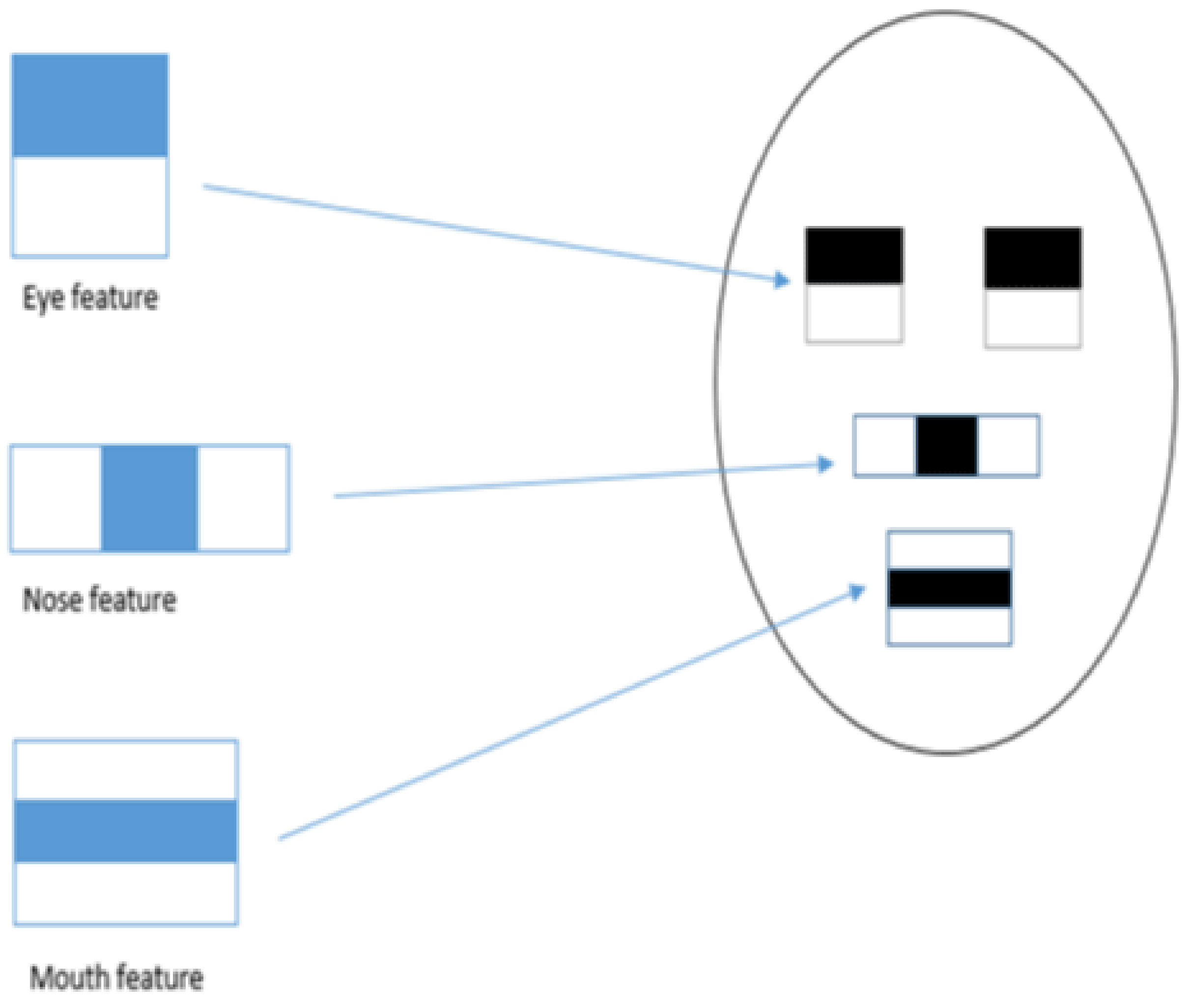


Fig.3.1: Different Types of Features

- Value = Σ (pixels in black area) - Σ (pixels in white area)
- Various types of features like two, three and four-rectangles, Viola & Jones used two-rectangle features.
- Difference in brightness between the white and black rectangles over a specific area.
- Each feature is related to a special location in the sub-window.

iii. An integral image:

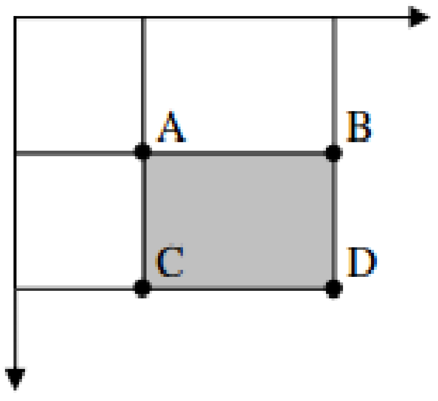
In the subsequent step of the Viola-Jones face detection algorithm is rotate the input image into an integral image. This is completed by creation of every pixel equivalent to the total addition of all pixels above and to the left of the pixel. This is established in Figure 2.

1	1	1
1	1	1
1	1	1

1	2	3
2	4	6
3	6	9

Fig.3.2: The Integral Image

This makes the computation of the addition to the entire pixels within any specified rectangle using only four values. In the integral image, these values are the pixels that correspond with the corners of the rectangle in the input image. This is established in figure 3.



Sum of grey rectangle = D - (B + C) + A

Fig.3.3: Sum Calculations

iv. Adaboost training

Viola Jones algorithm uses a 24x24 window as the base window size to begin evaluating all the features in any given image. If we think about all feasible parameters of the Haar features like situation, degree and type, then we need to calculate about 160,000+ features in any given window. By using this algorithm, we need to evaluate huge sets of features for every 24x24 sub-window in any new image. The basic idea is to eliminate a lot of features which are redundant and not useful. To select only those features that is very useful for us, which are done by Adaboost. Adaboost eliminate all the redundant features.

Adaboost is a machine learning algorithm which helps in judgment only the most excellent features between the entire those 160,000+ features. After these features are establish a weighted arrangement of all these features in used in evaluating and deciding any given window has face or not. These features are also called as weak classifiers.

A major component of the modified Adaboost algorithm is the determination of the most excellent feature and threshold. There seem to be no smart solution to this problem and Viola-Jones suggest a simple brute force method. This means that the determination of every latest weak classifier involves evaluating each feature on all the training examples in order to find the best performing feature. This is estimated to be the most time-consuming part of the training method.

iv. Cascading classifiers

The cascaded classifier is collection of stages that contains a strong classifier. The work of every phase is to verify whether a particular sub-window is definitely not a face or may be a face. When a sub-window is classified to be a non-face by a given phase it is discarded. A sub-window classified as a may be face is passed on to the next stage in the cascade. It follows that the additional stages a given sub-window passes, the higher chance that the sub-window really contains a face.

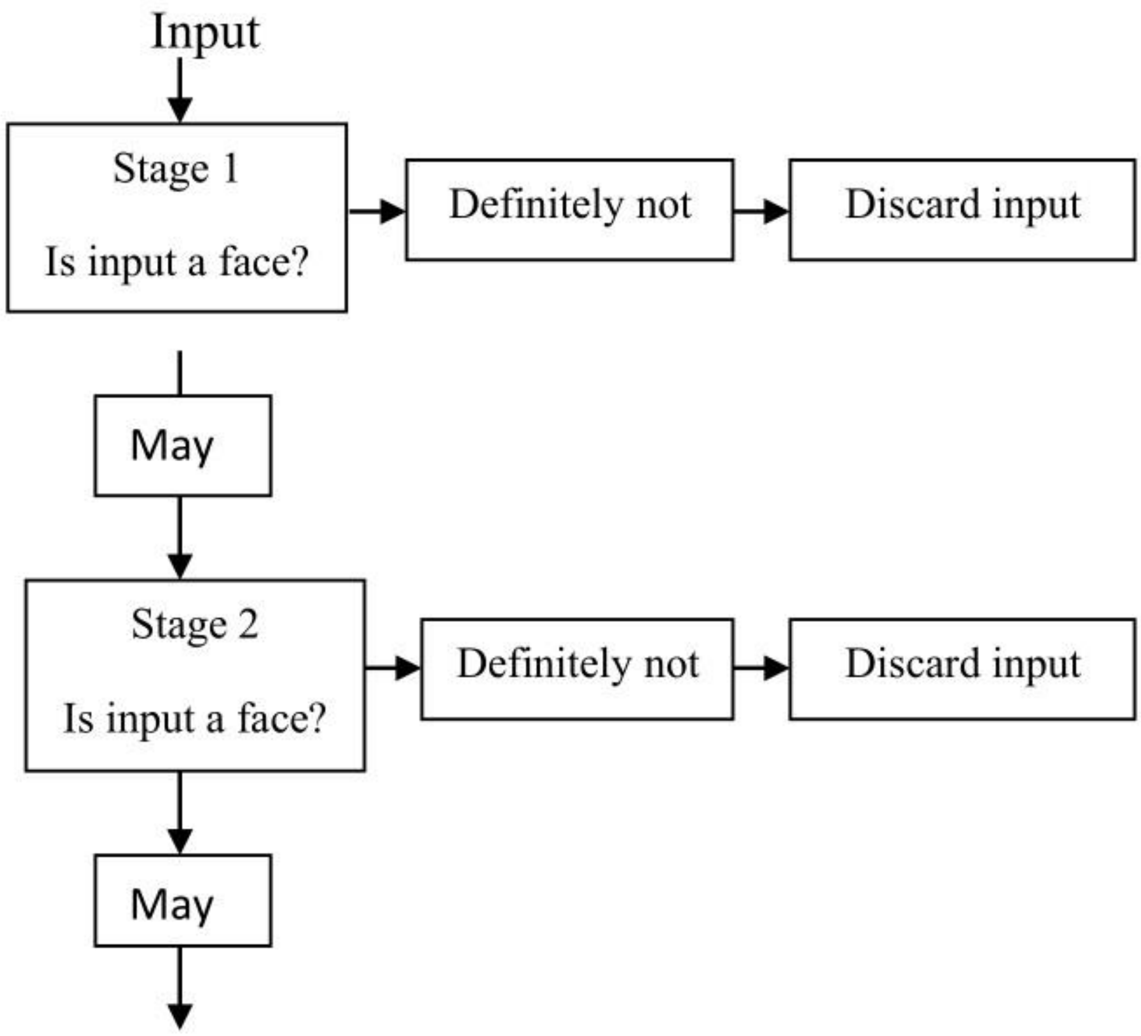


Fig.3.4: The Cascade Classifier

CHAPTER 4

SYSTEM DESIGN ANALYSIS

4.1 SYSTEM DESIGN

The system consists of the Main GUI which has two options which lead to two separate modules: one is for Image Based Face Detection and the second module is of Face Tracking. When the user presses the Face Detection button a separate GUI opens which has four different detectors namely Face, Eyes, Mouth and Nose.

4.2 FLOW OF DESIGN

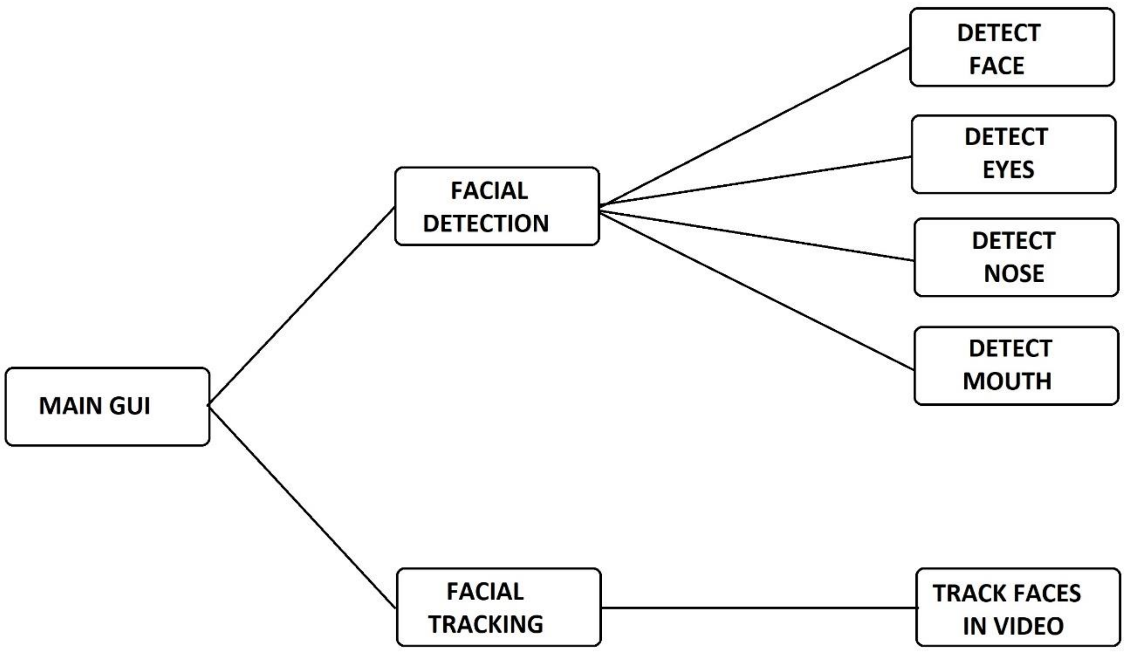


Fig. 4.1: Design of GUI

4.3 IMAGE BASED FACE DETECTION IMPLEMENTATION

Let’s see how to detect face, nose, mouth and eyes using the MATLAB built-in class and function. Based on Viola-Jones face detection algorithm, the computer vision system toolbox contains vision.CascadeObjectDetector System object which detects objects based on above mentioned algorithm.

Prerequisite: Computer vision system toolbox

4.3.1 FACE DETECTOR:

The following code is used to implement Face Detection:

```
clear all

clc

%Detect objects using Viola-Jones Algorithm


%To detect Face

FDetect = vision.CascadeObjectDetector;


%Read the input image

I = imread('HarryPotter.jpg');

%Returns Bounding Box values based on number of objects

BB = step(FDetect,I);


figure,

imshow(I); hold on


for i = 1:size(BB,1)
```

```
    rectangle('Position',BB(i,:),'LineWidth',5,'LineStyle','-','EdgeColor','r');

end

title('Face Detection');

hold off;
```

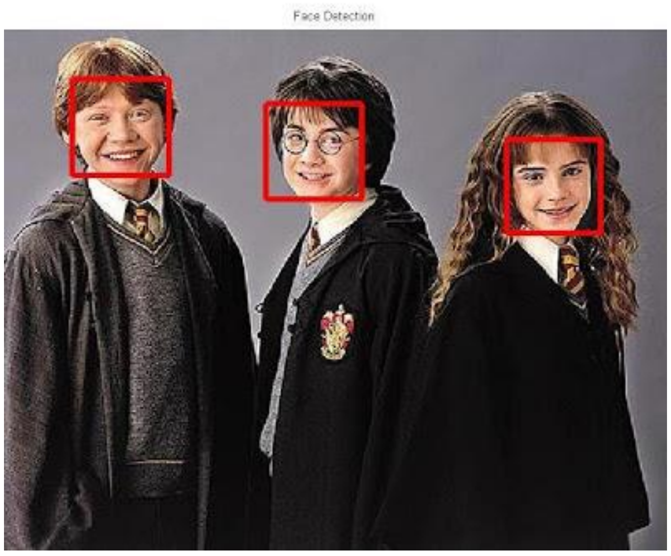


Fig. 4.2: Detected Faces

The step(Detector,I) returns Bounding Box value that contains [x,y,Height,Width] of the objects of interest.

```
BB =

    52    38    73    73
   379    84    71    71
   198    57    72    72
```

4.3.2 NOSE DETECTOR:

The following code is used to implement Nose Detection:

```
%To detect Nose
```

```
NoseDetect = vision.CascadeObjectDetector('Nose','MergeThreshold',16);



BB=step(NoseDetect,I);



figure,

imshow(I); hold on

for i = 1:size(BB,1)

    rectangle('Position',BB(i,:),'LineWidth',4,'LineStyle','-','EdgeColor','b');

end

title('Nose Detection');

hold off;
```

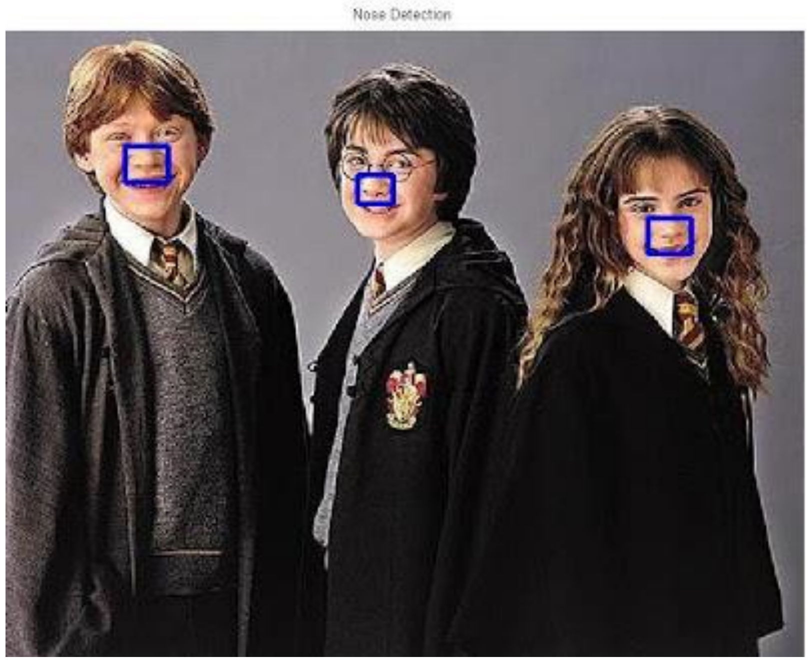


Fig. 4.3: Detected Noses

EXPLANATION:

To denote the object of interest as 'nose', the argument 'Nose' is passed.

vision.CascadeObjectDetector('Nose','MergeThreshold',16);

The default syntax for Nose detection :

vision.CascadeObjectDetector('Nose');

Based on the input image, we can modify the default values of the parameters passed to vision.CascaseObjectDetector. Here the default value for 'MergeThreshold' is 4.

When default value for 'MergeThreshold' is used, the result is not correct.

Here there are more than one detection on Hermione.

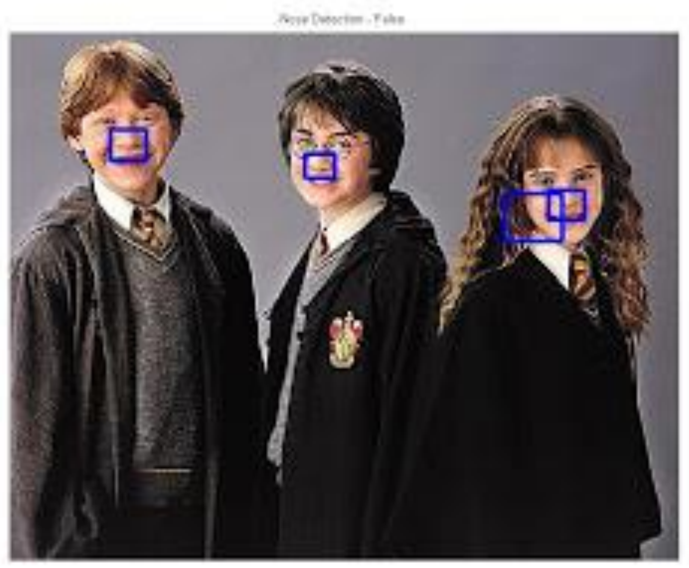


Fig. 4.4: Detected Noses

To avoid multiple detection around an object, the 'MergeThreshold' value can be overridden.

4.3.3 MOUTH DETECTOR:

The following code is used to implement Mouth Detection:

```
%To detect Mouth
MouthDetect = vision.CascadeObjectDetector('Mouth','MergeThreshold',16);

BB=step(MouthDetect,I);

figure,
imshow(I); hold on

for i = 1:size(BB,1)
 rectangle('Position',BB(i,:),'LineWidth',4,'LineStyle','-','EdgeColor','r');
end

title('Mouth Detection');
hold off;
```

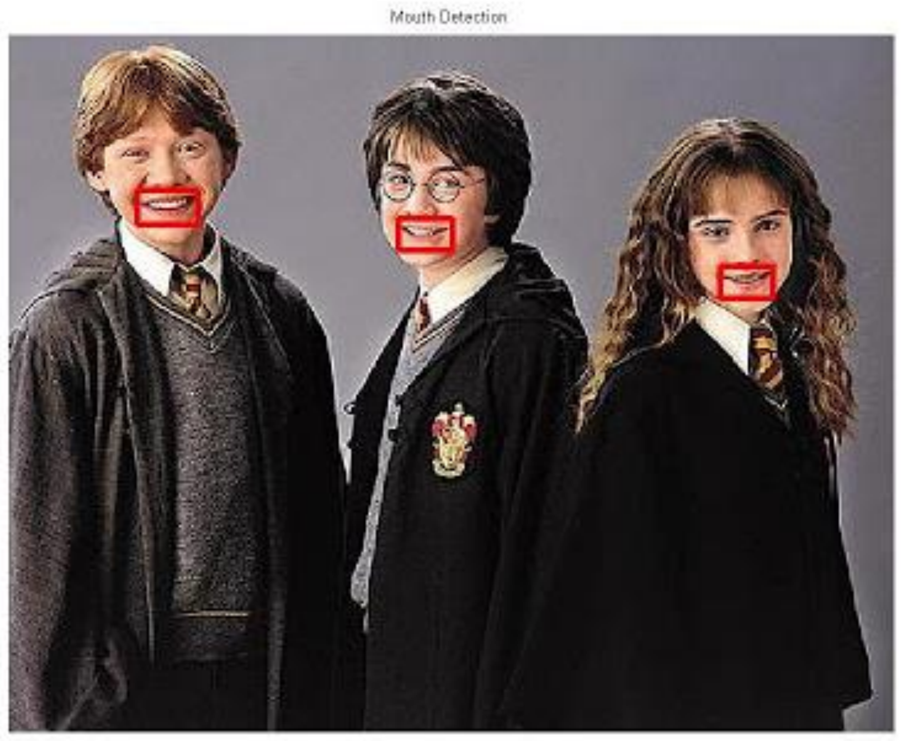


Fig. 4.5: Detected Mouths

4.3.4 EYES DETECTOR:

The following code is used to implement Mouth Detection:

```
%To detect Eyes

EyeDetect = vision.CascadeObjectDetector('EyePairBig');


%Read the input Image

I = imread('harry_potter.jpg');


BB=step(EyeDetect,I);

figure,imshow(I);
```

```
rectangle('Position',BB,'LineWidth',4,'LineStyle','-','EdgeColor','b');

title('Eyes Detection');

Eyes=imcrop(I,BB);

figure,imshow(Eyes);
```

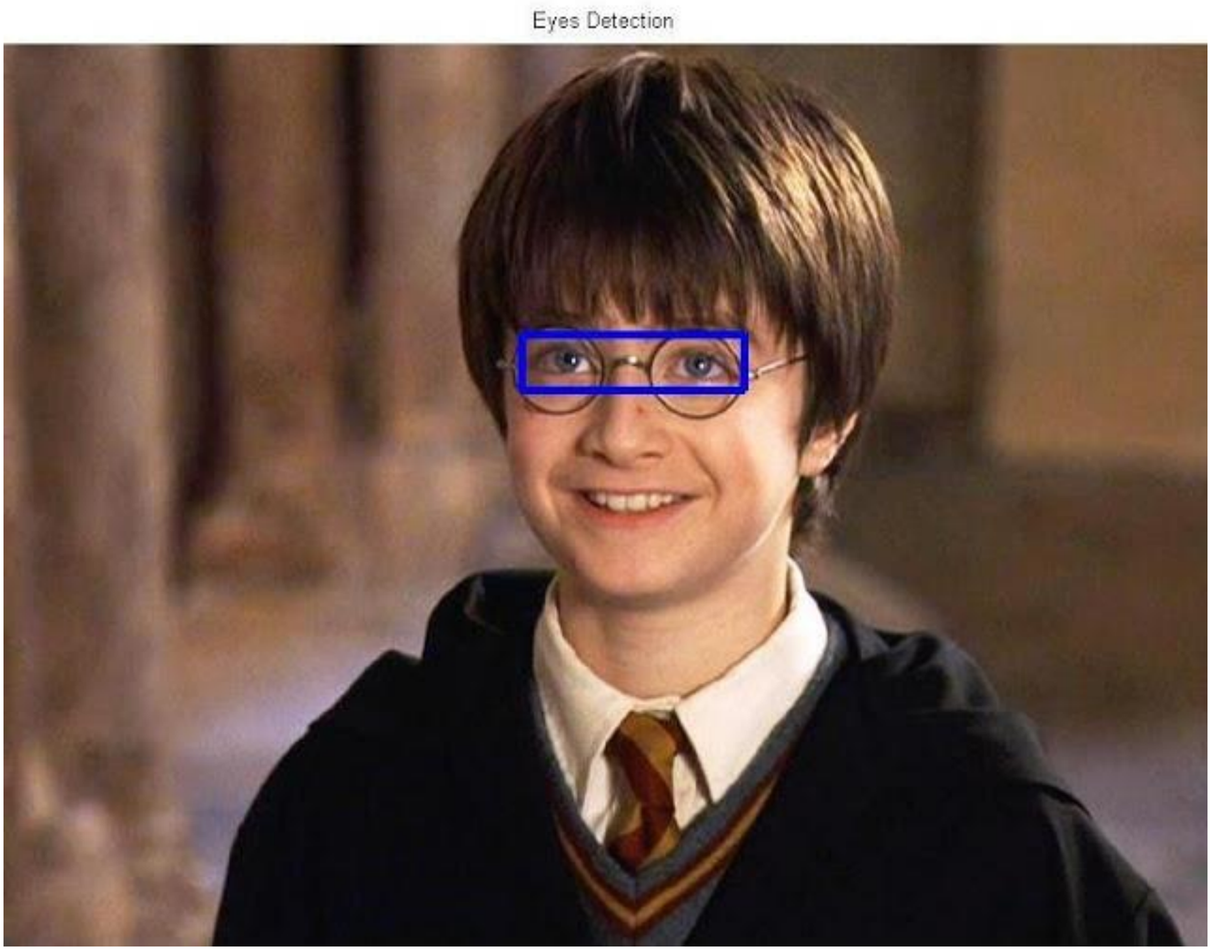


Fig. 4.6: Detected Mouths

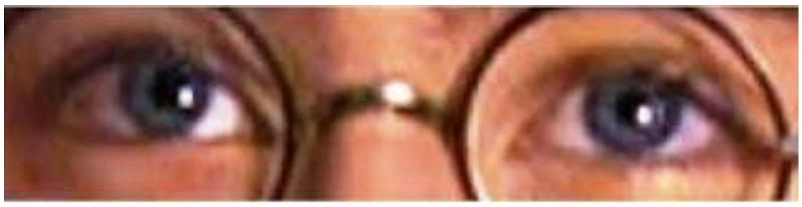


Fig. 4.7: Cropped Eye

4.4 REAL-TIME FACE TRACKING CODE IMPLEMENTATION

First, we have to find the format supported by the camera and its device ID using the command given below:

[stextbox id=”grey”]info = imaqhwinfo(‘winvideo’)[/stextbox]

After finding the device ID, we can change the device ID.

The device ID of my system is {1}, so I have written ‘1’ in the code, as mentioned below:

[stextbox id=”grey”]vid = videoinput(‘winvideo’,1,’YUY2_

640×480’);[/stextbox]

We also have other formats in MATLAB. You can check which format your camera supports by using the commands below:

[stextbox id=”grey”]

info.DeviceInfo (1)

info.DeviceInfo.SupportedFormats

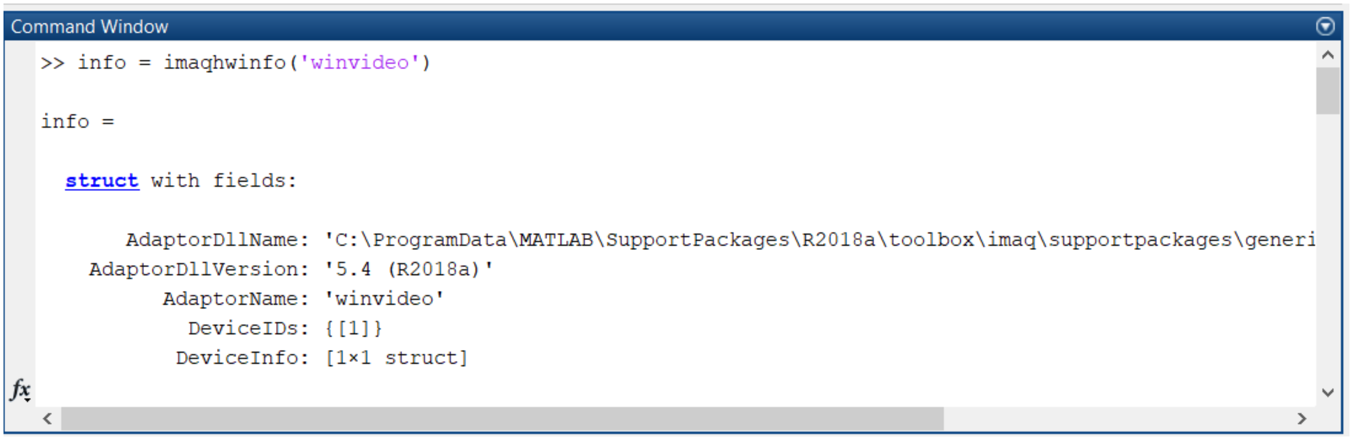
[/stextbox]

We can see that format ‘YUY2_160x120’ is the one supported by the camera by default. But, there are other formats (resolutions) that our camera can support, as shown in the last line of this screenshot. If we select a different format and device number, we should make changes in the source code accordingly. To detect a face or a particular feature on the faces of people, we use the following steps in MATLAB program:

1. Define and set-up the cascade object detector using the constructor:

[stextbox id=”grey”]detector=vision.CascadeObjectDetector[/stextbox]

It creates a system object detector that detects objects using Viola-Jones algorithm. Its classification model property controls the type of object to detect. By default, the detector is configured to detect faces.

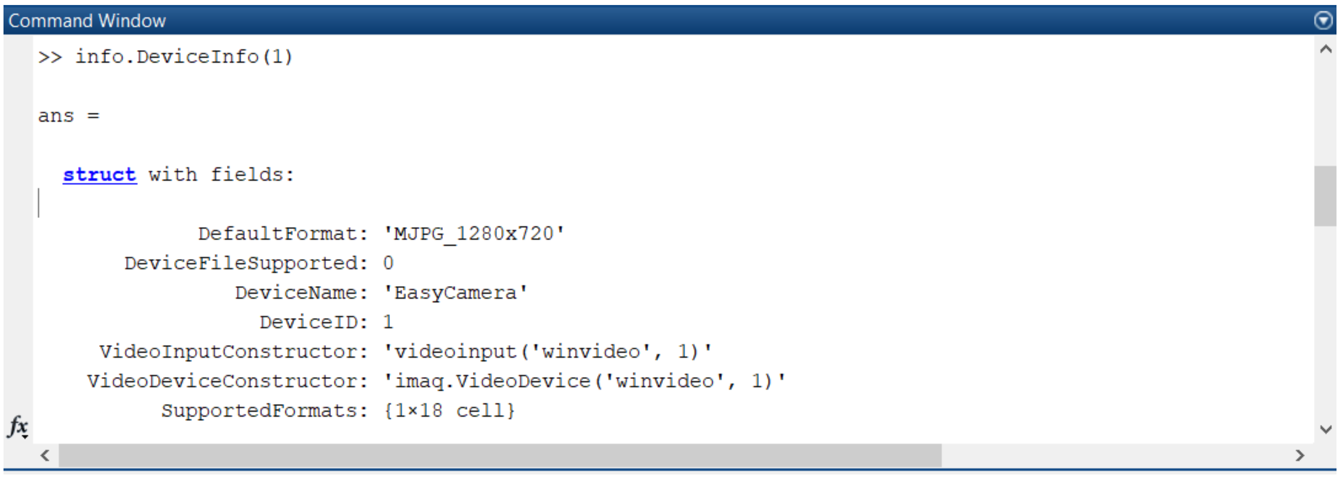


```
>> info = imaqhwinfo('winvideo')

info =

  struct with fields:

       AdaptorDllName: 'C:\ProgramData\MATLAB\SupportPackages\R2018a\toolbox\imaq\supportpackages\generi
    AdaptorDllVersion: '5.4 (R2018a)'
          AdaptorName: 'winvideo'
            DeviceIDs: {[1]}
           DeviceInfo: [1×1 struct]
```

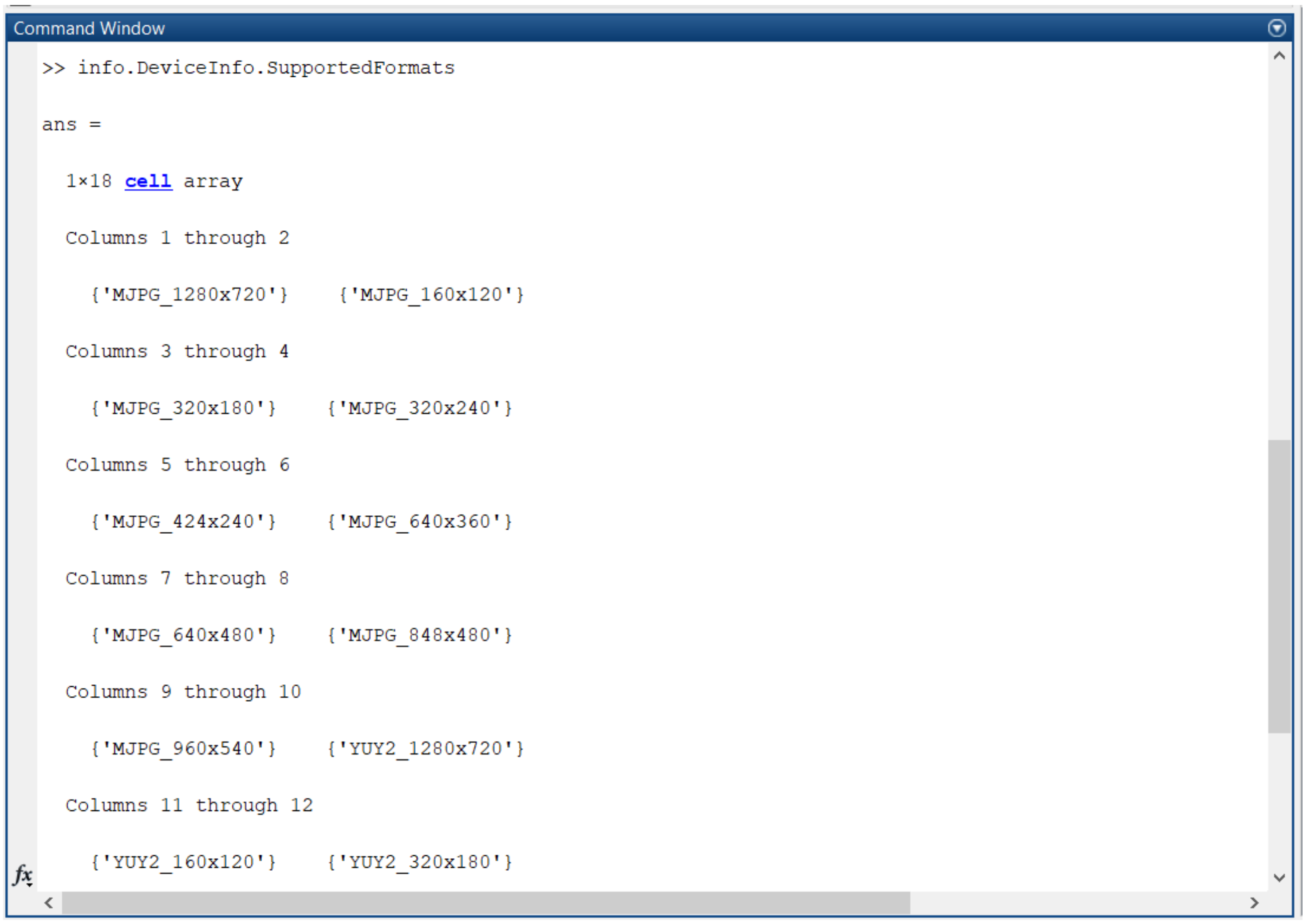


```
>> info.DeviceInfo(1)

ans =

  struct with fields:

             DefaultFormat: 'MJPG_1280x720'
       DeviceFileSupported: 0
                DeviceName: 'EasyCamera'
                  DeviceID: 1
     VideoInputConstructor: 'videoinput('winvideo', 1)'
    VideoDeviceConstructor: 'imaq.VideoDevice('winvideo', 1)'
          SupportedFormats: {1×18 cell}
```



```
>> info.DeviceInfo.SupportedFormats

ans =

  1×18 cell array

  Columns 1 through 2

    {'MJPG_1280x720'}    {'MJPG_160x120'}

  Columns 3 through 4

    {'MJPG_320x180'}    {'MJPG_320x240'}

  Columns 5 through 6

    {'MJPG_424x240'}    {'MJPG_640x360'}

  Columns 7 through 8

    {'MJPG_640x480'}    {'MJPG_848x480'}

  Columns 9 through 10

    {'MJPG_960x540'}    {'YUY2_1280x720'}

  Columns 11 through 12

    {'YUY2_160x120'}    {'YUY2_320x180'}
```

Fig. 4.8: Formats Supported by the Camera

2. Call the step method with input image I, cascade object detector, points PTS and any other optional properties.

Below is the syntax for using the step method. We use the step syntax with input image I, selected cascade object detector and other optional properties to perform detection.

[stextbox id="grey"]BBOX = step (detector, I)[/stextbox]

It returns BBOX, an M-by-4 matrix defining M-bounding boxes, containing detected objects. This method performs multi-scale object detection on input image I. Each row of output matrix BBOX contains a four-element vector (x, y, width and height) that specifies in pixels, the upper-left corner and size of a bounding box. Input image I must be a gray scale or true colour (RGB) image.

3. The third step is:

[stextbox id="grey"]insertObjectAnnotation(I,'rectangle',

Position,Label)[/stextbox]

It inserts rectangles and corresponding labels at the location indicated by the position matrix. The position input must be an M-by-4 matrix, where each row (M) specifies a rectangle as a four-element vector (x, y, width and height). Elements x and y indicate the upper-left corner of the rectangle, and the width and height specify the size.

4.5 GUI AND CODE IMPLEMENTATION

Here we give the Graphical User Interfaces and the MATLAB coding of the project.

4.5.1 The Main GUI

The Main GUI is given in the image below:

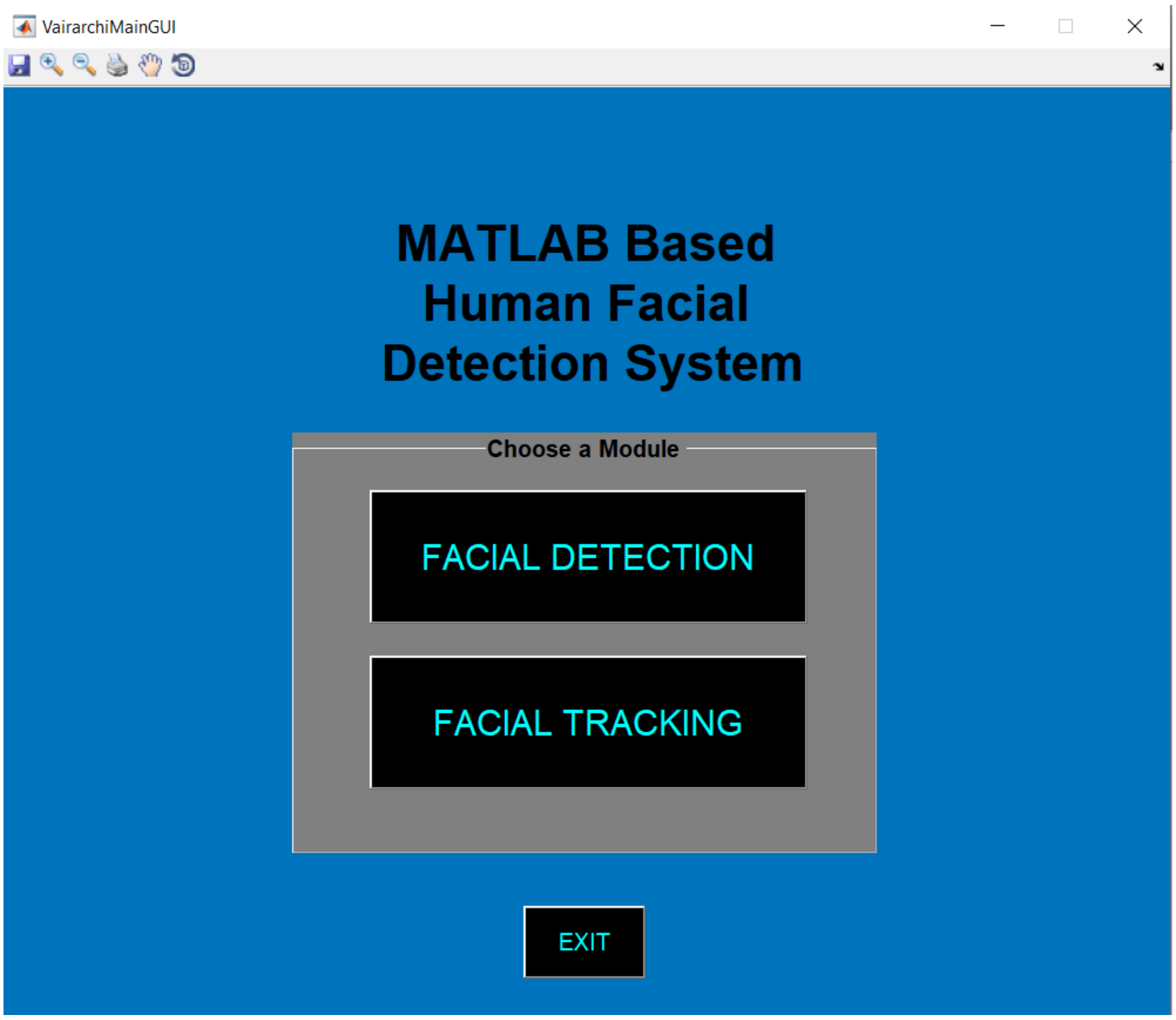


Fig. 4.9: The Main GUI

4.5.2 Main GUI MATLAB Code

The MATLAB code behind the Main GUI is:

```
function varargout = VairarchiMainGUI(varargin)
% VAIRARCHIMAINGUI MATLAB code for VairarchiMainGUI.fig
%      VAIRARCHIMAINGUI, by itself, creates a new VAIRARCHIMAINGUI or
raises the existing
%      singleton*.
%
%      H = VAIRARCHIMAINGUI returns the handle to a new
VAIRARCHIMAINGUI or the handle to
%      the existing singleton*.
%
%      VAIRARCHIMAINGUI('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in VAIRARCHIMAINGUI.M with the given
input arguments.
%
```

```
%      VAIRARCHIMAINGUI('Property','Value',...) creates a new
VAIRARCHIMAINGUI or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before VairarchiMainGUI_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to VairarchiMainGUI_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help VairarchiMainGUI

% Last Modified by GUIDE v2.5 09-Mar-2019 22:31:59

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @VairarchiMainGUI_OpeningFcn, ...
                   'gui_OutputFcn',  @VairarchiMainGUI_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

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

```
% End initialization code - DO NOT EDIT



% --- Executes just before VairarchiMainGUI is made visible.
function VairarchiMainGUI_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to VairarchiMainGUI (see VARARGIN)

% Choose default command line output for VairarchiMainGUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes VairarchiMainGUI wait for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
function varargout = VairarchiMainGUI_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



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

display Goodbye!!
close(handles.figure1)

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

VairarchiDetector



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

VairarchiTracking
```

4.5.3 Face Detection GUI

The Facial Detection GUI opens after the user presses the 'FACIAL DETECTION' button in the Main GUI. It consists of 6 buttons in total. The first Button named 'Load Original Image' loads the selected image in the first Axes, the next 4 buttons are the Detector buttons which detect Face, Eyes, Mouth and Nose in the loaded image respectively. The last button is used to exit the module and return to the Main GUI.

The GUI is given in the image below:

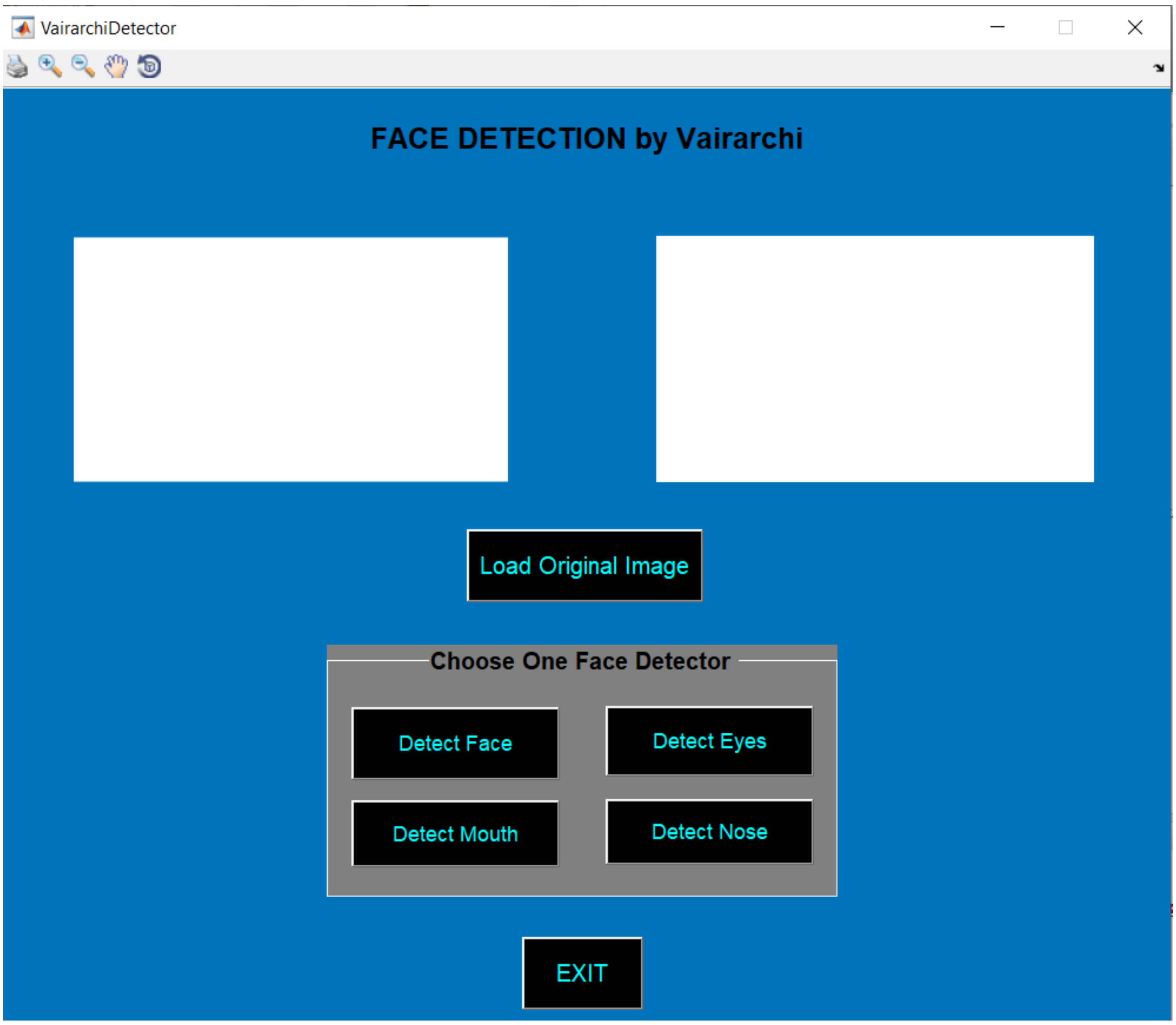


Fig. 4.10: Face Detection GUI

4.5.4 Face Detection MATLAB Code

The MATLAB code behind the Facial Detection GUI is:

```
function varargout = VairarchiDetector(varargin)
% VAIRARCHIDETECTOR MATLAB code for VairarchiDetector.fig
%      VAIRARCHIDETECTOR, by itself, creates a new VAIRARCHIDETECTOR
or raises the existing
%      singleton*.
%
%      H = VAIRARCHIDETECTOR returns the handle to a new
VAIRARCHIDETECTOR or the handle to
%      the existing singleton*.
%
```

```
%      VAIRARCHIDETECTOR('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in VAIRARCHIDETECTOR.M with the given
input arguments.
%
%      VAIRARCHIDETECTOR('Property','Value',...) creates a new
VAIRARCHIDETECTOR or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before VairarchiDetector_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to VairarchiDetector_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help VairarchiDetector

% Last Modified by GUIDE v2.5 22-Feb-2019 16:48:07

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @VairarchiDetector_OpeningFcn,
...
                   'gui_OutputFcn',  @VairarchiDetector_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before VairarchiDetector is made visible.
function VairarchiDetector_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to VairarchiDetector (see
VARARGIN)

% Choose default command line output for VairarchiDetector
handles.output = hObject;
axes(handles.axes1);
imshow('blank.jpg');
axis off;

axes(handles.axes2);
imshow('blank.jpg');
axis off;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes VairarchiDetector wait for user response (see UIRESUME)
% uiwait(handles.figure1);



% --- Outputs from this function are returned to the command line.
```

```
function varargout = VairarchiDetector_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



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

FileName=getappdata(hObject,'FN')

FDetect = vision.CascadeObjectDetector;

%Returns Bounding Box values based on number of objects
BB = step(FDetect,FileName);

axes(handles.axes2)
imshow(FileName);
hold on
for i = 1:size(BB,1)
    rectangle('Position',BB(i,:),'LineWidth',3,'LineStyle','-
','EdgeColor','r');
end
hold off;
%Face=imcrop(FileName,BB);
%figure,imshow(Face);

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

FileName=getappdata(hObject,'FN')

%To detect Eyes
EyeDetect = vision.CascadeObjectDetector('EyePairBig');



BB=step(EyeDetect,FileName);

axes(handles.axes2)
imshow(FileName);
rectangle('Position',BB,'LineWidth',4,'LineStyle','-
','EdgeColor','r');
title('Eyes Detection');
%Eyes=imcrop(I,BB);
%figure,imshow(Eyes);



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

FileName=getappdata(hObject,'FN')

%To detect Mouth
MouthDetect =
vision.CascadeObjectDetector('Mouth','MergeThreshold',120);

BB=step(MouthDetect,FileName);

axes(handles.axes2)
imshow(FileName);
hold on
for i = 1:size(BB,1)
```

```
 rectangle('Position',BB(i,:),'LineWidth',4,'LineStyle','-
','EdgeColor','r');
end
title('Mouth Detection');
hold off;




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

FileName=getappdata(hObject,'FN')

NoseDetect = vision.CascadeObjectDetector('Nose','MergeThreshold',16);

BB=step(NoseDetect,FileName);

axes(handles.axes2)
imshow(FileName);
hold on
for i = 1:size(BB,1)
    rectangle('Position',BB(i,:),'LineWidth',4,'LineStyle','-
','EdgeColor','b');
end
title('Nose Detection');
hold off;



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

```
[filename filepath]=uigetfile({'*.*';'*.jpg';'*.png';'*.bmp'}, 'Search
Image to be Displayed');
fullname = [filepath filename];
%now we read the image fullname
ImageFile = imread(fullname);
%Now let's display the image
axes(handles.axes1)
imshow(ImageFile)

setappdata(handles.face,'FN',ImageFile);
setappdata(handles.eyes,'FN',ImageFile);
setappdata(handles.mouth,'FN',ImageFile);
setappdata(handles.nose,'FN',ImageFile);



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

display 'Exiting Module 1'
close(handles.figure1)
```

4.5.5 Face Tracking GUI

The Facial Tracking GUI opens after the user presses the 'FACIAL TRACKING' button in the Main GUI. It consists of 3 buttons in total. After the GUI appears click on 'START WEBCAM' button to initialise camera settings. Next, click on 'TRACK FACE' button and the camera will detect the face. The face will be detected and displayed on the right side of the screen. After the process of tracking the user should remember to click 'STOP' button to stop the tracking process in and shut down the algorithm. After that as a final step we should click the 'EXIT' button to exit the Tracking module and go back to the Main GUI.

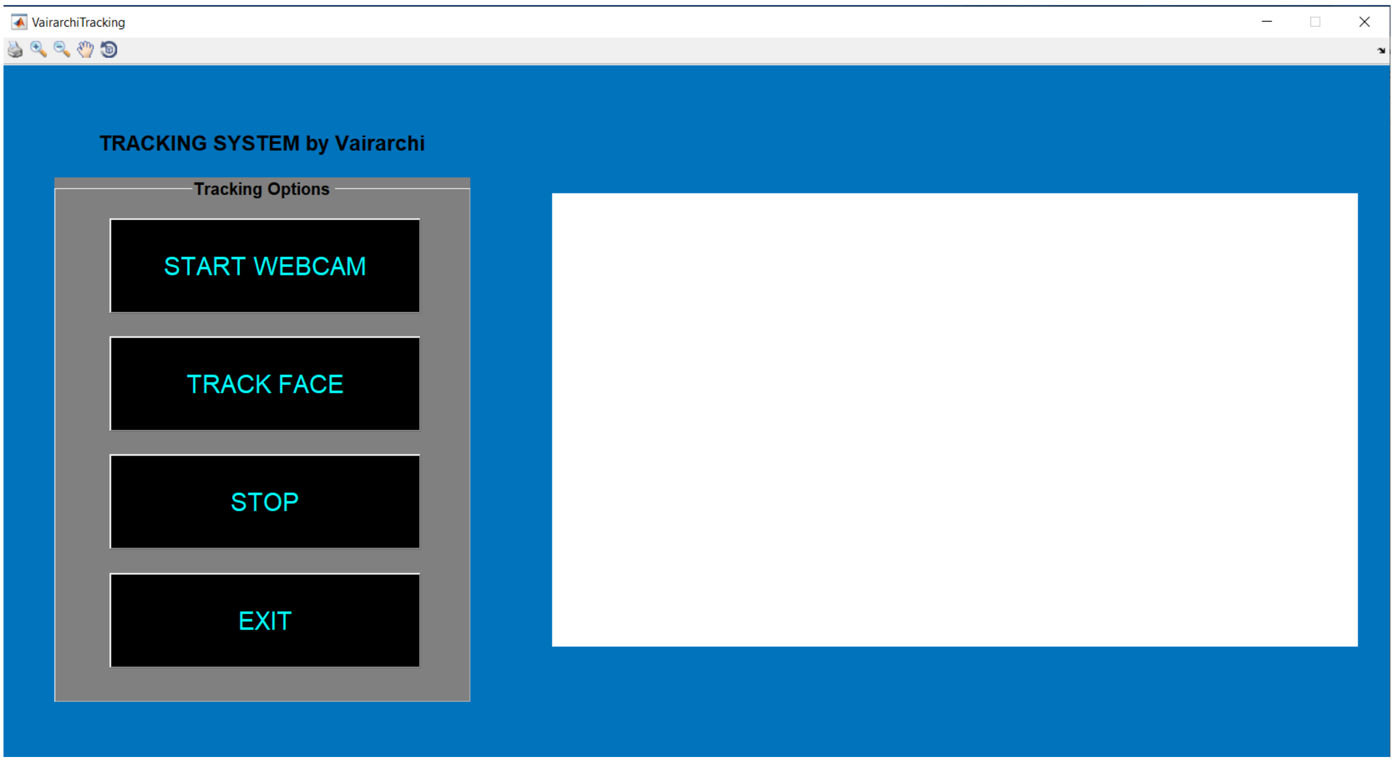


Fig. 4.11: Face Tracking GUI

4.5.6 Face Tracking MATLAB Code

The MATLAB code behind the Facial Tracking GUI is:

```
function varargout = VairarchiTracking(varargin)
% VAIRARCHITRACKING MATLAB code for VairarchiTracking.fig
%      VAIRARCHITRACKING, by itself, creates a new VAIRARCHITRACKING
or raises the existing
%      singleton*.
%
%      H = VAIRARCHITRACKING returns the handle to a new
VAIRARCHITRACKING or the handle to
%      the existing singleton*.
%
%      VAIRARCHITRACKING('CALLBACK',hObject,eventData,handles,...)
calls the local
%      function named CALLBACK in VAIRARCHITRACKING.M with the given
input arguments.
%
%      VAIRARCHITRACKING('Property','Value',...) creates a new
VAIRARCHITRACKING or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before VairarchiTracking_OpeningFcn gets
called.  An
%      unrecognized property name or invalid value makes property
application
%      exit.  All inputs are passed to VairarchiTracking_OpeningFcn
via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help VairarchiTracking
% Last Modified by GUIDE v2.5 05-Apr-2019 22:42:58

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @VairarchiTracking_OpeningFcn,
...
                   'gui_OutputFcn',  @VairarchiTracking_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before VairarchiTracking is made visible.
function VairarchiTracking_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to VairarchiTracking (see
VARARGIN)

% Choose default command line output for VairarchiTracking
handles.output = hObject;
axes(handles.axes1);
imshow('blank.jpg');
axis off;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes VairarchiTracking wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = VairarchiTracking_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


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

handles.vid = videoinput('winvideo' , 1, 'YUY2_640X480');

guidata(hObject, handles);




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

triggerconfig(handles.vid ,'manual');
set(handles.vid, 'TriggerRepeat',inf);
set(handles.vid, 'FramesPerTrigger',1);
handles.vid.ReturnedColorspace = 'rgb';
 handles.vid.Timeout = 5;
start(handles.vid);

while(1)
facedetector = vision.CascadeObjectDetector;
trigger(handles.vid);
handles.im = getdata(handles.vid, 1);
bbox = step(facedetector, handles.im);
hello = insertObjectAnnotation(handles.im,'rectangle',bbox,'Face');
imshow(hello);
end

guidata(hObject, handles);



% --- Executes on button press in exit.
function stop_Callback(hObject, eventdata, handles)
% hObject    handle to exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
handles.output = hObject;
stop(handles.vid),clear handles.vid %, ,delete(handles.vid)
guidata(hObject, handles);









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

clc
display 'Exiting Tracking Module!!'
close(handles.figure1)
```

CHAPTER 5

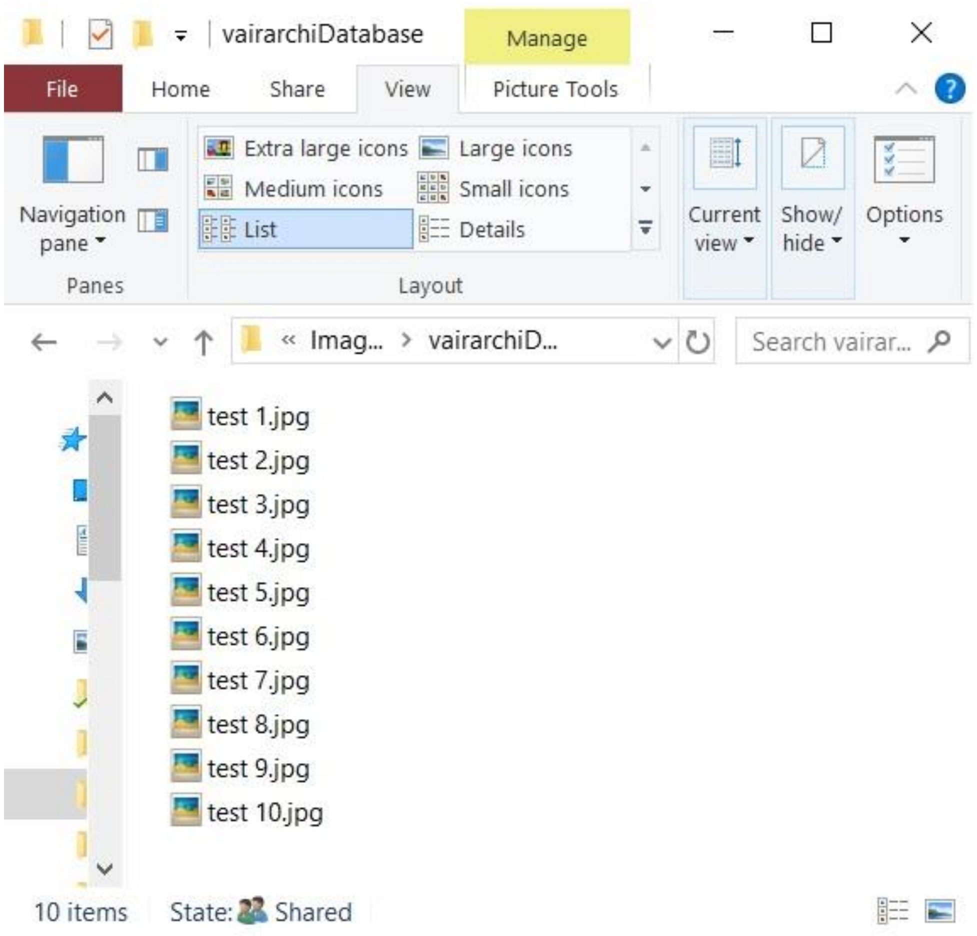
PERFORMANCE AND RESULTS

5.1 PREREQUISITES

We carry out the face detection experiment on a test database. In order to conduct face detection experiment, a comprehensive database of human faces have been created. Experiment has been carried out on 10 images. The accuracy may vary for different images.

5.1.1 TEST Database

The following is our Test Database on which we carry out our Face Detection Experiment:



test 1.jpg
test 2.jpg
test 3.jpg
test 4.jpg
test 5.jpg
test 6.jpg
test 7.jpg
test 8.jpg
test 9.jpg
test 10.jpg

10 items State: Shared

Fig. 5.1: The Test Database

The images in database are quite different from each other, with each image have different occlusion, pose and illumination. This makes the database versatile.

5.1.2 System Accuracy

When the images of the Database are worked upon by the system, we need to check the accuracy of out proposed system. For calculating the accuracy of our proposed system we use the following formula:

$$\text{Detection Accuracy} = \frac{\text{No. of Actual Detected Features - No. of False Positives}}{\text{No. of Features in Image}}$$

Where,

Detection accuracy is the percentage of accuracy of the Detector

No. of Actual Detected features are those detected features which are not false positives

No. of False positives are the detected features which are not actually features

No. of Features in Image are the original Features present in the image

This is the formula which will give us our accuracy.

5.2 PERFORMANCE OF FACIAL DETECTION

We will see the performance of the first module which gives us the image-based face detection. The detectors are Face, Eyes, Nose and Mouth.

5.2.1 Face Detector

We test the face detector on our Test Database. All the 10 images in the Database are processed. Since each image has varied pose, illumination and occlusion the result varies throughout. There are multiple false positives depending upon the previously mentioned factors. We will see one working of the Face detector on one image in detail, with its output matrix in the command window.

The following is the Output Snapshot of the Face detector in work:

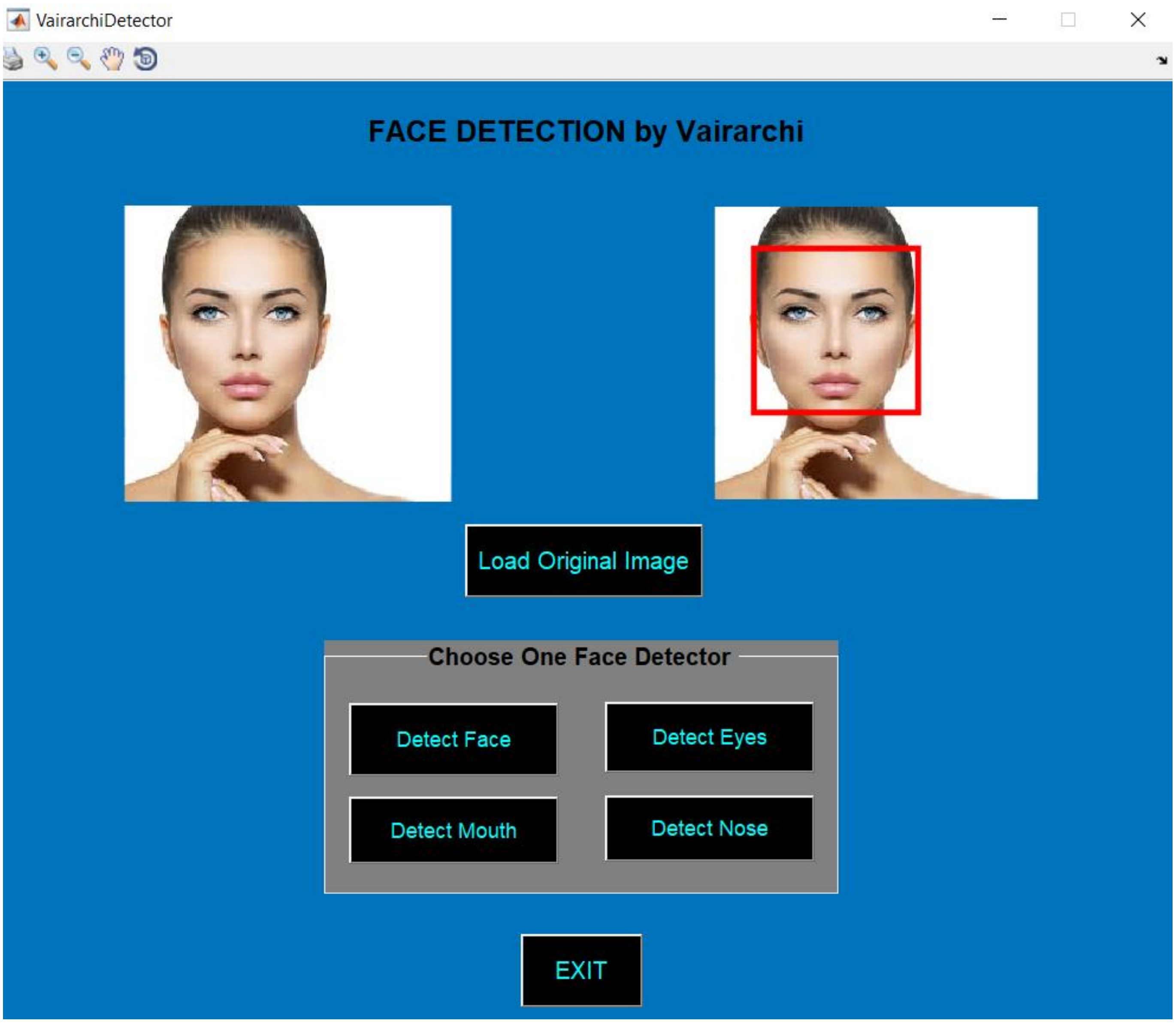


Fig. 5.2: Face Detector Output

At the end we get the output in the form of a matrix of grey scale values.

5.2.1.1 Performance of the Face Detector

The Face detector was worked upon each image in our test database. The accuracy varies for different images depending upon the factors. The performance can be seen by the following table:

Image Name	No. of Actual Detected Faces	No. of False Positives	Total No. of Faces	Accuracy (in %)
test1.jpg	2	0	2	100%
test2.jpg	1	0	1	100%
test3.jpg	3	1	3	66.66%
test4.jpg	4	2	5	40%
test5.jpg	2	0	2	100%
test6.jpg	3	1	3	66.66%
test7.jpg	4	2	4	50%
test8.jpg	3	0	3	100%
test9.jpg	4	1	4	75%
test10.jpg	5	2	5	60%

Table 5.1: Face Detector Accuracy

After carrying out the Face Detection experiment, we find out that the overall accuracy for Face detector on the test database is **75.83 %**.

5.2.2 Eyes Detector

We test the Eyes detector on our Test Database. All the 10 images in the Database are processed. Since each image has varied pose, illumination and occlusion the result varies throughout. There are multiple false positives depending upon the previously mentioned factors. We will see working of the Eyes detector on one image in detail, with its output matrix in the command window.

The following is the Output Snapshot of the Eyes detector in work:

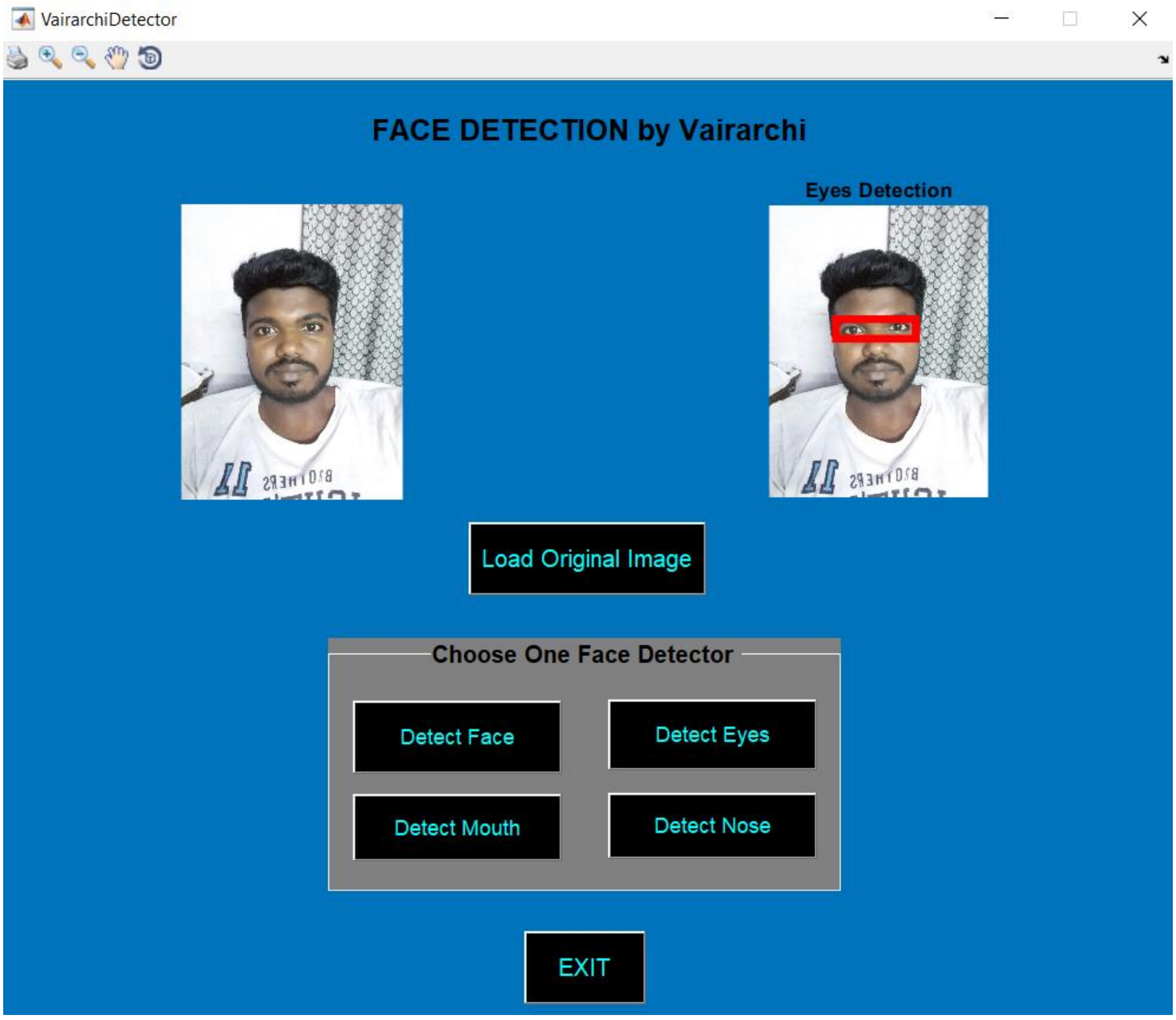


Fig. 5.3: Eyes Detector Output

At the end we get the output in the form of a matrix of grey scale values.

5.2.2.1 Performance of the Eyes Detector

The Eyes detector was worked upon each image in our test database. The accuracy varies for different images depending upon the factors. The performance can be seen by the following table:

Image Name	No. of Actual Detected Eyes	No. of False Positives	Total No. of Eyes	Accuracy (in %)
test1.jpg	2	1	2	50%
test2.jpg	1	0	1	100%
test3.jpg	0	0	3	--
test4.jpg	3	1	5	40%
test5.jpg	2	1	2	50%
test6.jpg	0	0	3	--
test7.jpg	1	0	4	25%
test8.jpg	2	1	3	33.33%
test9.jpg	3	1	4	50%
test10.jpg	0	0	5	--

Table 5.2: Eyes Detector Accuracy

After carrying out the Eyes Detection experiment, we find out that the overall accuracy for Eye detector on the test database is **49.76 %**.

5.2.3 Mouth Detector

We test the Mouth detector on our Test Database. All the 10 images in the Database are processed. Since each image has varied pose, illumination and occlusion the result varies throughout. There are multiple false positives depending upon the previously mentioned factors. We will see working of the Mouth detector on one image in detail, with its output matrix in the command window.

The following is the Output Snapshot of the Mouth detector in work:

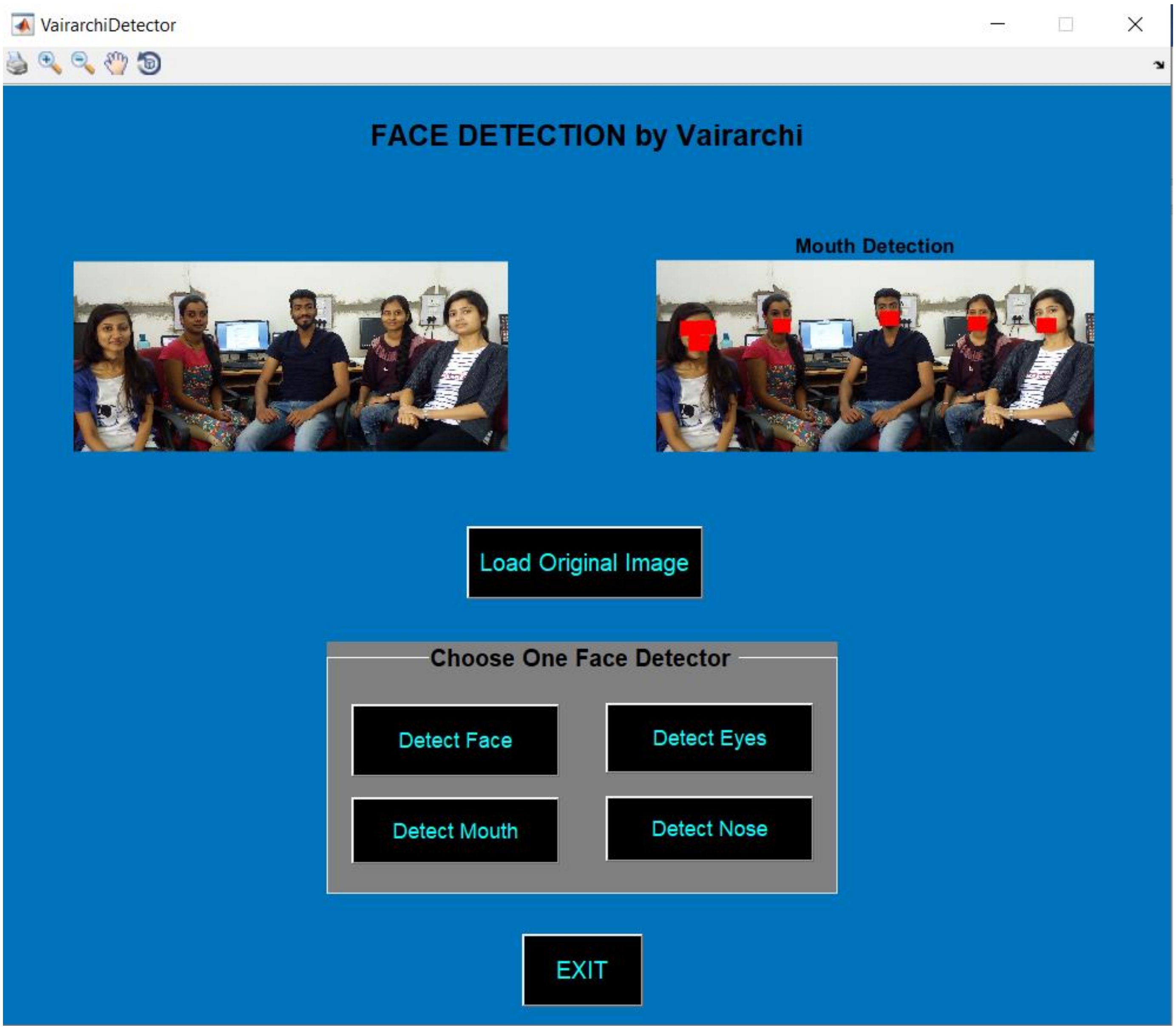


Fig. 5.3 Mouth Detector Output

At the end we get the output in the form of a matrix of grey scale values.

5.2.3.1 Performance of the Mouth Detector

The Mouth detector was worked upon each image in our test database. The accuracy varies for different images depending upon the factors. The performance can be seen by the following table:

Image Name	**No. of Actual Detected Mouth**	**No. of False Positives**	**Total No. of Mouths**	**Accuracy (in %)**
test1.jpg	2	1	2	50%
test2.jpg	1	0	1	100%
test3.jpg	1	0	3	33.33%
test4.jpg	3	0	5	60%
test5.jpg	2	0	2	100%
test6.jpg	3	1	3	66.66%
test7.jpg	4	3	4	25%
test8.jpg	3	1	3	66.66%
test9.jpg	4	2	4	50%
test10.jpg	5	2	5	60%

Table 5.3: Mouth Detector Accuracy

After carrying out the Mouth Detection experiment, we find out that the overall accuracy for Mouth detector on the test database is **61.16 %**.

5.2.4 Nose Detector

We test the Nose detector on our Test Database. All the 10 images in the Database are processed. Since each image has varied pose, illumination and occlusion the result varies throughout. There are multiple false positives depending upon the previously mentioned factors. We will see working of the Nose detector on one image in detail, with its output matrix in the command window.

The following is the Output Snapshot of the Nose detector in work:

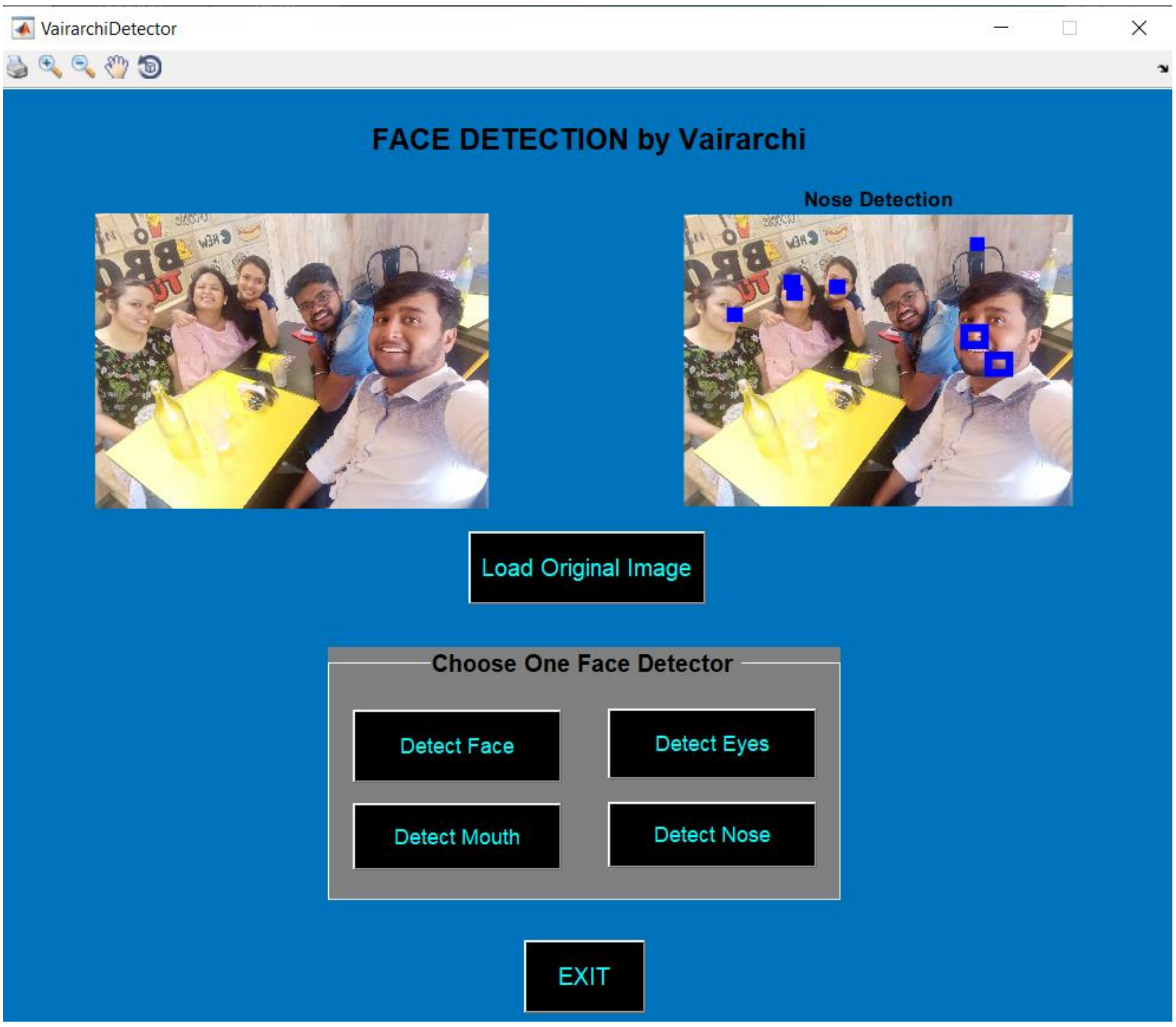


Fig. 5.4: Nose Detector Output

At the end we get the output in the form of a matrix of grey scale values.

5.2.4.1 Performance of the Nose Detector

The Nose detector was worked upon each image in our test database. The accuracy varies for different images depending upon the factors. The performance can be seen by the following table:

Image Name	No. of Actual Detected Noses	No. of False Positives	Total No. of Noses	Accuracy (in %)
test1.jpg	2	1	2	50%
test2.jpg	1	0	1	100%
test3.jpg	3	1	3	66.66%
test4.jpg	4	2	5	40%
test5.jpg	2	1	2	50%
test6.jpg	3	2	3	33.33%
test7.jpg	4	1	4	75%
test8.jpg	2	1	3	33.33%
test9.jpg	4	2	4	50%
test10.jpg	5	2	5	60%

Table 5.4: Nose Detector Accuracy

After carrying out the Nose Detection experiment, we find out that the overall accuracy for Nose detector on the test database is **61.16 %**.

5.3 PERFORMANCE OF FACIAL TRACKING

We will see the performance of the second module which gives us the real-time Face tracking using the webcam. We implement it using MATLAB. The video input is taken from webcam.

Here are the snapshots from the real time Face tracking:

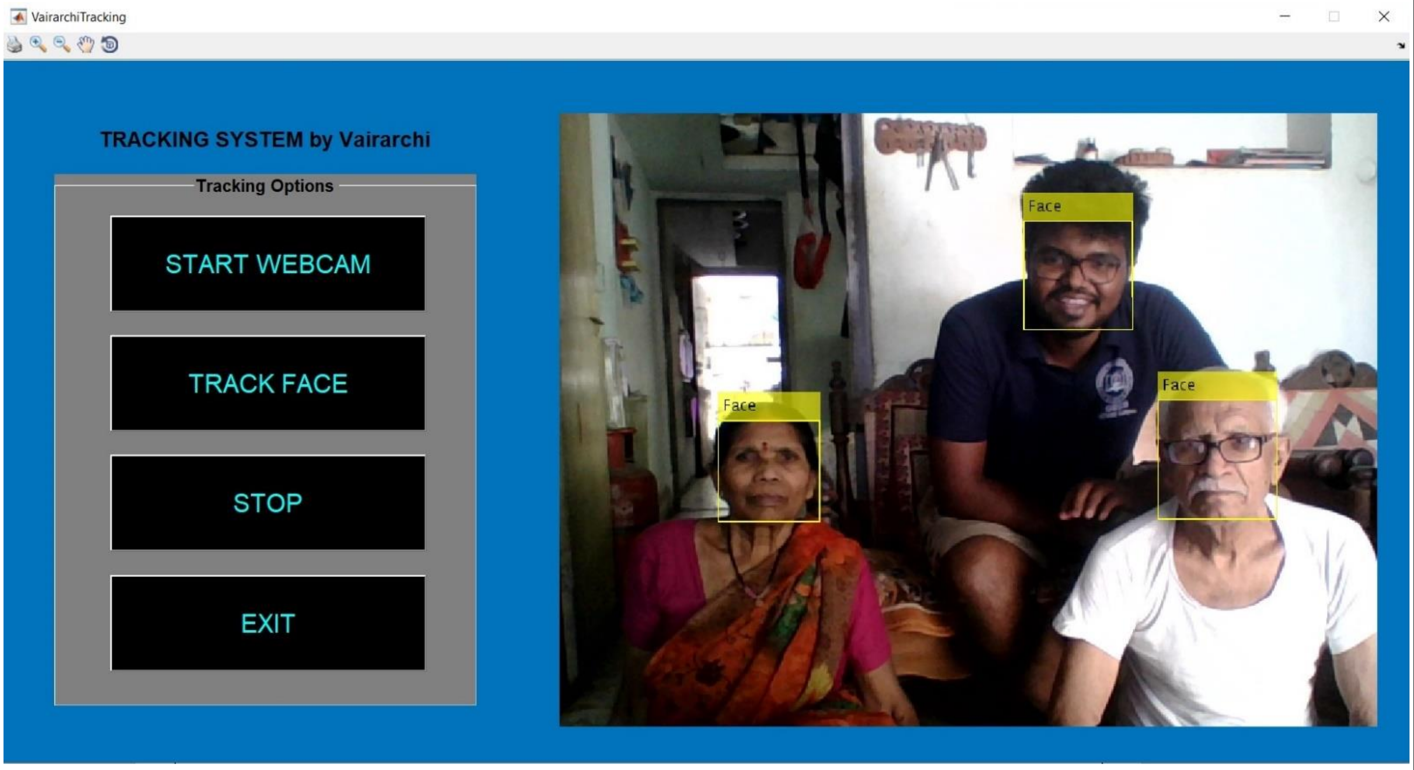


Fig. 5.5: Snapshot from first Video Input

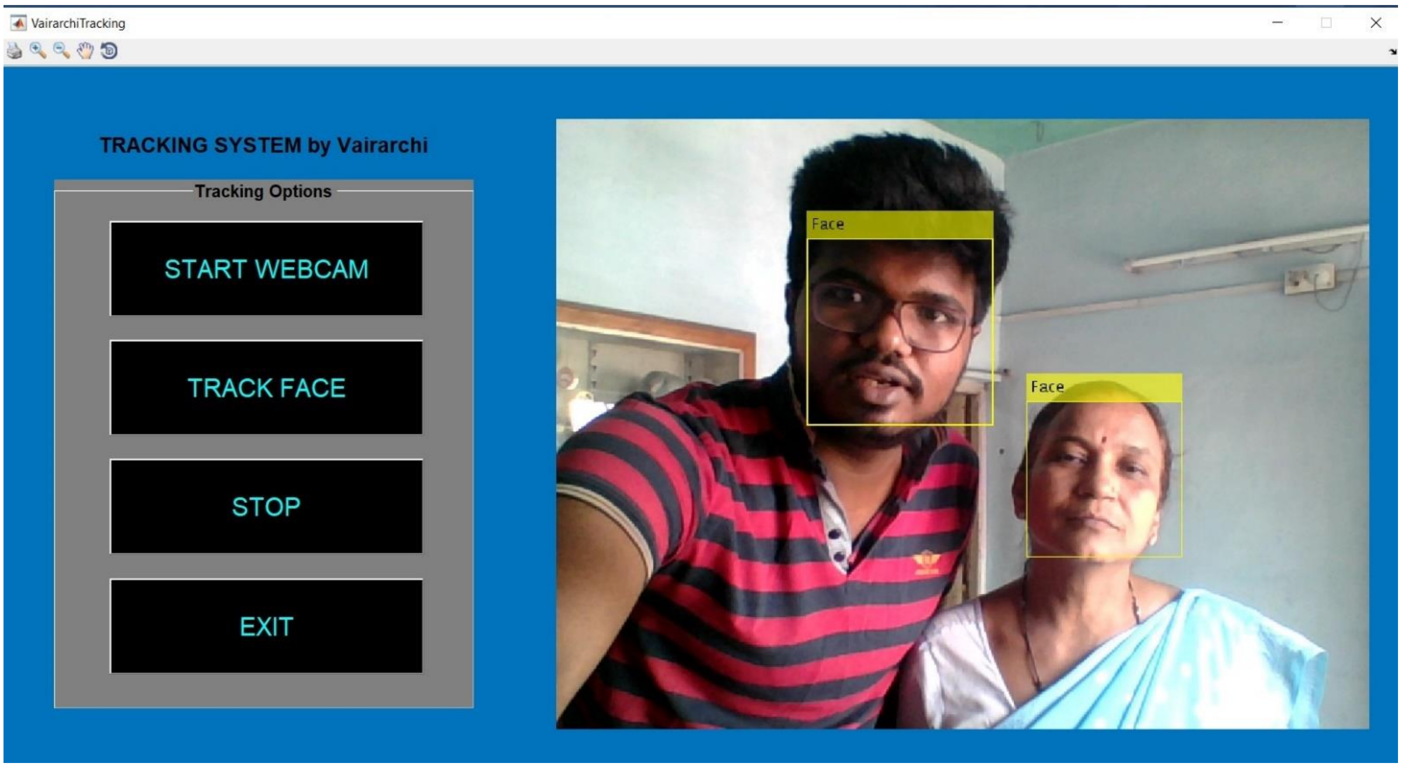


Fig. 5.6: Snapshot from second Video Input

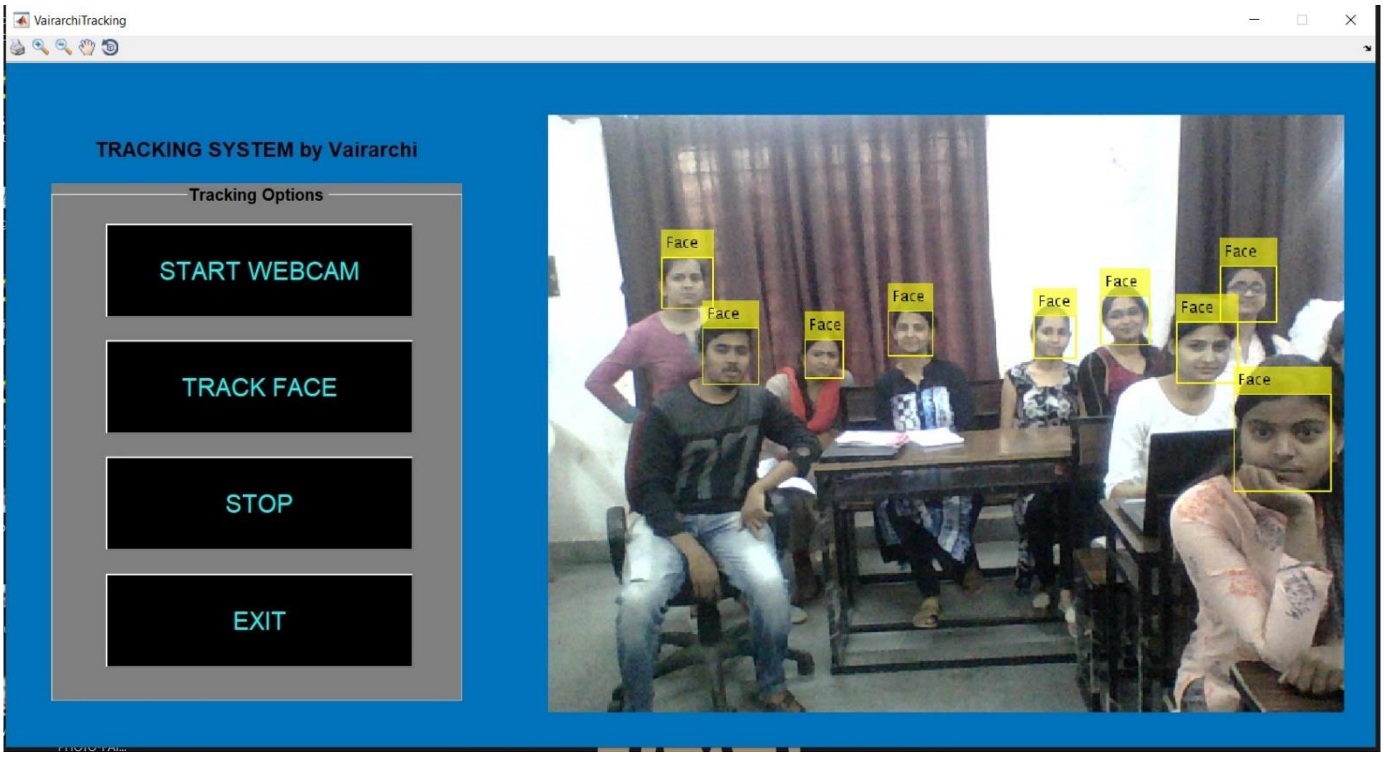


Fig. 5.7: Snapshot from third Video Input

5.3.1 Accuracy of Simulation Result

We analyse the result of our Real time simulation. The following table depicts the accuracy of the Facial Tracking system:

Snapshot	No. of Faces Detected	No. of Non-Faces Detected	No. of Faces not Detected
1.	3	0	0
2.	2	0	0
3.	9	0	0

Table 5.5 Accuracy of Simulation Result

The above table shows accuracy of simulation result. As the video moves it gets quite difficult to keep track but it does so quite efficiently. Checking it practically we can say that the accuracy is close to **100%**.

CHAPTER 6

CONCLUSION

In this project the goal of implementing a MATLAB Based Human Facial Detection System was achieved. The Viola Jones Algorithm is implemented in MATLAB. The Project is divided into two separate modules, the first one is for Image Based Face Detection and the second module is dedicated to the Real-Time Face Tracking from a live Webcam feed. The first module has four detectors namely Face, Eyes, Mouth and Nose detectors having the accuracy of 75.83%, 49.76%, 61.16% and 55.83% respectively.

Sr. No.	**Detector**	**Accuracy (in %)**
1.	Face	75.83
2.	Eyes	49.76
3.	Mouth	61.16
4.	Nose	55.83

Table 6.1 Accuracy of Detectors

The Face Detector with an overall accuracy of 75.83% is the most accurate among the lot, followed by Nose Detector, then by Mouth Detector and lastly by Eyes Detector.

The results of second module which does the Real-Time tracking of Human faces are highly accurate. Although when the video moves the tracking gets difficult and but it does so quite efficiently. Checking it practically, the simulation results show us that the accuracy is close to **100%**. The system has proved to work in real time with no lagging and under varying conditions of facial expressions, skin tones, and lighting.

Thus, we have successfully implemented the MATLAB Based Human Facial Detection System.

CHAPTER 7

FUTURE SCOPE

Face Detection is a first step towards developing a state-of-the-art Facial Recognition System for Surveillance. The general goal of any Face Detection system is to identify all regions in the input image which resembles a face regardless of the orientation, position and occlusion.

The task at hand is quite challenging and since we have successfully implemented the system, we are very well equipped to discuss the future scope of our system. The major future scope would be to develop a state of the art fully automated Face recognition system using our Face detection system as the base. Since for any Face Recognition the first part is to develop a neatly accurate Face Detection system upon which we can recognize the goal of recognizing the detected faces.

The second goal would be to recognize the facial expressions which would determine the emotions of the person. The states can include even emotional states which are neutral, joy, sadness, surprise, anger, fear and disgust based on facial expressions.

Face acquisition is a processing stage to automatically find the face region for the input images or sequences. It can be a detector to detect face for each frame or just detect face in the first frame and then track the face in the remainder of the video sequence. To handle large head motion, the head finder, head tracking, and pose estimation can be applied to a facial expression analysis system.

CHAPTER 8

REFERENCES

1. *Paul Viola and Michael Jones. 2001. Rapid Object Detection using a Boosted Cascade of Simple Features. Conference on Computer Vision and Pattern Recognition.*
2. *L. Stan and Z. Zhang. 2004. FloatBoost learning and statistical face detection. IEEE Trans. On Pattern Analysis and Machine Intelligence.*
3. *L. Zhi-fang, Y. Zhi-sheng, A.K.Jain and W. Yun-qiong, 2003, "Face Detection And Facial Feature Extraction In Color Image", Proc. The Fifth International Conference on Computational Intelligence and Multimedia Applications (ICCIMA'03), pp.126-130, Xi'an, China.*
4. *K. Seo, W. Kim, C. Oh and J. Lee, 2002, "Face Detection And Facial Feature Extraction Using Color Snake", Proc. ISIE 2002 - 2002 IEEE International Symposium on Industrial Electronics, pp.457-462, L 'Aquila, Italy.*
5. *J. Ruan and J. Yin, 2009, "Face Detection Based On Facial Features And Linear Support Vector Machines", Proc. 2009 International Conference on Communication Software and Networks, pp.371-375, Macau, China.*
6. *M. A. Berbar, H. M. Kelash and A. A. Kandeel, 2006, "Faces And Facial Features Detection In Color Images", Proc. Geometric Modeling and Imaging— New Trends (GMAI'06), pp.209-214, London, UK.*
7. *S. Kherchaoui and A. Houacine, 2010, "Face Detection Based On A Model Of The Skin Color With Constraints And Template Matching", Proc. 2010 International Conference on Machine and Web Intelligence, pp. 469 - 472, Algiers, Algeria.*
8. *Qiang-rong, Jiang, and Li Hua-lan. "Robust human face detection in complicated color images." Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on. IEEE, 2010.*
9. *Das, Akanksha, Ravi Kant Kumar, and DakshinaRanjanKisku. "Heterogeneous Face Detection." Proceedings of the International Conference on Internet of things and Cloud Computing. ACM, 2016.*
10. *M. H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting face in images: a survey," IEEE Trans. Patter Analysis and Machine Intelligence, vol. 24, pp. 34–58, 2002.*
11. *E. Hjelmas and B. K. Low, "Face detection: A survey," Computer Vision and Image Understanding, vol. 83, pp. 236–274, 2001.*
12. *Mrs. Sunita Roy et.al., "A Tutorial Review on Face Detection", International Journal of Engineering Research & Technology (IJERT), Vol. 1 Issue 8, October - 2012, ISSN: 2278-0181.*

13. *K. Sobottka and I. Pitas, "Face localization and feature extraction based on shape and color information,"Proc. IEEE Int"l Conf. Image Processing, pp. 483-486, 1996.*

14. *T. Sasaki, S. Akamatsu, and Y. Suenaga. Face image normalization based on color information. Tech. Rep. I.E.I.C.E., IE91-2, pp. 9–15 (1991).*

15. *C. Kotropoulos and I. Pitas, "Rule-based face detection in frontal views," Proc. Int"l Conf. Acoustics, Speech and Signal Processing, vol. 4, pp. 2537-2540, 1997.*

16. *C. Lin, K.C. Fan, "Human face detection using geometric triangle relationship," Proc. 15th ICPR, pp. 945–948, 2000.*

17. *E. Hjelmas and B. K. Low, "Face detection: A survey," Computer Vision and Image Understanding, vol. 83, pp. 236–274, 2001.*

18. *Alan L. Yuille, "Deformable Templates for Face Recognition", Journal of Cognitive Neuroscience Volume 3, pp. 59-70,Number 1991.*

19. *D. G. Lowe, "Distinctive image features from scale-invariant keypoints," Int"l Journal of Computer Vision, vol. 60, no. 2, pp. 91-110, 2004.*

20. *H.A. Rowley, " Neural Network-Based Face Detection", PhD thesis, Carnegie Mellon Univ, 1999.*

21. *Craw, I., Tock, D. & Bennett, A. (1992), Finding Face Features, in „European Conference on Computer Vision", pp. 92–96.*

22. *A. K. Jain, R. P. W. Duin, and J. C. Mao, "Statistical pattern recognition: a review," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 22, no. 1, pp. 4–37, 2000.*

23. *K. J. Liao, Face detection by outline, color, and facial features, Mater thesis, GICE, NTU, Taipei, 2010.*

24. *D. Cristinacce, T. Cootes, "Facial Feature Detection Using AdaBoost With Shape Constrains", British Machine Vision Conference, 2003.*

25. *Adolf, F. How-to build a cascade of boosted classifiers based on Haar-like features. http://robotik.inflomatik.info/other/opencv/OpenCV_ObjectDetection_HowTo.pdf, June 20 2003.*

26. *S. J. McKenna, S. Gong, R. P. Wurtz, J. Tanner, D. Banin "Tracking Facial Feature Points with Gabor Wavelets and Shape Models" 1st Int. Conf. on Audio*-and *Videobased Biometric Person Authentication, Lecture Notes in Computer Science 1997.*