HAND AND EYE GESTURE RECOGNITION

GROUP 24

<u>GROUP MEMBERS:</u>

SACHIN SHARMA: 20161060
AASHISH KUMAR: 20161111

Github link - “https://github.com/aashtrek/DIP_PROJECT”

AIM:

Real time hand and eye gesture recognition using image processing only.

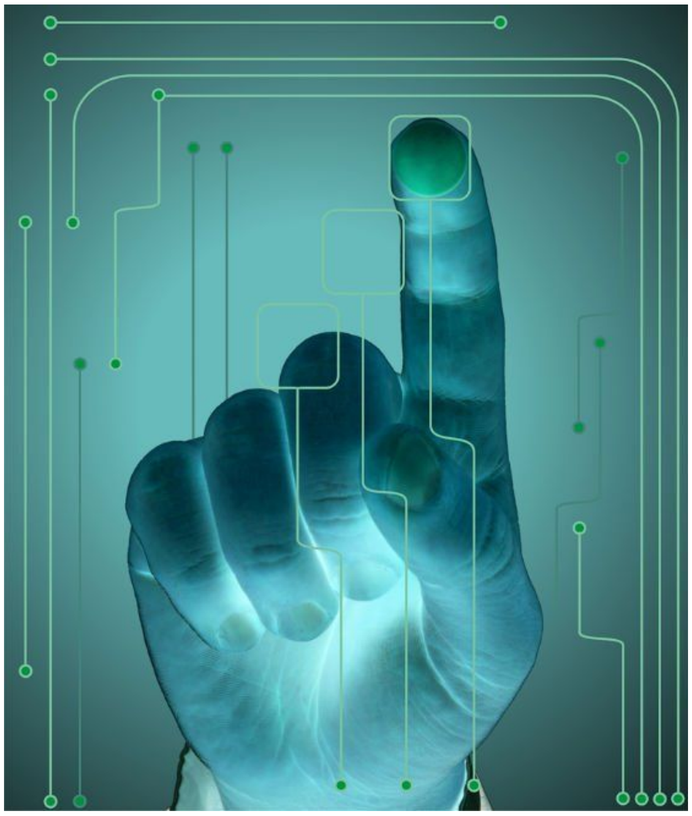
ABSTRACT:

Hand and eye gesture recognition provides natural ,fast,innovative non verbal way of communication. It has a wide area of application in computer vision,gaming and other areas.

We implemented hand and eye gesture recognition separately. In hand gesture recognition we used used skin recognition and segmentation to detect hand in real time, and then finger detection and counting to detect the gesture .

In Eye gesture recognition, we used cascade object detector function of matlab which uses viola-jones algorithm to detect eye from face,and then track the position and movement of eyeball to detect gesture

HAND GESTURE RECOGNITION



Introduction

Hand gesture recognition is a topic in computer science and technology with the goal of interpreting hand gestures via mathematical algorithms. Users can use simple hand gestures to control or interact with devices without physically touching them.

Problem Statement:

The goal of this part of project is to design and implement an algorithm which detects gestures of hand in real time and calculate the number of fingers in a gesture. The main challenge here is to detect all the gestures quickly and efficiently.

Motivation:

There are two motivations for selecting this project:

- Primarily motivation of hand gestures recognition is that it can easily be implemented by machine learning techniques but implementing with image processing is a tough and interesting task.
- It is the demand of advanced technology to recognize, classify and interpret various simple hand gestures and use them in wide range of application through computer vision.

Overview:

The basic steps in hand gesture recognition are as follows:

- **Input data** - Taking a recorded video of different hand gestures as input for algorithm. Recorded video has only vertical hand gestures.
- **Hand detection** - Detect the hand gestures of input data by the segmentation based on YCbCr.

- **Feature extraction** - Count the number of fingers in hand gesture.
- **Mapping** - Map the hand gestures on the basis of count of fingers in images and assigning a particular value to a particular gesture.

Constraints:

- Input images required a plain background because hand detection step is very sensitive of background colour.
- We implement the algorithm for single hand gestures.

PROCEDURE

- **Input data**:
    - Recorded a video having different hand gestures and plain background. Colour of background should not be matched to the colour of hand.
    - Get the frames of recorded video by using VideoReader function and read the frame by using read function.

- **Hand detection** :
    - Frames getting in step 2 is passed from the function name grayworld which compensates illumination.

- Steps in grayworld function:
    - Inverse the average values of R,G,B of input image
    - Find the smallest average value
    - Calculate the scaling factor
    - Scale the values of R,G,B
- Convert the colour space of image from rgb to yCbCr
- Cb and Cr values for skin is lies in range of (77, 133) and (133, 173) respectively. So if the value of pixels are lying in above range then we assign 1 (white) to that pixel otherwise 0 (black).
- Use bwareaopen and imfill functions to make hand gesture more enhanced.

- **Feature extraction** :
    - Finding the orientation of image
        - If column 1 of image contains white pixel then the image is vertical
        - If last row of image having white pixels then image is horizontal
        - Else both column1 and last row of image, both having white pixels then we will not show any output.
    - Locate the bounding box by using the function regionprops.

    - Calculate the centroid (Cx, Cy) of image by using the image moment formula.
    - Calculating the radius by taking the difference between Cx and upper bound of box.
    - Take take radius = 0.7 * (Cx - upper bound)
    - Now take centroid as centre and calculate the pixels such that:
        - theta = theta + pi/36 and theta < pi
        - x=Cy + R*Cos(theta) ; y=Cx - R*Sin(theta)
        - If img(y, x) changes from 0 to 1 then increase the count by 1.
- **Mapping :**
    - Map the value of count to characters (like 1 , 2, 3... or a ,b, c...)
- Update frame = frame + 50

Other approaches considered and their drawbacks

- We tried to use laptop webcam for input data but did not get the hand gestures after hand gesture detection step.

Image by laptop webcam

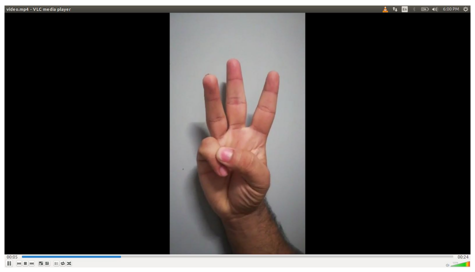


Image by mobile camera

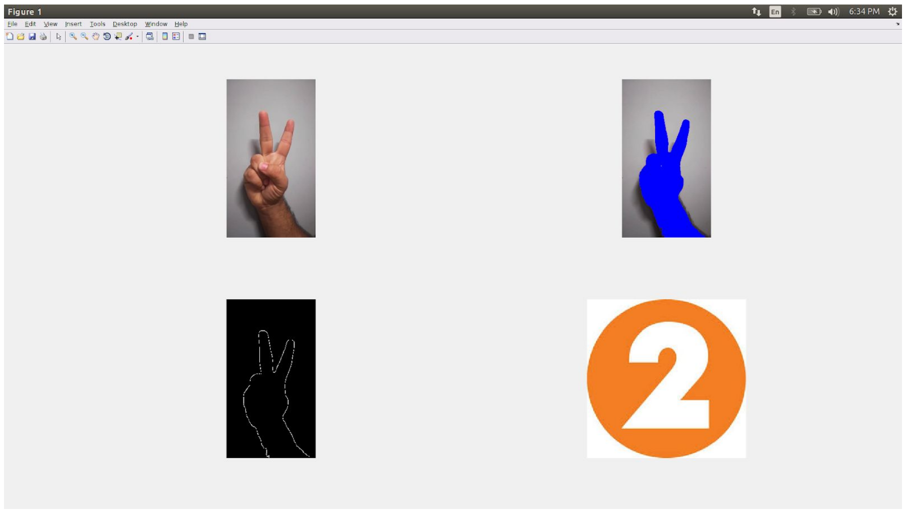
- In hand gesture detection step we tried use the laptop webcam and store a image(I1) initially and then take snapshot (i2)after every pause(2). (i2 having hand gestures)
    - Then
        - if abs(i2(x, y) - i1(x, y)) < 10 then i(x,y) =0
        - Else i(x,y) = 1
    - After this we use function imfill holes and bwareaopen function on image but did not get the good image.
- For feature extraction process, we create edge image of binarize hand image and then try to find out the pixel coordinates of local maxima but got more than 5 maximas due to rough boundary of image. Then we did not figure out the correct maxiamas.

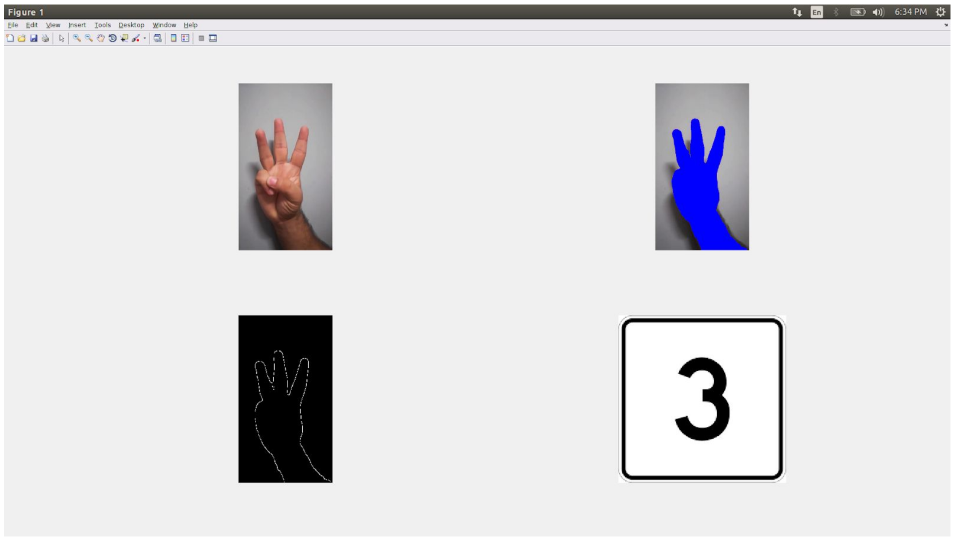
Results

subplot(2, 2,1) : hand gesture	subplot(2,2,2) : hand detection region
subplot(2, 2, 3) : edge of hand	subplot(2, 2, 4) : Gesture output

Vertical mapping	Horizontal mapping
1 ---> 1	1 --------> A
2 ----> 2	2 ---------> B
3 -----> 3	3 --------> C
4 ------> 4	4 --------> D
5 -----> 5	5 --------> E

Vertical hand gesture





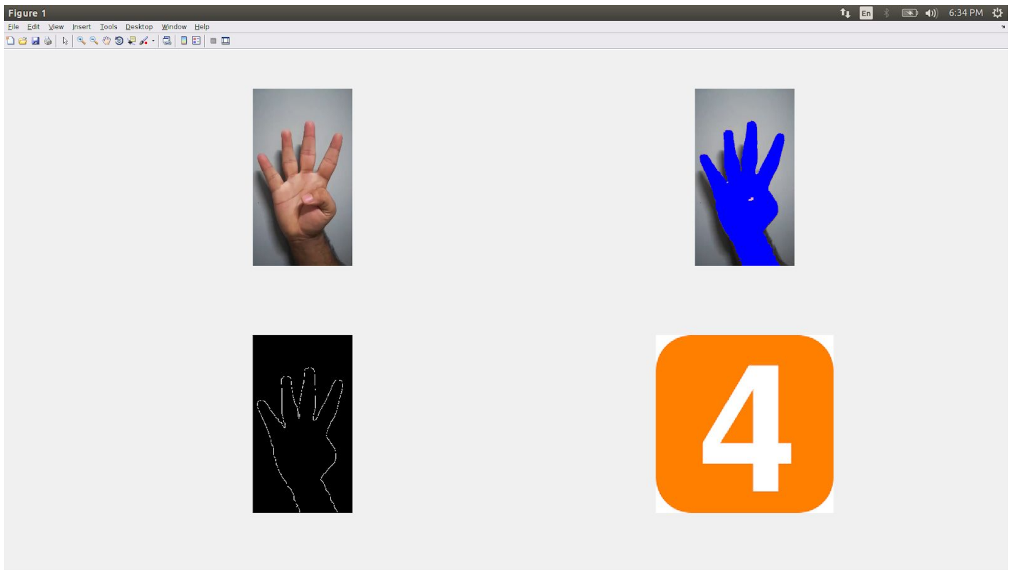


Figure 1
File Edit View Insert Tools Desktop Window Help
6:34 PM
4

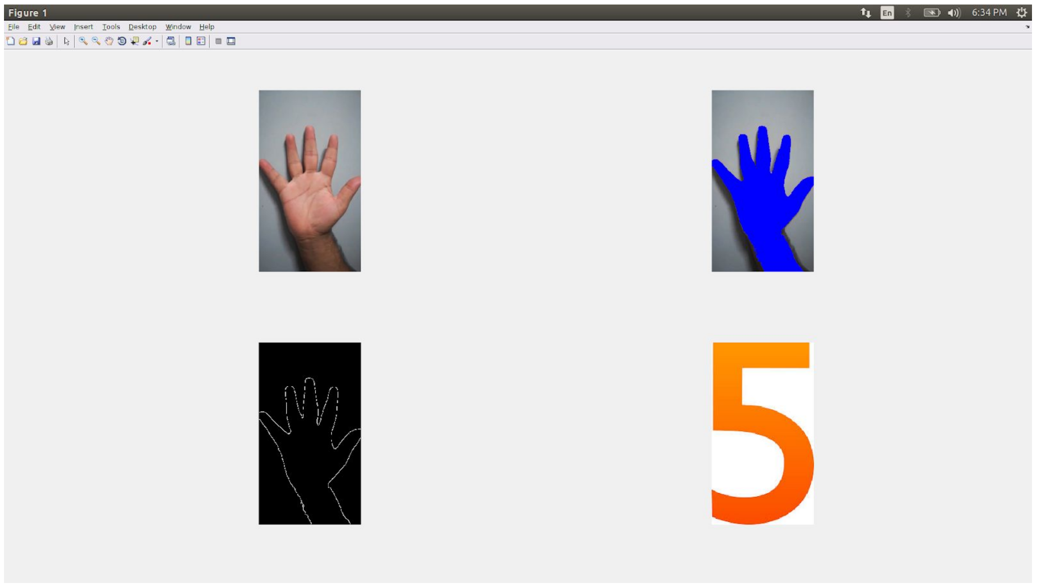
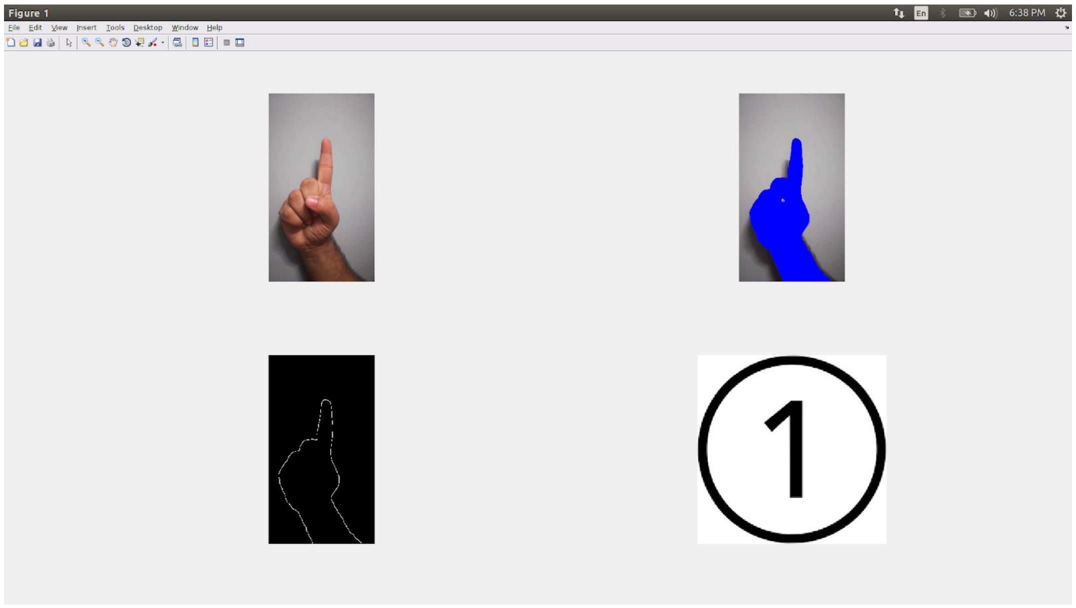


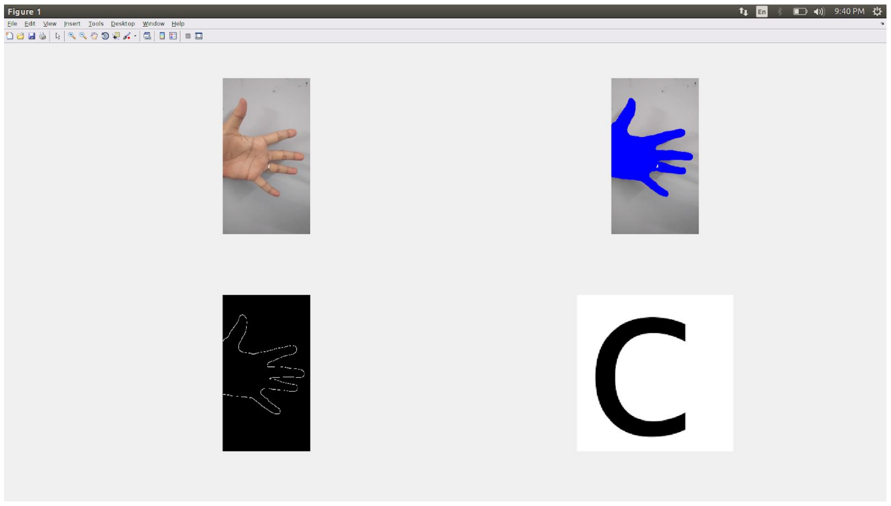
Figure 1
File Edit View Insert Tools Desktop Window Help
6:34 PM
5



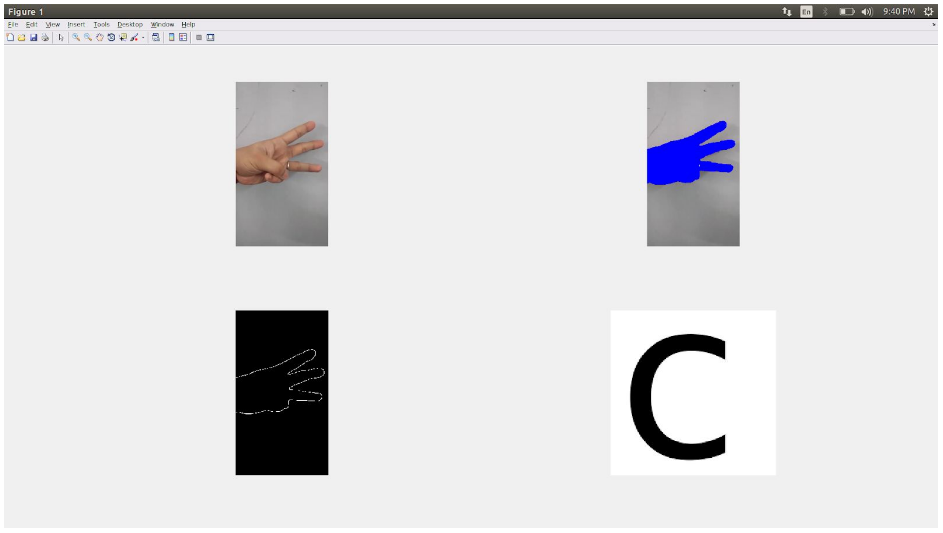
Wrong test:

At the time of report submission only vertical images are giving correct result. While some cases of horizontal hand gestures are giving correct answer and some are given wrong answers. So we are trying to fix this before the presentation.

Wrong matching



Correct matching



EYE GESTURE RECOGNITION

Introduction

Problem Statement:

The goal of the project is to design and implement an algorithm which detects gestures of eye in real time. The main challenge here is to detect all the gestures quickly and efficiently.

Motivation:

With advancement of technology people are shifting towards more smaller and portable devices. And to do a small task people wants lesser and lesser effort.Also implementing this without using machine learning is the real charm of this project.

Constraints:

- Does not detects movement of an eyeball in up-down direction.
- Does not detect blinking.

Overview:

The basic steps followed are as follow:

- **Face Detection:** We detected the face using cascade object detector in matlab which uses *Viola-Jones algorithm* to detect face and its components(eye ,nose etc).
- **Pupil Detection:** After successfully detecting the eye, find the circle in eye region using imcircle function in matlab which uses *Hough Transform to detect circles.*
- **Gesture Recognition:** Now checking distance of centre of eye ball from ends of edge of eye to find left,right and centre.

PROCEDURE

Here are the steps to detect eye gesture from start:

- Create webcam object to get the real time live input.
- Create detector for face and eye pair vision cascade object detector to detect eye pair and face using Viola-Jones algorithm..
- Take snapshot and create bounding box of face and eye pair using detector.
- If face exist the extract the face and find the biggest face.
- If eyepair available find the biggest eye pair available.
- Find the circle in eye pair region using hough transform.
- Check distance of centre of pupil from edges of eye. And detect left,right,straight position.

Voila-Jones Algorithm

The Viola-Jones algorithm is a widely used mechanism for object detection. The main property of this algorithm is that training is slow, but detection is fast. This algorithm uses Haar basis feature filters, so it does not use multiplications.

$$II(y,x) = \sum_{p=0}^{y} \square \sum_{q=0}^{x} Y(p,q)$$

The efficiency of the Viola-Jones algorithm can be significantly increased by first generating the integral image

The integral image allows integrals for the Haar extractors to be calculated by adding only four numbers. For example, the image integral of area ABCD (Fig.1) is calculated as $II(y_A,x_A) – II(y_B,x_B) – II(y_C,x_C) + II(y_D,x_D)$.

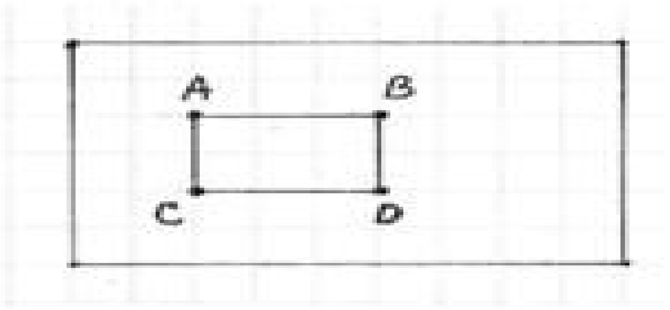


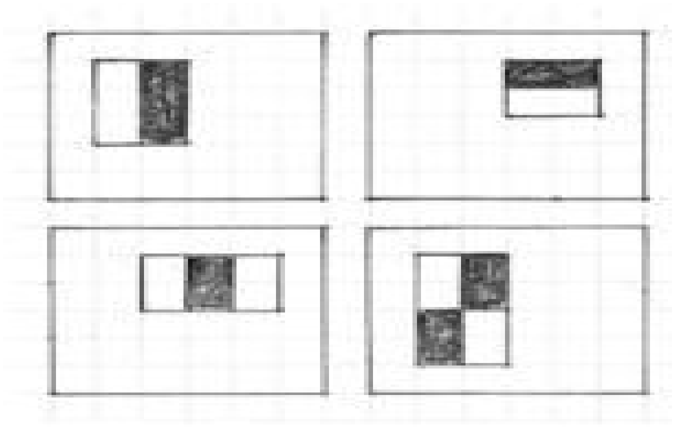
Fig.1 Image area integration using integral image

Detection happens inside a detection window. A minimum and maximum window size is chosen, and for each size a sliding step size is chosen. Then the detection window is moved across the image as follows:

1. Set the minimum window size, and sliding step corresponding to that size.
2. For the chosen window size, slide the window vertically and horizontally with the same step. At each step, a set of *N* face recognition filters is applied. If one filter gives a positive answer, the face is detected in the current widow.
3. If the size of the window is the maximum size stop the procedure. Otherwise increase the size of the window and corresponding sliding step to the next chosen size and go to the step 2.

Each face recognition filter (from the set of *N* filters) contains a set of cascade-connected classifiers. Each classifier looks at a rectangular subset of the detection window and determines if it looks like a face. If it does, the next classifier is applied. If all classifiers give a positive answer, the filter gives a positive answer and the face is recognized. Otherwise the next filter in the set of *N* filters is run.

Each classifier is composed of Haar feature extractors (weak classifiers). Each Haar feature is the weighted sum of 2-D integrals of small rectangular areas attached to each other. The weights may take values ±1.



The classifier decision is defined as:

$$C_m = \begin{cases} 1, & \sum_{i=0}^{I_m-1} F_{m,i} > \theta_m \\ 0, & \text{otherwise} \end{cases}$$

$$F_{m,i} = \begin{cases} \alpha_{m,i}, & \text{if } f_{m,i} > t_{m,i} \\ \beta_{m,i}, & \text{otherwise} \end{cases}$$

$f_{m,i}$ is the weighted sum of the 2-D integrals. is the decision threshold for the *i*-th feature extractor. $\alpha_{m,i}$ and $\beta_{m,i}$ are constant values associated with the *i*-th feature extractor. θ_m is the decision threshold for the *m*-th classifier.

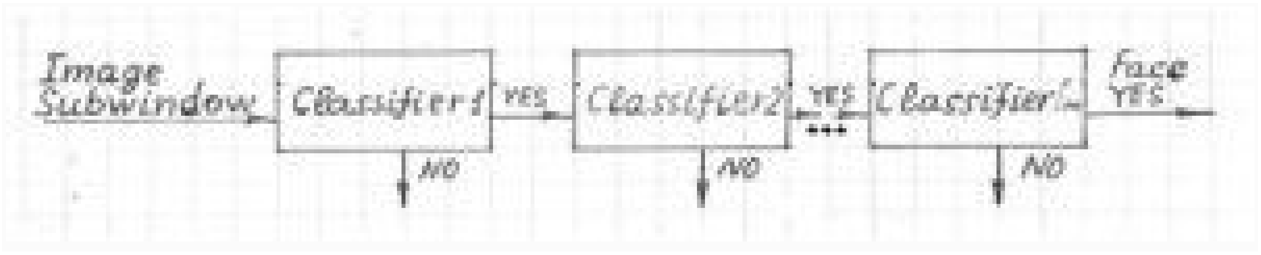


Fig.3 Object detection Viola-Jones filter

HOUGH TRANSFORM

A circle can be described completely with three pieces of information: the center (a, b) and the radius. (The center consists of two parts, hence a total of three)

$x = a + R\cos\theta$

$y = b + R\sin\theta$

When the θ varies from 0 to 360, a complete circle of radius R is generated.

So with the Circle Hough Transform, we expect to find triplets of (x, y, R) that are highly probably circles in the image. That is, we want to find three parameters.

To begin, we'll start with the assumption that you're looking for circles of a particular radius, that is, R is known. The equation of each circle is:

$x = a + R\cos\theta$

$y = b + R\sin\theta$

So, every point in the xy space will be equivalent to a circle in the ab space (R isn't a parameter, we already know it). This is because on rearranging the equations, we get:

$a = x1 - R\cos\theta$

$b = y1 - R\sin\theta$

for a particular point (x1, y1). And θ sweeps from 0 to 360 degrees.

So, the flow of events is something like this:

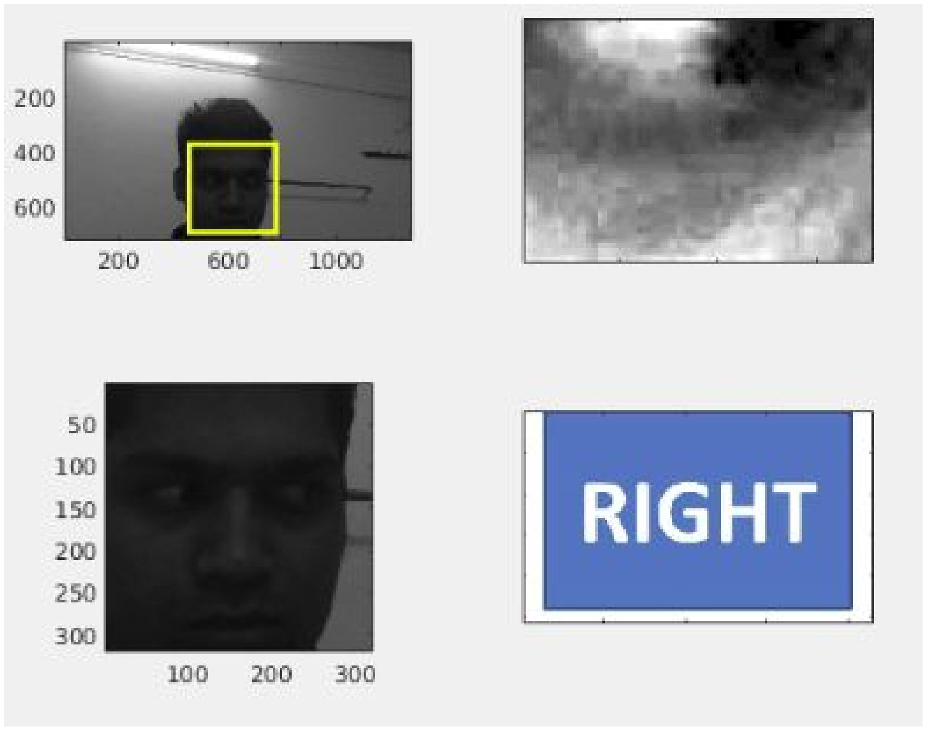
1. Load an image
2. Detect edges and generate a binary image
3. For every 'edge' pixel, generate a circle in the ab space
4. For every point on the circle in the ab space, cast 'votes' in the accumulator cells
5. The cells with greater number of votes are the centers

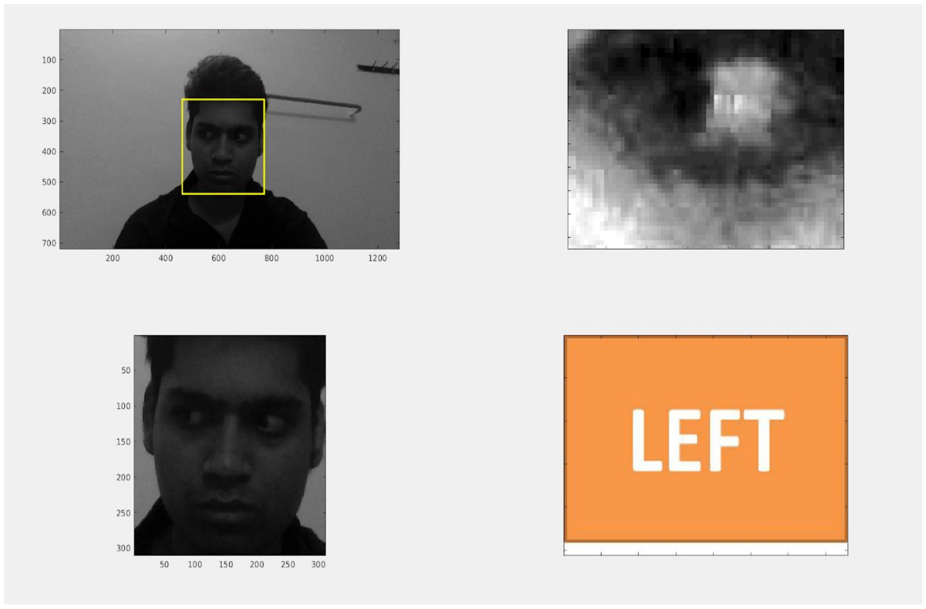
APPROACH CONSIDERED

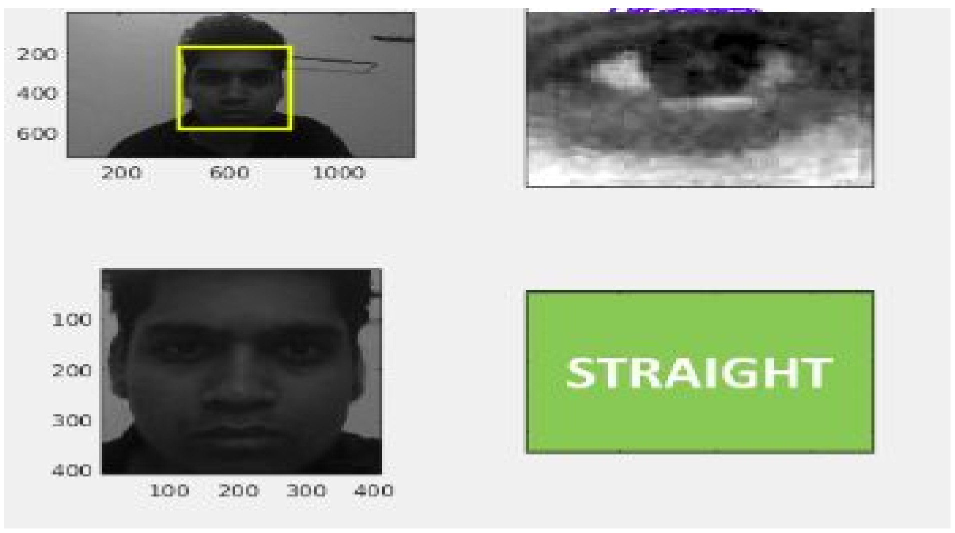
Detect skin. Then find the biggest blob .And consider that as face.Then predict the location of eye. As the relative position is almost same for every person.Then find pupil location and detect gesture.

This will have very poor accuracy. As detection of eye will not be accurate .And since pupil is very small .Then Even a slight change will give false output.

RESULT


200
400
600
200
600
1000
50
100
150
200
250
300
100
200
300
RIGHT


LEFT


200
400
600
200
600
1000
100
200
300
400
100
200
300
400
STRAIGHT