PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

100-ft Ring Road, Bengaluru – 560 085, Karnataka, India

Report on

‘EXPLORING HUMAN COMPUTER INTERACTION USING OPEN COMPUTER VISION (IMAGE PROCESSING)’

Submitted by

SHRILAXMI BHAT(PES1201800419)
AHAN BOSE(PES1201800467)
PRERANA RAMACHANDRA(PES1201800491)

JAN - MAY 2020

under the guidance of

Rajini M

Asst. Professor

Department of ECE

PES University

Bengaluru -560085

CONTENTS

CHAPTER: 01

ABTRACT:

Every system despite being powerful and complex requires man to be near it or somehow in physical contact with it for it to work efficiently. Computer vision has made it possible to make this human-computer interaction hands free along with providing equally efficient results. This project focuses on four such basic models that have been implemented to emphasize on the usage of image processing using well-known algorithms and classifiers. If perfected and used correctly, human computer interaction can be successfully used for wide applications.

CHAPTER: 02

INTRODUCTION:

A computer vision system uses the image processing algorithms to try and perform emulation of vision at human scale. Most well-known algorithms can be devised by conveniently applying the working principles of their described methods and models to the specific requirements of each application. Four such basic models have been implemented to emphasize on the usage of image processing using well-known algorithms and classifiers.

Gesture recognition has been used to create a system which can identify specific human gestures and use them for device control or to convey information.

Tracking the movement of the eye using OpenCV and Python implemented for various hands-free applications.

Haar Feature-based Cascade Classifiers has been used to detect faces and eyes.
It identifies human faces and eyes given a video stream of pictures.

Histogram of Oriented Gradients (HOG) is an efficient way to extract features out of the pixel colours for building an object recognition classifier which has been used for body detection.

CHAPTER: 03

MODEL 1: GESTURE-RECOGNITION BASED CONTROL

3.1. INTRODUCTION

In this section we explore human computer interaction under the heading of gesture-recognition based control.

Gesture recognition is a topic in computer science and language technology with the goal of interpreting human gestures via mathematical algorithms.

It can be seen as a way for computers to begin to understand human body language, thus building a richer bridge between machines and humans than primitive text user interfaces or even GUIs (graphical user interfaces), which still limit the majority of input to keyboard and mouse and interact naturally without any mechanical devices.

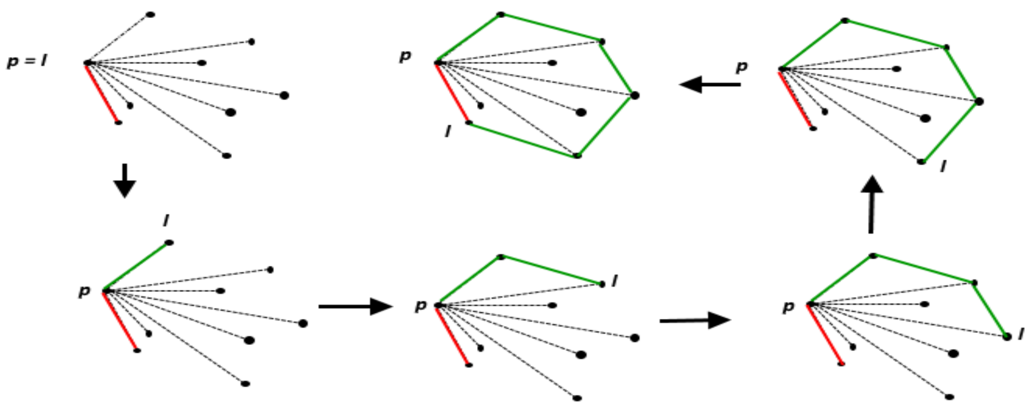
Many approaches have been made using cameras and computer vision algorithms to interpret sign language. Gesture recognition can also be used to create a system which can identify specific human gestures and use them for device control or to convey information.

Here we use the concept of convex hull and convexity defect to extract features of the gesture to recognize it and give a real time output as response.

3.2. REVIEW OF LITERATURE

3.2.1. JARVIS MARCH ALGORITHM & WRAPPING ALGORITHM

The idea of Jarvis's Algorithm is simple, we start from the leftmost point (or point with minimum x coordinate value) and we keep wrapping points in counter-clockwise direction.



The execution of jarvis's March

3.2.2. RGB HSV COLOUR SPACES

This is done to filter the skin coloured pixels from the background.

RGB Colour Model :

The RGB colour model is an additive colour model in which red, green and blue light are added together in various ways to reproduce a broad array of colours. The name of the model comes from the initials of the three additive primary colours, red, green, and blue.

HSV Colour Model :

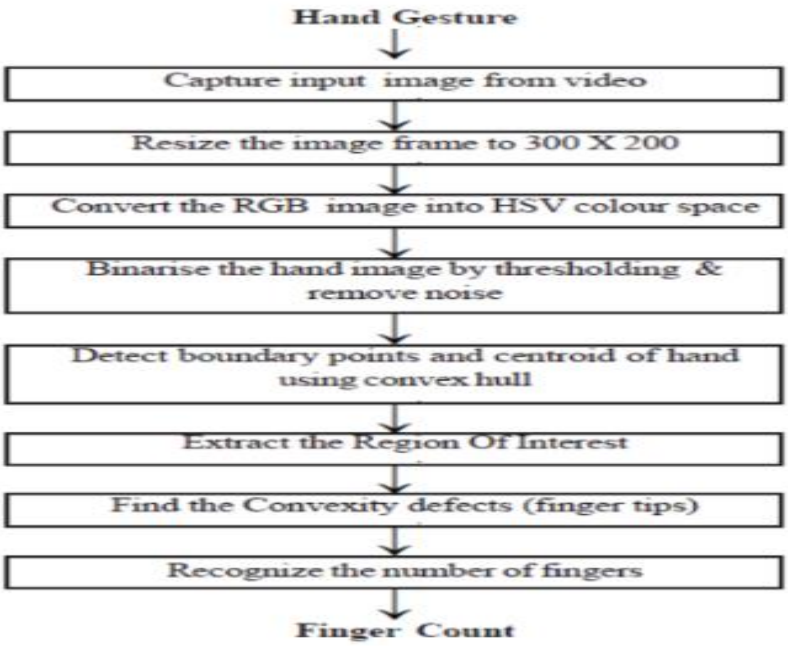
HSV – (hue, saturation, value), also known as HSB (hue, saturation, brightness), is often used by artists because it is often more natural to think about a colour in terms of hue and saturation than in terms of additive or subtractive colour components. HSV is a transformation of an RGB colour space, and its components and colorimetry are relative to the RGB colour space from which it was derived.

3.2.3. Image thresholding & binarization

Image binarization is the process of separation of pixel values into two groups, white as background pixel and black as foreground or object pixel.

3.3. METHODOLOGY

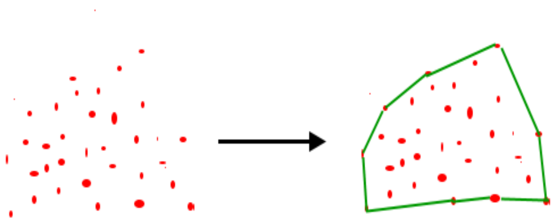
3.3.1. BLOCK DIAGRAM



3.3.2. CONCEPT OF CONVEX HULL

The method is all about constructing, circumscribing, or encompassing a given set of points in plane, here the gesture, by a polygonal capsule called convex polygon. The convex hull of the set is the smallest convex polygon that contains all the points of it to capture the gesture accordingly.

Given a set of points in the plane. the convex hull of the set is the smallest convex polygon that contains all the points of it. Using convex hull we determine the outer edge of our hand i.e. ‘contour’



3.3.3. CONTOUR DETECTION

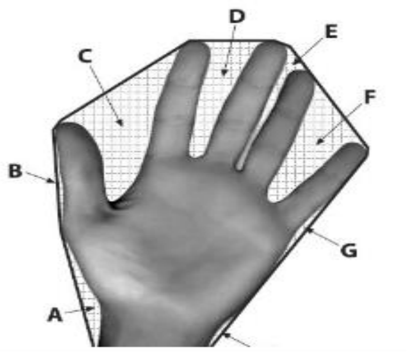
A contour by definition is an outline of an object or a body usually defined by a line. In computer vision the contour of an object can be used to compute the convex hull of the object from which the convexity defects can be computed. After the skin detection and noise filtering algorithms have been applied, an outline of the hand needs to be generated by applying the contour detection algorithm in OpenCV.

This is achieved by applying the cvDrawContours() function in OpenCV.

3.3.4. CONVEX HULL & CONVEXITY DEFECT

In order to determine the shape of objects or a contour that has already been derived, the objects convex hull and convexity defects can be computed. The figure below shows the outline of the human hand.

The convex hull of the image is shown as the dark outline, and the defects which are relative to that outline are labelled A through H. These defects can be used to distinguish the condition of the hand.



3.3.5. FINGER COUNTING

In order to get the count of the fingers, the convexity defects described previously are used.
Four major points are required from the convexity defects as shown in Figure.
They are the start point SX, the Depth Point DY, the Box center point BY, and the length of the defect LD.
For each convexity defect CD, the following algorithm [10] is applied to get the total finger count. Count = {1}If (SY < BY or DY < BY)and (SY < DY) and (LD > box height/ n)Where, LD = (SX – DX)2 + (SY – DY)2 Else Count = {0}. Number of Fingers counted =Σ Count.

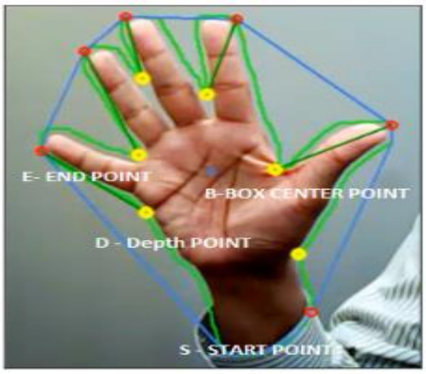
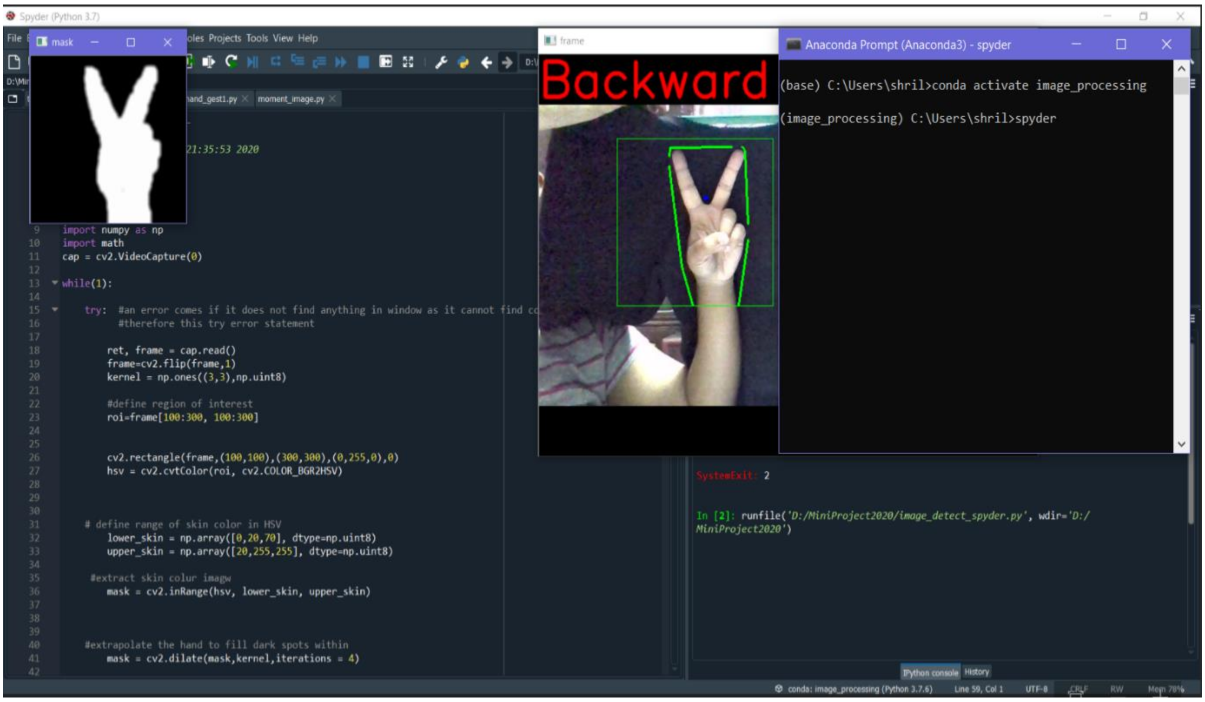


Figure 4: Points used to compute finger count

3.4 RESULT


Spyder (Python 3.7)
mask
frame
Backward
Anaconda Prompt (Anaconda3) - spyder
(base) C:\Users\shril>conda activate image_processing
(image_processing) C:\Users\shril>spyder
import numpy as np
import math
cap = cv2.VideoCapture(0)
while(1):
try: #an error comes if it does not find anything in window as it cannot find co
#therefore this try error statement
ret, frame = cap.read()
frame=cv2.flip(frame,1)
kernel = np.ones((3,3),np.uint8)
#define region of interest
roi=frame[100:300, 100:300]
cv2.rectangle(frame,(100,100),(300,300),(0,255,0),0)
hsv = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
# define range of skin color in HSV
lower_skin = np.array([0,20,70], dtype=np.uint8)
upper_skin = np.array([20,255,255], dtype=np.uint8)
#extract skin colur imagw
mask = cv2.inRange(hsv, lower_skin, upper_skin)
#extrapolate the hand to fill dark spots within
mask = cv2.dilate(mask,kernel,iterations = 4)
SystemExit: 2
In [2]: runfile('D:/MiniProject2020/image_detect_spyder.py', wdir='D:/MiniProject2020')
IPython console
History

CHAPTER: 04

MODEL 2: EYE MOTION TRACKING

4.1. INTRODUCTION

An eye tracker is used for measuring eye positions and eye movement. Eye trackers are used in research on the visual system, in psychology, in psycholinguistics, marketing, as an input device for human-computer interaction. There are a number of methods for measuring eye movement. The most popular variant uses video images from which the eye position is extracted.

4.2. REVIEW OF LITERATURE

4.2.1. UNDERSTANDING THE EYE

Before getting into details about image processing, let's know a bit about the eye.

The eye is composed of three main parts:

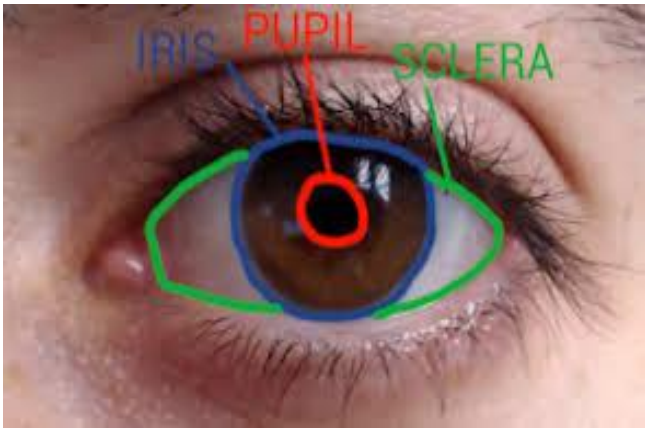
1. Pupil – the black circle in the middle

2. Iris – the bigger circle that can have different color for different people

3. Sclera – it's always white

4.2.2. SELECTION OF ROI (REGION OF INTEREST)

In this way we are restricting the detection only to the pupil, iris and sclera and cutting out all the unnecessary things like eyelashes and the area surrounding the eye, hence the small dimension.



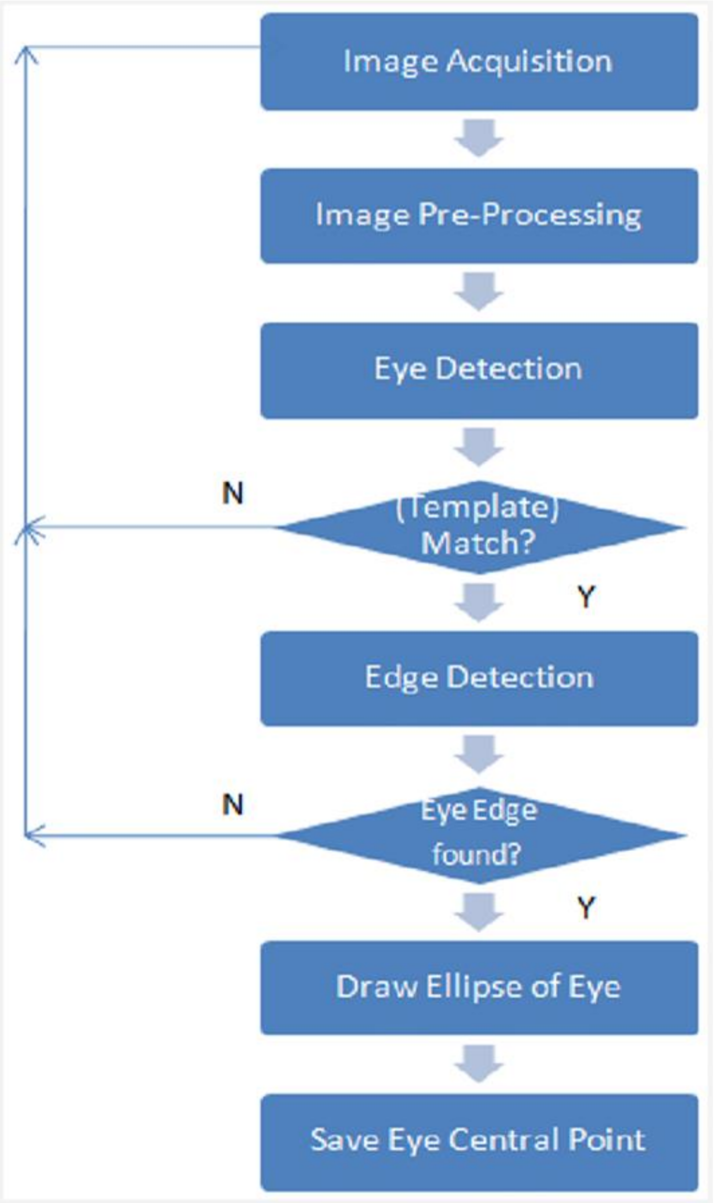
4.2.3. CONTOUR DETECTION

A contour by definition is an outline of an object or a body usually defined by a line. Here the contour of the pupil can be used to obtain the position of the eye ball. It is generated by applying the contour detection algorithm in OpenCV.

This is achieved by applying the cvDrawContours() function in OpenCV.

4.3. METHODOLOGY

4.3.1. BLOCK DIAGRAM



4.3.2. IMPLEMENTATION

Eye-trackers necessarily measure the rotation of the eye with respect to some frame of reference.

Finding the right approach for the detection of the motion:

- Considering all possible directions that the eye can have (center, left, right) we need to find the common and uncommon elements between them all so that we can use it to our advantage in segregating them.
- For the detection we could use different approaches, focusing on the sclera, the iris or the pupil.
  We're going for the easiest approach possible, and probably the best solution anyway.
- Yes, we will simply focus on the pupil.

Note:

We may either import the video where the eye is moving or track the eye movement real time.

Here to implement this model we have used the latter.

But if we choose to upload a video, we will have to load the video and then we put it in a loop so that we can loop through the frames of the video and process image by image.

4.3.3. DETECTING MOTION

By converting the image into grayscale format we will see that the pupil is always darker then the rest of the eye.

No matter where the eye is looking at and no matter what color is the sclera of the person.

i) Conversion to grayscale

ii) Find the threshold to extract only the pupil

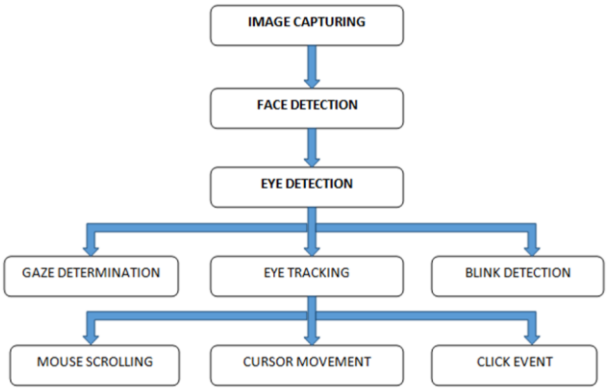
iii) From the threshold we find the contours (edge detection)

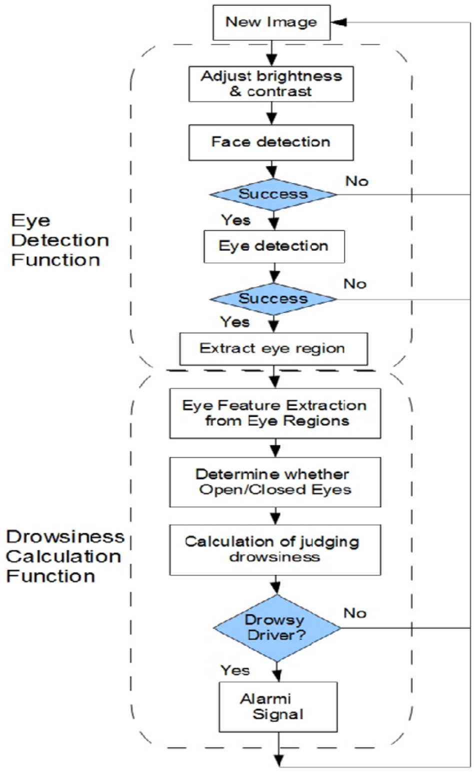
iv) We remove or process the data by means of selecting the element with the biggest area (which is supposed to be the pupil).

v) The display shows us the location of the eye hence tracking its movement

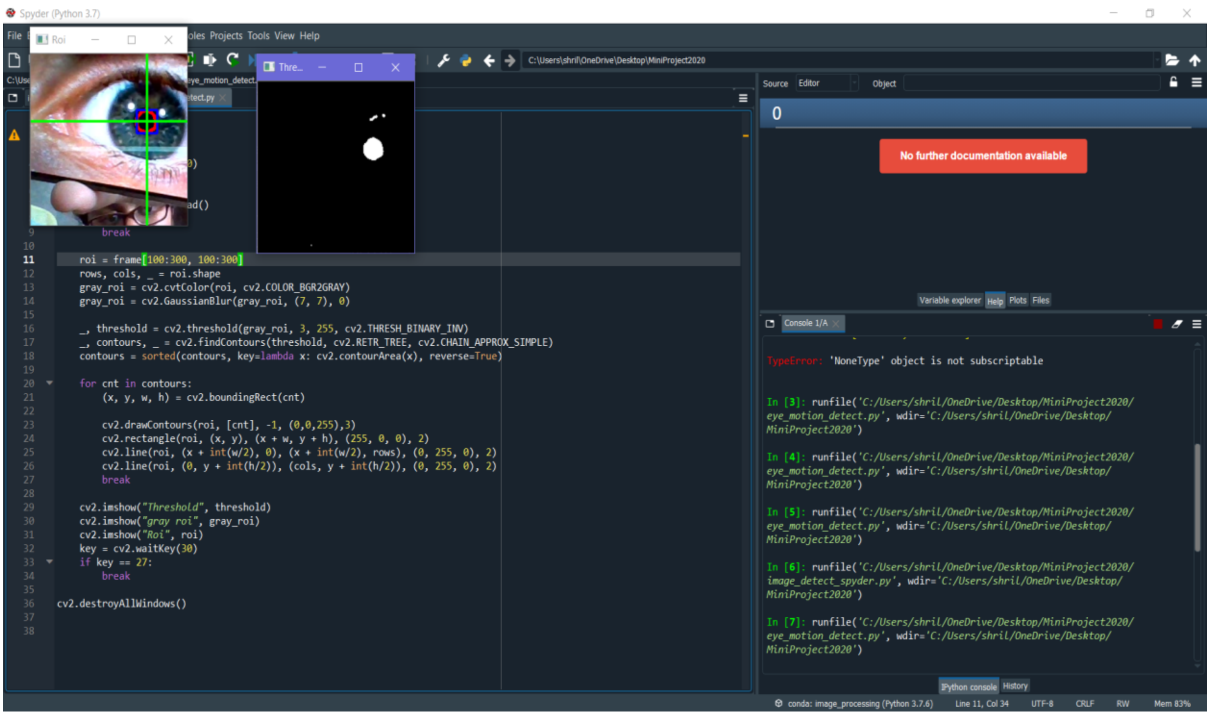
4.3.4. APPLICATIONS

Eye trackers are also being increasingly used for rehabilitative and assistive applications (related for instance to control of wheel chairs, robotic arms and prostheses).





4.4. RESULT



CHAPTER: 05

MODEL 3: FACE & EYE DETECTION

5.1. INTRODUCTION

Computer vision is a field of study which encompasses on how computer see and understand digital images and videos. One of the many applications of computer vision is face and eye detection. The precise face and eyes detection are crucial in many Human-Machine Interface system. As precise eyes location enables accurate alignment, one must first design an efficient face and eye localization method to develop an automatic face recognition system. High detection ratio obtained with those computationally efficient Haar Cascade Classifiers (HCC) suggests the possibility of using them in a reliable real-time HMI systems.

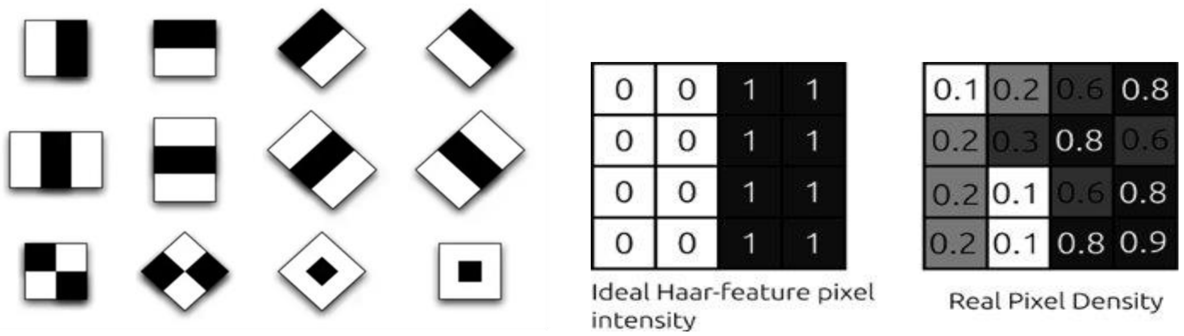
5.2. REVIEW OF LITERATURE

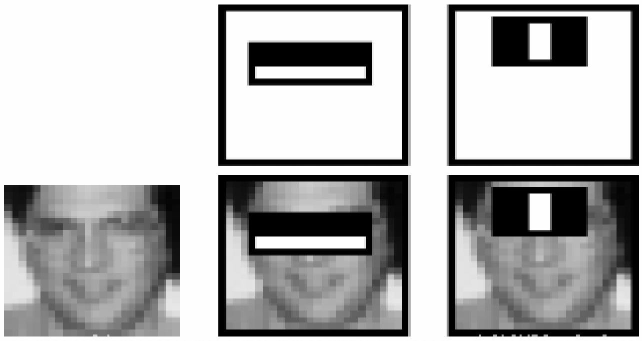
5.2.1. VIOLA-JONES ALGORITHM

Developed in 2001 by Paul Viola and Michael Jones, the Viola-Jones algorithm is an object-recognition framework that allows the detection of image features in real-time. Viola-Jones was designed for frontal faces, so it is able to detect frontal the best rather than faces looking sideways, upwards or downwards. Before detecting a face, the image is converted into grayscale, since it is easier to work with and there's lesser data to process. The Viola-Jones algorithm first detects the face on the grayscale image and then finds the location on the coloured image.

5.2.2. HAAR-LIKE FEATURES

A Haar-like feature is represented by taking a rectangular part of an image and dividing that rectangle into multiple parts. They are often visualized as black and white adjacent rectangles. They are basically convolutional kernels. Haar-like features are organized into something called classifier cascade to form a strong learner or classifier.





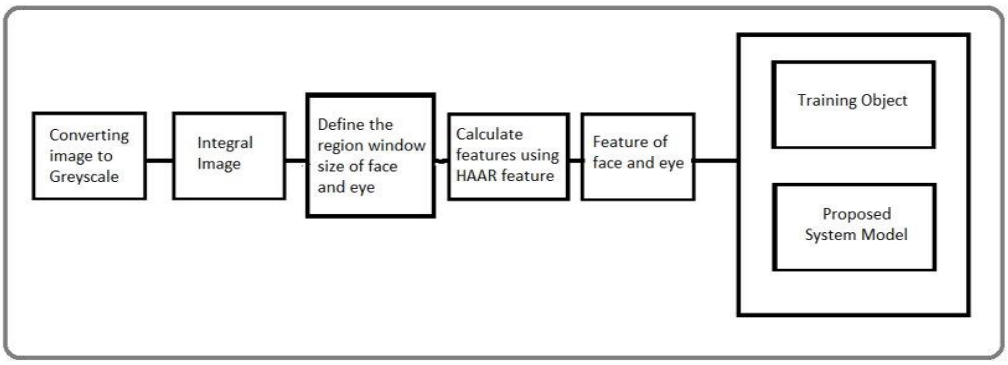
As the Haar features matrix is slid over the image the value of delta is calculated. Delta is given by the formula:

$$\Delta = \text{dark} - \text{white} = \frac{1}{n}\sum_{\text{dark}}^{n} I(x) - \frac{1}{n}\sum_{\text{white}}^{n} I(x)$$

The closer the value is to 1 it is considered as a match. (Ideal value of delta is 1)

5.3. METHODOLOGY

5.3.1. BLOCK DIAGRAM



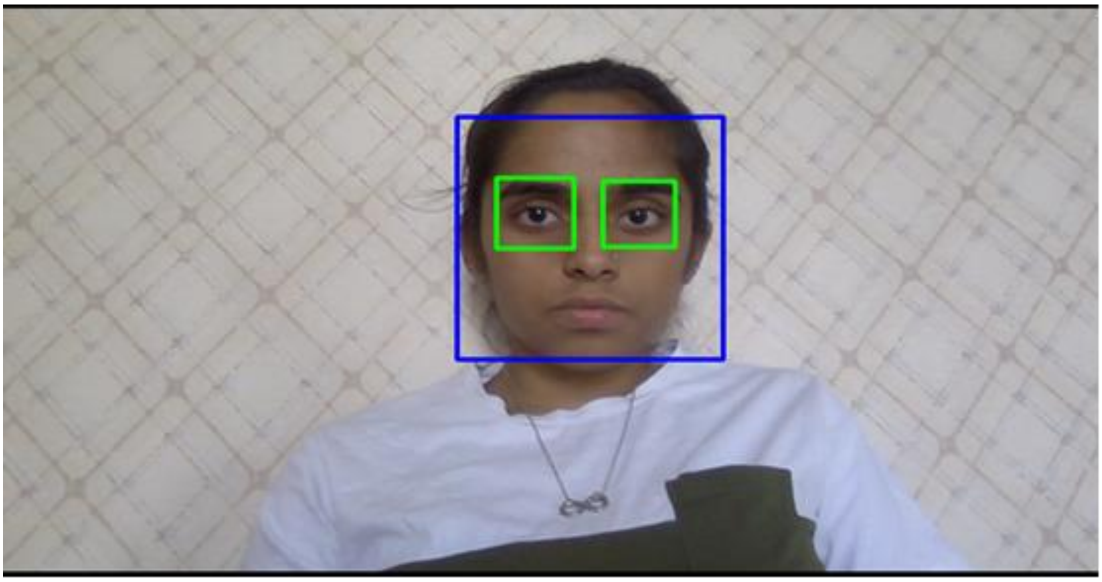
5.3.2. IMPLEMENTATION

To implement this computer vision example, we have used the OpenCV library. In this case we have used existing haar cascade classifier which are huge '.xml' files which contains lot of haar features. In this use-case we will try to detect the face of individuals using the haarcascade_frontalface_default.xml. Similarly, for eye detection the haarcascade_eye_default.xml. Since Haar-like features classifier or cascades only works with greyscale image, we need to convert our image to greyscale. We capture video from your webcam and pick a frame from those videos.

5.3.3. SOME APPLICATIONS

Face detection is the ability to detect and locate human faces in a photograph. This is the first step to make an application for face recognition, personality match, apply digital filters etc. This can further be used to get a count of number of people in each frame. This is the first step towards human machine interactions.

5.4 RESULT



CHAPTER: 06

MODEL 4: BODY DETECTION

6.1. INTRODUCTION

In this section we explore human computer interaction under the heading of body detection. Detecting human beings in images is a challenging task, for their varying height, colour and race provide computers a complex calculation of accounting for such diversity.
However body detection is a powerful tool in human computer interaction and finds applications in self driving cars, detecting pedestrians crossing the road, in surveillance systems etc.
Here we use the concept of Histogram of Oriented Gradients (HOG) to extract features of the human body and then use a linear Support Vector Machine(SVM) in order to detect human bodies in a video stream.

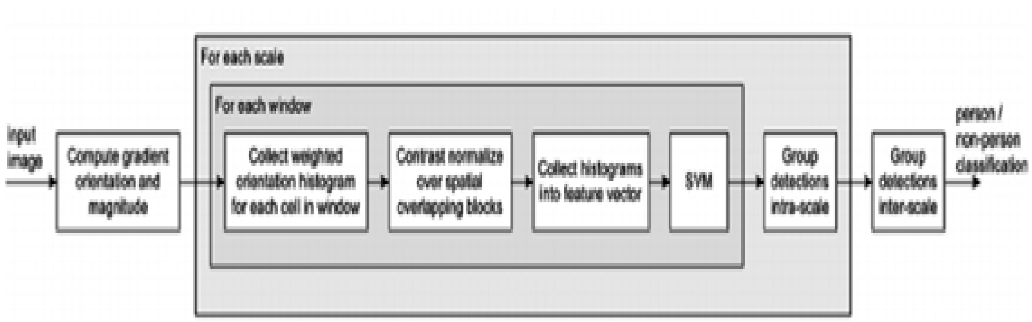
6.2. REVIEW OF LITERATURE

In 2001 the Haar Cascade classifier also known as the viola jones algorithms were considered the state of the art for feature extraction ,however since then more complicated and robust algorithms for feature extraction have been explored. Gavrila and Philomen built a pedestrian detector by extracting edges in images and matching them to a set of learned exemplars using chamfer distance. SIFT appearance detectors are popular approach for object detection.

SIFT algorithm was proposed by David L Gowe in 2004.Ronfard built an articulated body detector y incorporating SVM based limb classifiers over 1st and 2nd order Gaussian filters in a dynamic programming framework. Finally Dalal and Triggs introduced the use of HOG feature extraction to detect pedestrians on the MIT pedestrian database.

6.3. METHODOLOGY

6.3.1. BLOCK DIAGRAM



In this project we make use of opencv and python. OpenCV ships with a pre-trained HOG plus Linear SVM model that can be used to perform pedestrian detection in both images and video streams. We use this pre trained classifier on a video stream, from a web camera.

6.3.2. COMPUTING HOG

The Histogram of Oriented Gradients (HOG) is an efficient way to extract features out of the pixel colours for building an object recognition classifier

1. Calculating image gradient vector

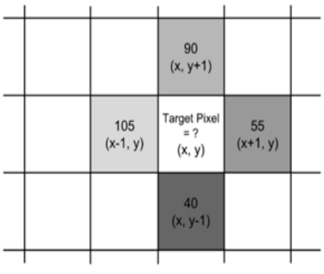
   The image gradient vector is defined as a metric for every individual pixel, containing the pixel color changes in both x-axis and y-axis. The definition is aligned with the gradient of a continuous multi-variable function, which is a vector of partial derivatives of all the variables. Suppose f(x, y) records the color of the pixel at location (x, y), the gradient vector of the pixel (x, y) is defined as follows:

   Magnitude is the L2-norm of the vector, g=g2x+g2y−−−−−−√g=gx2+gy2.

   Direction is the arctangent of the ratio between the partial derivatives on two directions, θ=arctan(gy/gx)θ=arctan(gy/gx).

$$\nabla f(x,y) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} f(x+1,y) - f(x-1,y) \\ f(x,y+1) - f(x,y-1) \end{bmatrix}$$

Now calculating this gradient magnitude and arctan for every pixel is a heavy computation cost. Hence predefined operators exist for this purpose which convolve on the image hence we call them kernels ,example prewitt ,sobel etc ,in our code we use sobel kernel



Sobel operator: To emphasize the impact of directly adjacent pixels more, they get assigned with higher weights.

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A \text{ and } G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

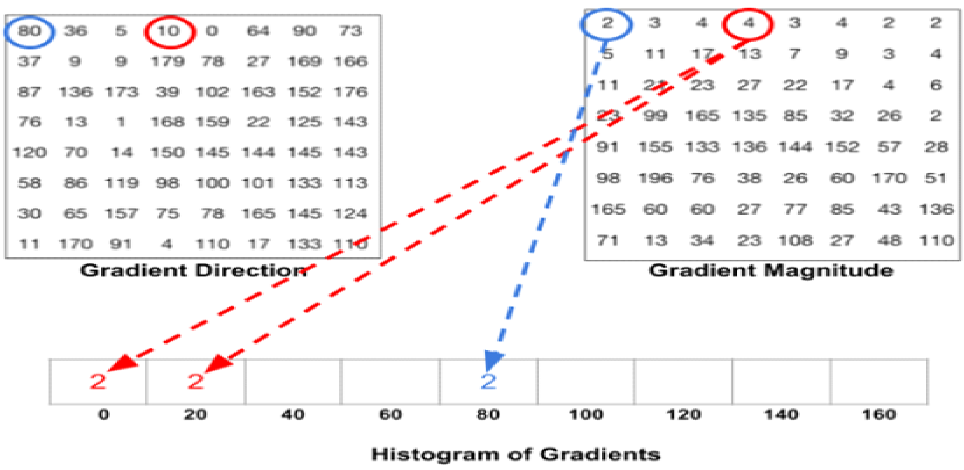
2. Calculating HOG

i) Pre-process the image, including resizing and color normalization.

ii) Compute the gradient vector of every pixel, as well as its magnitude and direction.

iii) Divide the image into many 8x8 pixel cells. In each cell, the magnitude values of these 64 cells are binned and cumulatively added into 9 buckets of unsigned direction (no sign, so 0-180 degree rather than 0-360 degree; this is a practical choice based on empirical experiments).For better robustness, if the direction of the gradient vector of a pixel lays between two buckets, its magnitude does not all go into the closer one but proportionally split between two. For example, if a pixel’s gradient vector has magnitude 8 and degree 15, it is between two buckets for degree 0 and 20 and we would assign 2 to bucket 0 and 6 to bucket 20.

iv) Then we slide a 2x2 cells (thus 16x16 pixels) block across the image. In each block region, 4 histograms of 4 cells are concatenated into one-dimensional vector of 36 values and then normalized to have an unit weight. The final HOG feature vector is the concatenation of all the block vectors. It can be fed into a classifier like SVM for learning object recognition tasks.



3. Training HOG

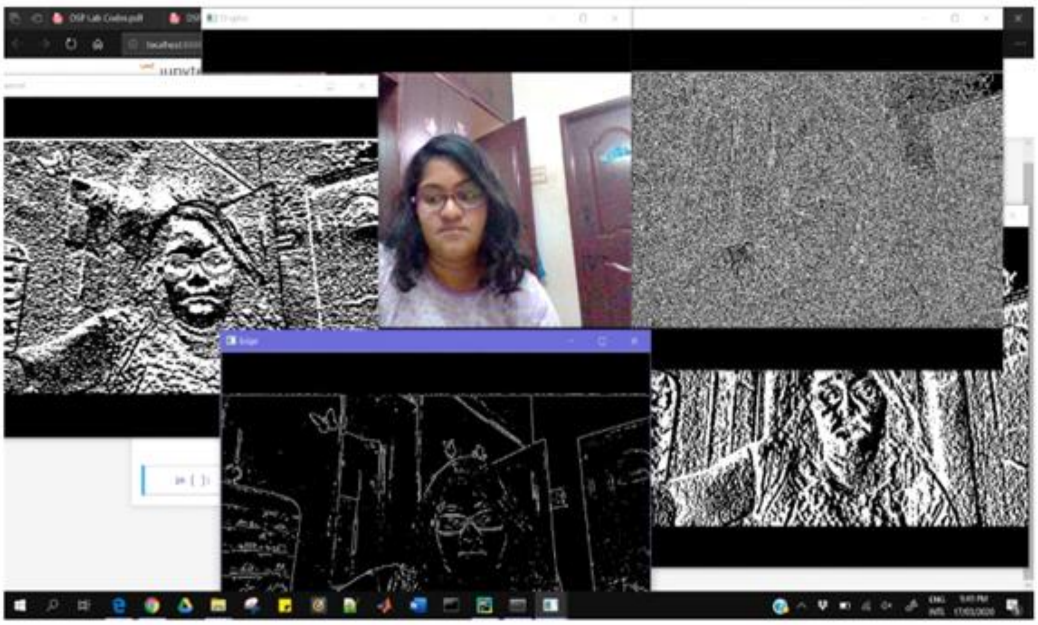
Step1: Sample P positive samples from your training data of the object(s) you want to detect and extract HOG descriptors from these samples.

Step2: Sample N negative samples from a negative training set that does not contain any of the objects you want to detect and extract HOG descriptors from these samples as well. In practice N >> P.

Step3: Train a Linear Support Vector Machine on your positive and negative samples.

Step4: Apply to dataset

IMAGE GRADIENTS



1.Original

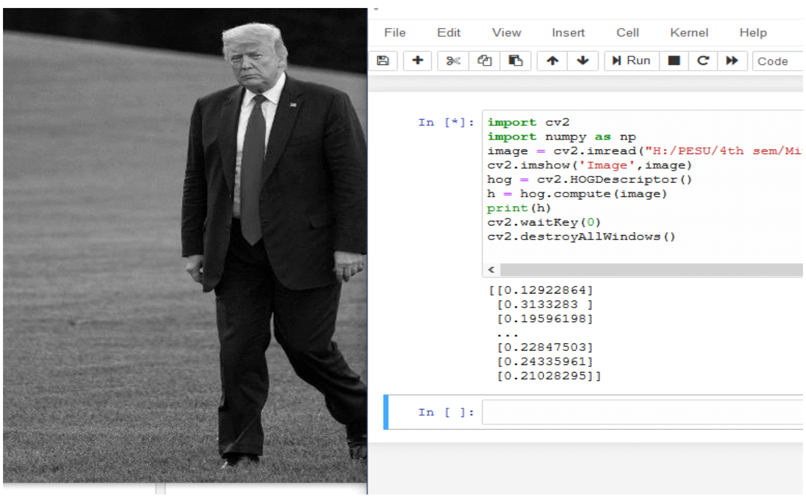
2.Horizontal Gradient

3.Vertical Gradient

4.Laplacian

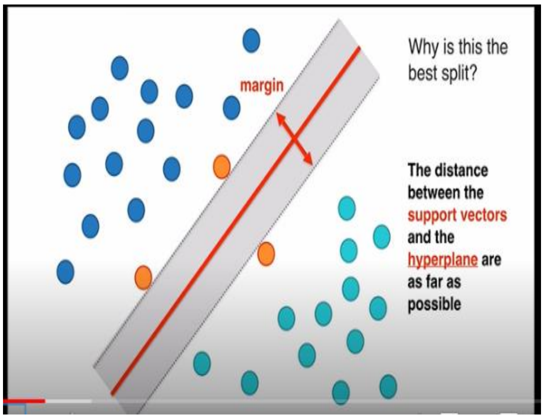
5.Edge detection

Sample HOG calculation of image



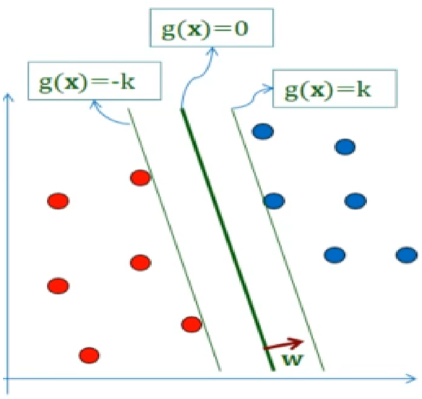
6.3.3. SUPPORT VECTOR MACHINE

SVM or support vector machine is the classifier that maximizes the margin. The goal of a classifier in our example below is to find a line or (n-1) dimension hyper-plane that separates the two classes present in the n-dimensional space.



The seperating line is called hyper plane and the nearest points to the hyperplane are called support vectors. A hyperplane is a line for 2d plane for 3d and so on. This is also know as constrained optimization constrained because the points cannot be on the line optimum because it is the widest separation. SVM are based on lagranges multipliers.

Formulation Of SVM



$g(x) = w^T x + b$

$Maximize\ k\ such\ that:$

$-w^T x + b \geq k\ for\ d_i == 1$

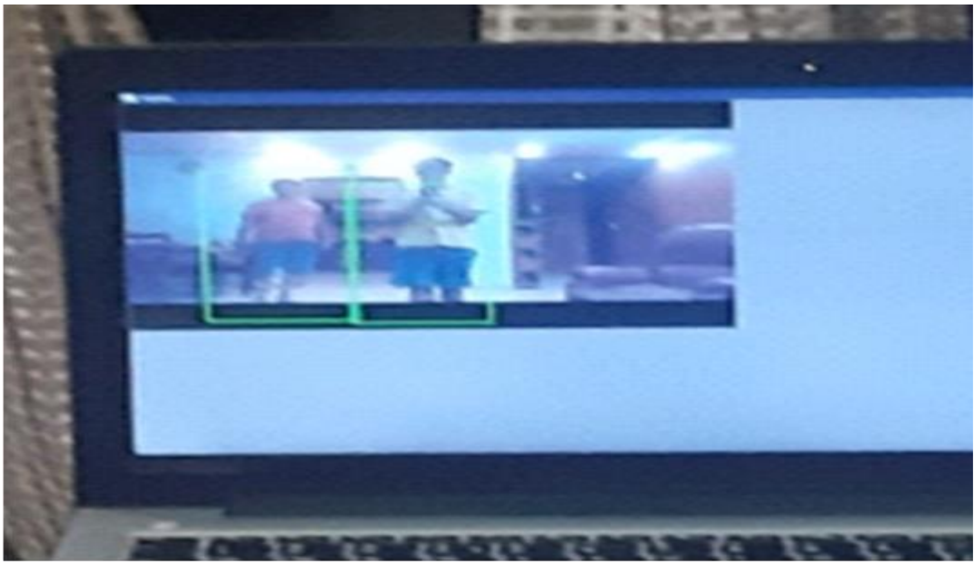
$-w^T x + b \leq k\ for\ d_i == -1$

$Value\ of\ g(x)\ depends\ upon\ ||w||:$

1) $Keep\ ||w|| = 1\ and\ maximize\ g(x)\ or,$

2) $g(x) \geq 1\ and\ minimize\ ||w||$

6.4. RESULT



CHAPTER: 07

CONCLUSION & FUTURE SCOPE:

By implementing this project we recognize that image processing along with computer vision based on machine learning has changed a lot of things around us.
There is immense opportunity in expanding our vision in this domain to come up with furthermore complex and efficient models than the ones implemented.
We conclude this report by suggesting various industrial scopes to portray the immense power of computer vision in general and human computer interaction.

We would be putting forward different applications for each of the models proposed by us.

Gesture recognition has been used to create a system which can identify specific human gestures and use them for device control or to convey information.

Tracking the movement of the eye using OpenCV and Python implemented for various hands-free applications like cursor movement, mouse scrolling, gaze determination, drowsiness calculation when driving, etc.

Haar Feature-based Cascade Classifiers has been used to detect faces and eyes. It identifies human faces and eyes given a video stream of pictures. These already have a wide application in face recognition and surveillance cameras.

Histogram of Oriented Gradients (HOG) is an efficient way to extract features out of the pixel colours for building an object recognition classifier which has been used for body detection.

In view of the pandemic of COVID-19, we also realize that hands free applications of devices can lead to a healthier and safer surrounding also giving confidence to people in the usage of many systems. The face detection can further be modified to detect masks on the face in real time. Whereas the body detection can find its use in detecting human presence as well as viewing social distancing norms.

CHAPTER: 08

REFERENCES:

[1] Ruchi Manish Gurav, Premanand K. Kadbe; "Real time Finger Tracking and Contour Detection for Gesture Recognition using OpenCV"; 2015 International Conference on Industrial Instrumentation and Control (ICIC) College of Engineering Pune, India. May 28-30, 2015

[2] Satyam M Achari1, Shashwat G Mirji2, Chetan P Desai3, Mailari S Hulasogi4, Sateesh P Awari5; "GESTURE BASED WIRELESS CONTROL OF ROBOTIC HAND USING IMAGE PROCESSING"; International Research Journal of Engineering and Technology (IRJET) Volume: 05 Issue: 05 | May-2018

[3]Harish Kumar Kaura, Vipul Honrao, Sayali Patil, Pravish Shetty;"Gesture Controlled Robot using Image Processing "; (IJARAI) International Journal of Advanced Research in Artificial Intelligence, Vol. 2, No. 5, 2013

[4]Supriya Zinjad ; "Gesture Recognition Robot Using Digital Image Processing";International Research Journal of Engineering and Technology (IRJET); Volume: 05 Issue: 01 | Jan-2018

[5] https://en.wikipedia.org/wiki/Gesture_recognition

[1] https://pysource.com/2019/01/04/eye-motion-tracking-opencv-with-python/

[2] https://en.wikipedia.org/wiki/Eye_tracking

[1] Souhail Guennouni ; Ali Ahaitouf ; Anass Mansouri; " Face detection: Comparing Haar-like combined with cascade classifiers and Edge Orientation Matching"; Publisher: IEEE

[2] Paul Viola ; Michael Jones Rapid Object Detection using a Boosted Cascade of Simple Features; Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern_Recognition_IEEE_Computer_Society_Conference_on_Computer_Vision_and_Pattern_Recognition

[3] Classifier:https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_frontalface_default.xml; https://github.com/opencv/opencv/blob/master/data/haarcascades/haarcascade_eye.xml; https://pythonprogramming.net

[1] https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/

[2] N. Dalal, B. Triggs, “Histogram of oriented gradients for human detection,” IEEE Conference on Computer Vision and Pattern recognition, vol. 2, pp. 886–893, June 2005.

[3] GL. Gan, J. Cheng, “Pedestrian detection based on HOG-LBP Feature,” International Conference on Computational Intelligence and Security,pp. 1184-1187, Dec 2011