

# **MOUSE CURSOR CONTROL USING FACIAL EXPRESSION**

**A Project Report submitted in partial fulfilment of the requirements for the  
award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE ENGINEERING**

**Submitted by**

P Madan Venkat Sai

121910311061

N Harish Reddy

121910311044

S Akshay

121910311013

K Tarun Sai

121910311043

**Under the esteemed guidance of**

**Dr. M Rajamani**

**Assistant Professor**



**DEPARTMENT OF**

**COMPUTER SCIENCE & ENGINEERING**

**GITAM (Deemed to be University)**

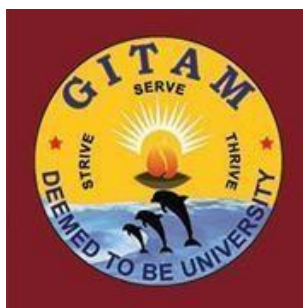
**VISAKHAPATNAM**

**OCTOBER 2022**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**GITAM SCHOOL OF TECHNOLOGY**

**GITAM**

**(Deemed to be University)**



**DECLARATION**

We, hereby declare that the project report entitled “**MOUSE CURSOR CONTROL USING FACIAL EXPRESSIONS**” is an original work done in the Department of Computer Science and Engineering, GITAM School of Technology, GITAM (Deemed to be University) submitted in partial fulfillment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

**Registration No(s).**  
**Signature(s)**

**Name(s)**

**121910311061**

**MADAN VENKAT SAI POLAROUTHU**

**121910311044**

**HARISH REDDY NAREDLA**

**121910311013**

**AKSHAY SANKINENI**

**121910311043**

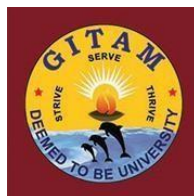
**TARUN SAI KARUMANCHI**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GITAM SCHOOL OF TECHNOLOGY**

**GITAM**

**(Deemed to be University)**



## **CERTIFICATE**

This is to certify that the project report entitled “**MOUSE CURSOR CONTROL USING FACIAL EXPRESSIONS**” is a bonafide record of work carried out by **Madan Venkat Sai Polarouthu (121910311061), Harish Reddy Naredla (121910311044), Akshay Sankineni (121910311013), Tarun Sai Karumanchi (121910311043)** students submitted in partial fulfillment of requirement for the award of degree of Bachelors of Technology in Computer Science and Engineering.

**Project Guide**

**Head of the Department**

**M Rajamani**  
**Assistant Professor**

**Dr. R.Sireesha**  
**Professor**

## **ACKNOWLEDGEMENT**

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned the efforts with success. It is a pleasant aspect that I have now the opportunity to express my gratitude for all of them.

The first person I would like to thank my project guide Assistant Professor Dr. Rajamani M, who had given continuous critical suggestions and extension of proper working atmosphere, abiding interest has finally evolved into this research work.

I would like to express my sincere thanks to Prof. R. Sireesha, Head of the Department of Computer Science and Engineering for providing the opportunity to Undertake this project and encouragement in the completion of the project.

I am also thankful to all the staff members of the Computer Science and Engineering Department for their valuable suggestions. I would like to thank my team mates who extended their help, encouragement and moral support either directly or indirectly in this project.

Madan venkat sai Polarouthu (121910311061)

Harish Reddy Naredla (121910311044)

Akshay Sankineni (121910311013)

Tarun Sai Karumanchi (121910311043)

## TABLE OF CONTENTS

<b>S. No</b>	<b>Contents</b>	<b>Page No</b>
1.	Abstract	3
2.	Introduction	4
3.	Literature Review	5-7
4.	Problem Identification & Objectives	8-9
5.	System Methodology	10-15
6.	Overview of Technologies	16-23
7.	Implementation	24-32
7.1	Coding	25-29
7.2	Testing / Output	30-32
8.	Results & Discussions	33
9.	Conclusion & Future Scope	34
10.	References	35

### List Of Tables:

<b>Table No</b>	<b>Table Name</b>	<b>Page No</b>
4.1	Existing & Proposed system Analysis	9
8.1	Action and its Function pairs	33

## List of Figures:

<b>Fig No</b>	<b>Fig Name</b>	<b>Page. No</b>
5.1	Block diagram of HCI system model	10
5.2	Class Diagram	11
5.3	Usecase Diagram	12
5.4	Component Diagram	13
5.5	Statechart Diagram	14
5.6	Activity Diagram	15
6.1	Filters or Kernels or Feature detector	17
a	Horizontal Kernel	
b	Vertical Kernel	
6.2	Resultant Absolute Gradient	17
a	Absolute X-Gradient	
b	Absolute Y-Gradient	
c	Magnitude of Gradient	
6.3	HOG Visualization	18
6.3.a	RGB image & Gradient Arrows	
6.3.b	Gradient numbers	
6.4	Rules 1 & 2 Visualization	19
6.5	Rule 3 Visualization	20
6.6	9-bin Histogram	20
6.7	Block Normalization	21
6.8	2D- Facial Landmarks	22
6.9	Eye Aspect Ratio [EAR]	23
6.10	Mouth Aspect Ratio [MAR]	23
7.1.1 - 7.1.10	Code	25 - 29
7.2.1 - 7.2.8	Output	30 - 32

## 1. ABSTRACT

The Human-Computer Interaction application i.e., cursor control using facial movements works with just regular webcam. The system targets on using user's facial movements. Its free of hands, avoidance of wearable hardware or sensors. The application deeply centred around facial landmarks prediction for face from eye-blinks and mouth movements detection in a video. Actions like mouth open to activate or deactivate mouse control, Right Eye wink to right click, Left Eye wink for left click, Squinting Eyes for activate or deactivate scrolling, Head Movements (Pitch and Yaw) –scrolling or cursor movement. With the help of these landmarks of the face make use of building corresponding features for further allowance to detect certain actions, for detection of winking and blinking of eye using eye-aspect-ratio and yawn and pout of mouth using mouth-aspect-ratio. The actions act as triggers for the mouse control. The model mainly looks into two main factors while analysing, they are Eye-Aspect-Ratio (EAR) that helps for detecting winks and blinks etc. and Mouth-Aspect-Ratio (MAR) to get to know the status of mouth either closed or opened.

## 2. INTRODUCTION

With an advancement in technology, Computers has just become very important companion to us in daily routine. There might be change in behavior of system from simple task to specialized tasks. But they are erected for only ordinary people, as many of the input devices are generally cannot be accessed by the physically disabled. It's problematic, but not insolvable, for hindered people to use computer generally. There are many people who cannot move their hands and use keyboard or mouse due to disability. For this reason, these assisting applications are gaining importance today.

Using the directed cursor control and facial expression tracking technology a robust Human-Computer Interaction (HCI) application for disabled persons can be build. One can control the cursor just by moving their head and wink and blink of eyes easily. This application can be made compatible for both laptops and desktops as well. It allows operations like moving cursor in all directions, click events and scroll events. In this way a physically disabled can be served.



### 3. LITERATURE SURVEY

[1] “Mouse Cursor Control using Facial Movements” it is proposed human computer interaction system using facial movements and propose algorithm for multimodal human computer interaction system. Histogram of Pyautogui module is used and where as the current system can be enhanced to add capability to specified conduct and relatable associated functions. Other computer-generated bias can also be controlled using HCI techniques. When there are numerous faces present, the model may be improved by simply taking into consideration one facial movement. This human-computer interface system's design and implementation were complete. Observation and scar characteristics are employed for similar eye shadowing and merging other senses by opening the corresponding mouth. It is possible to control all mouse actions, including mouse scrolling and mouse pointer movement in the left, right, up, and down directions; and mouse clicks are controlled by using right click and left click, are efficiently enforced by suggested algorithms of the system for multimodal HCI.

[2] “Eyeball Movement based Cursor Control using Raspberry Pi and OpenCV”. Raspberry pi used with ARMv8 (BCM2837) processor. The Eye Aspect Ratio (EAR) method for detecting pupils. The Dlib model. In order to apply this system on platforms like smartphones, we may add new functions as well that could be used around practical situations for controlling the user's cursor. In order to give the handlers a complete working experience from turning on the computer system to turning it off, & building a series of operational units. The project could be expanded to increase the system's effectiveness in covering all mouse tasks using eye motions. A raspberry pi and OpenCV-based ocular movement-based cursor control system is created. In terms of cursor control, the actions taken using this system are simple. This system gives people new ways to operate the computer. From the Using a cursor control system based on eye movement inferred that the application of IoT can significantly advance the field of human computer interaction.

[3] “Cursor Movement Control Using Eyes and Facial Movements for Physically Challenged People” Dlib and the Haar Cascade algorithm are utilized. System-based image processing operations include Conversion from BGR to Gray. Image blurring and feature detection. This can be improved by creating, changing, or upgrading additional techniques such as clicking events and human computer interfaces that employ eye movement and blinking to control the cursor.

Without using your hands, you may control the computer by changing your face expressions and eye movements. For those who are incapacitated, being able to control the cursor with their eyes alone instead of needing a partner is beneficial.

[4] “Facial Expression Based Computer Cursor Control System for Assisting Physically Disabled Person” Acquisition of data. The Viola & Jones algorithm is used by the system to automatically identify the user's face. A standard web camera is used to record the subject's emotion, and a microcontroller is used to send data. The individual is told to make the mentioned facial expressions while four tiny, fluorescent stickers are attached to his or her face. This project's current emphasis is on developing the design using high-speed microcontrollers and various light settings. The ability to control the mouse cursor with facial expressions enables impaired people to utilize the device independently. Webcams are utilized to record facial movements, which are then divided into frames and turned into grayscale in subsequent frames. The suggested methodology is quite straightforward and effective at regulating cursor movement.

[5] “Controlling Mouse Cursor Using Eye Movement”. Limbus Tracking, Pupil tracking, Electrooculography, Saccade, Camera of the system or the position of the head should be kept still, continues moving of head or device gives inaccurate results. Eye lids and parts of limbus gives negative impact on this technique. This technique doesn't support infrared light effect. Difference between the iris and sclera is more so, border detection of the eye is difficult. Controlling mouse cursor using eye movement the mouse is being controlled by the eye movement using multiple libraries that require high disk space, as this is the old research python had been updated since then all the separated libraries are now available in a single module. This model does not support squint eyes and closing of eyes, there should be continues head movement and cannot be still, does not support red eye effect output may vary. Contrast changes between the iris and pupil makes the border detection

[6] “Enhanced Cursor Control using Eye Mouse” Haar like Object detectors Houghman Circle Detection Open CV Eyes position is not calculated accurately; they were just detected using algorithms. The other smaller circles were detected due to the incidental arcs detected by the eyelashes on the upper and lower eyelids. Whole Eye image is taken into consideration for detecting the movement. Separate techniques are used for detecting the iris and its movement. No scrolling. Scrolling can be installed by using any of the new techniques Enhanced cursor control using eye mouse the eyes are detected by taking the entire image of the eye and that may cause detecting other smaller circles due to incidental arcs detected by eyelashes. There is no special feature to avoid red eye effect(infra-red). Squinting of eyes cannot be detected and Scrolling is not available which is very important for mouse controlling and to access web pages. By using OpenCV, dib, Imutis and pyautogui we can get exact position of the eye, scrolling can be achieved.

[7] “Eyeball based Cursor Movement Control” Center of pupil helps in controlling the mouse cursor movement. Placing the SD card at SD or MMC port and boosting it using Raspberry Pi. Using dedicated hardware like Raspberry pi for the application sets drawback for implementation. As an extension there is scope for clicking events using eye winks without usage of hands. This takes an advantage for physically challenged people for performing all cursor operations. But it's limited to only just cursor movement but not click events and scrolling process.

## **4. PROBLEM IDENTIFICATION AND OBJECTIVES**

Nowadays computers playing a prominent role in our daily lives in areas such as study, work, entertainment. But some people cannot operate it because of some illness. There are various reasons for which people need a different kind of assistive devices for physically disabled people. In order to promote Human Computer Interaction, the idea of cursor control using facial movement will be of great use for handicapped and disabled.

### **Existing system:**

1. Hand Gestured Cursor Control where using fingers in hand as gestures to operate as required.
2. Eye Pupil based cursor control where center of eye that is pupil is calculated and optimal algorithm is used to control . whereas basic mouse operation like minimize, drag, scroll up, scroll down , left-click right-click operations cannot be performed.

### **Proposed system:**

The proposed system is to allow users to have control over the mouse using facial movements. The system works with normal webcam or the laptop's camera and it is hands-free where the user neither requires wearable hardware nor sensors. The actions include:

- Application activation using opening of mouth action.
- Scroll mode activation and deactivation using squinting of eyes.
- Cursor click events using winking of either the eyes.
- All directional cursor movement just with head movement.

Table 4.1 Proposed System with Existing System Analysis

S. No	System	Multimodal	Device Control Support	Supported Device Functions
1	Proposed Multimodal HCI	Yes	Yes	Movement of Cursor (Left, Right, Up, Down). Mouse Scroll (Up, Down), Click event (Right and Left)
2	Vision-Based Multimodal HCI	Yes	Yes	Movement of Cursor (Left, Right, Up, Down). Click Event (Right and Left)
3	Eye-gaze Tracking	No	No	No
4	Eye Blink Monitoring	No	No	No

Above Table 4.1 is the comparison analysis of existing systems to the proposed system with all the device functions that can be supported.

## 5. System Methodology

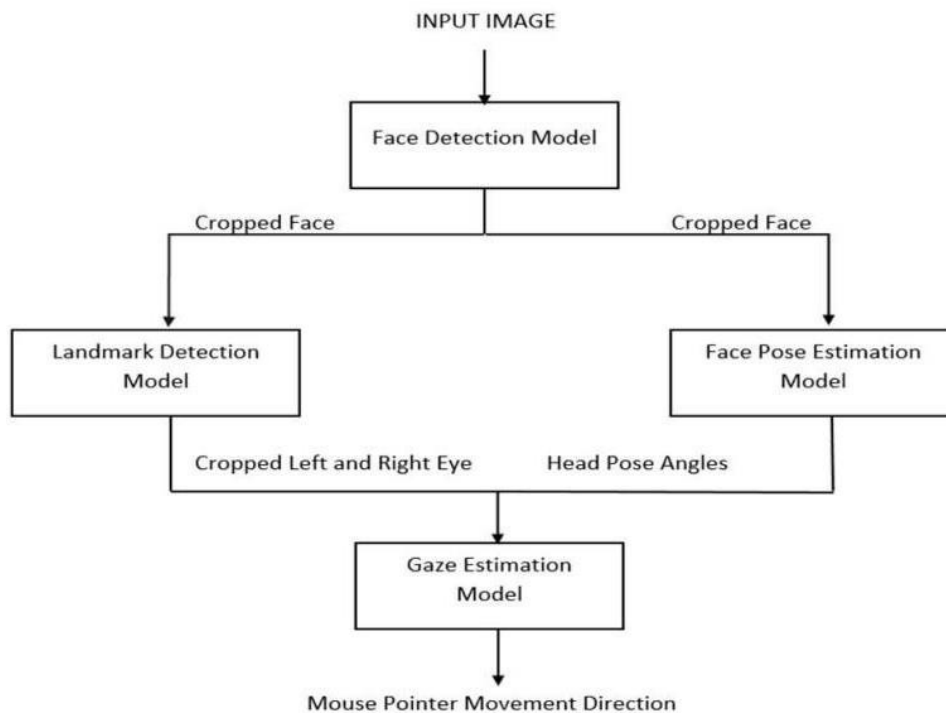


Fig. 5.1 HCI Block model

Above Fig. 5.1 is the block model consisting of models involved in application.

They are:

- **Face Detection Model**

Face image is captured with the videotape, which is taken from the webcam or camera.

- **Landmark Detection Model**

From the detected face, predefined land marks are superimposed, that is to detect the exact location of both the eyes and mouth.

- **Head Pose Estimation Model.**

Movement of head position associated to mouse cursor movement are predicted by using this model.

## UML Diagrams [ Structural & Behavioral]:

### Class diagram:

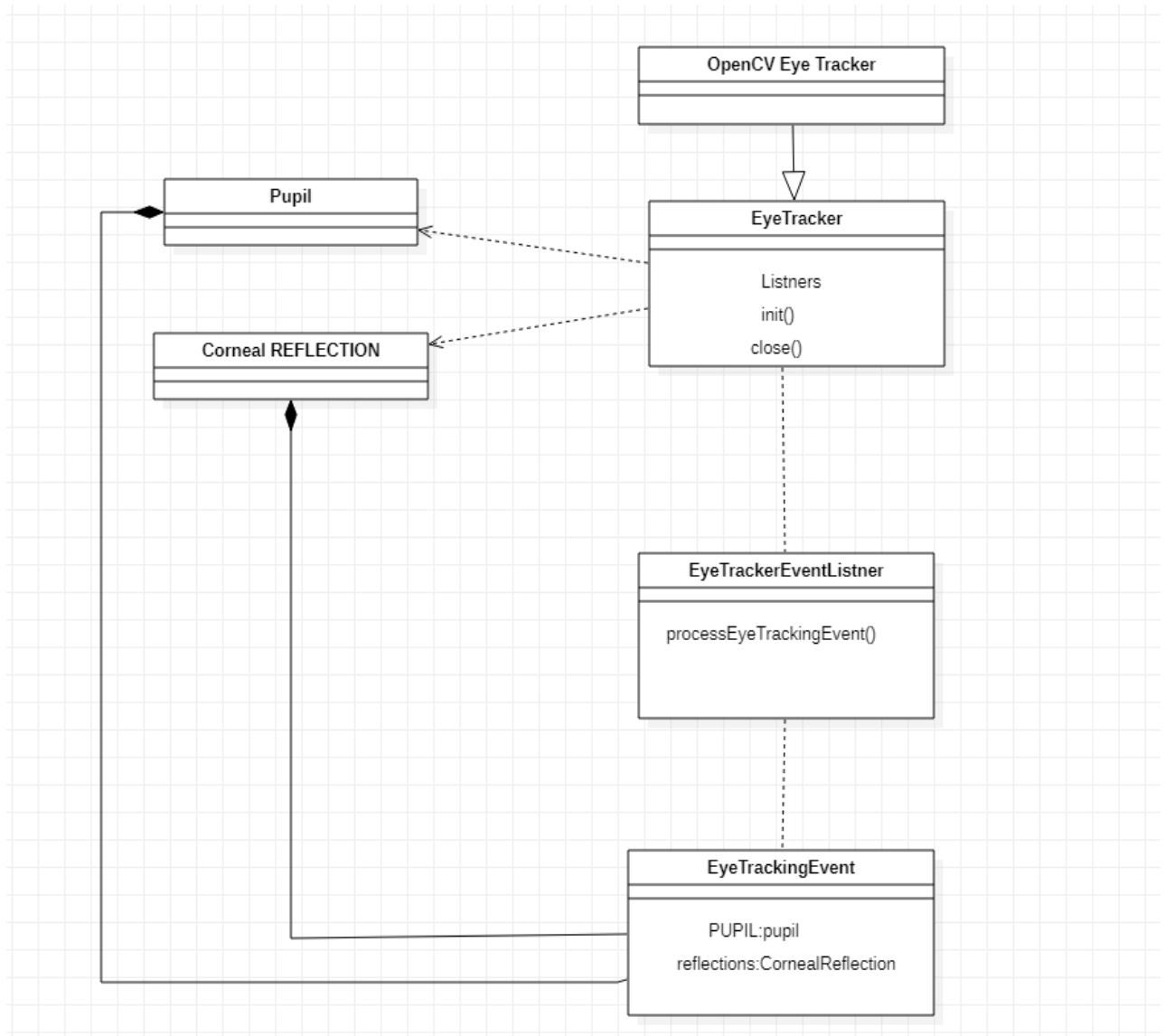


Fig. 5.2 Class Diagram

Above Fig. 5.2 illustrates the class diagram which is a static view of an operations involved.

### Usecase Diagram:

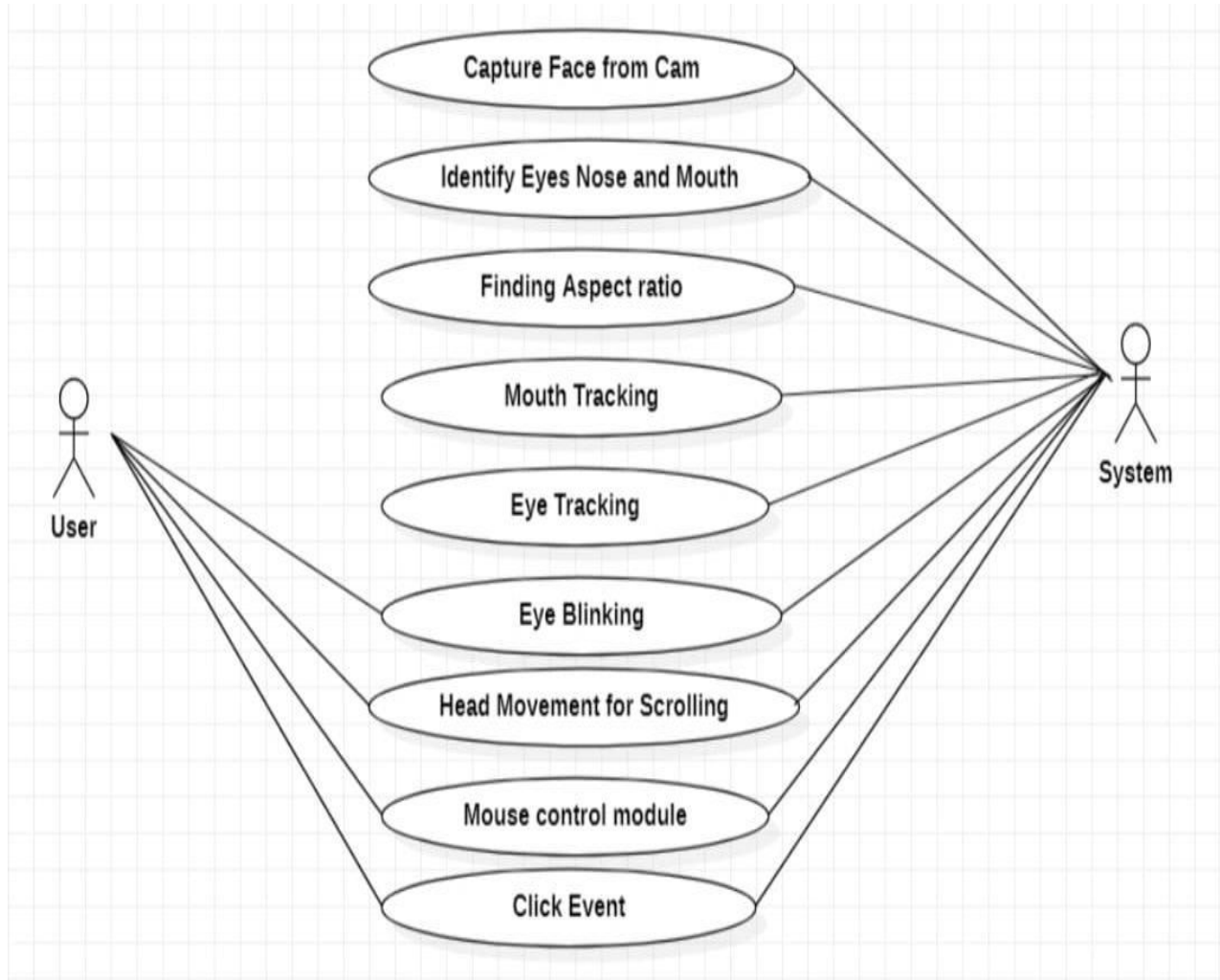


Fig. 5.3 Usecase Diagram

Above Fig. 5.3 is an illustration of Usecase diagram which consists of details of users & their relations with system.



## Component Diagram:

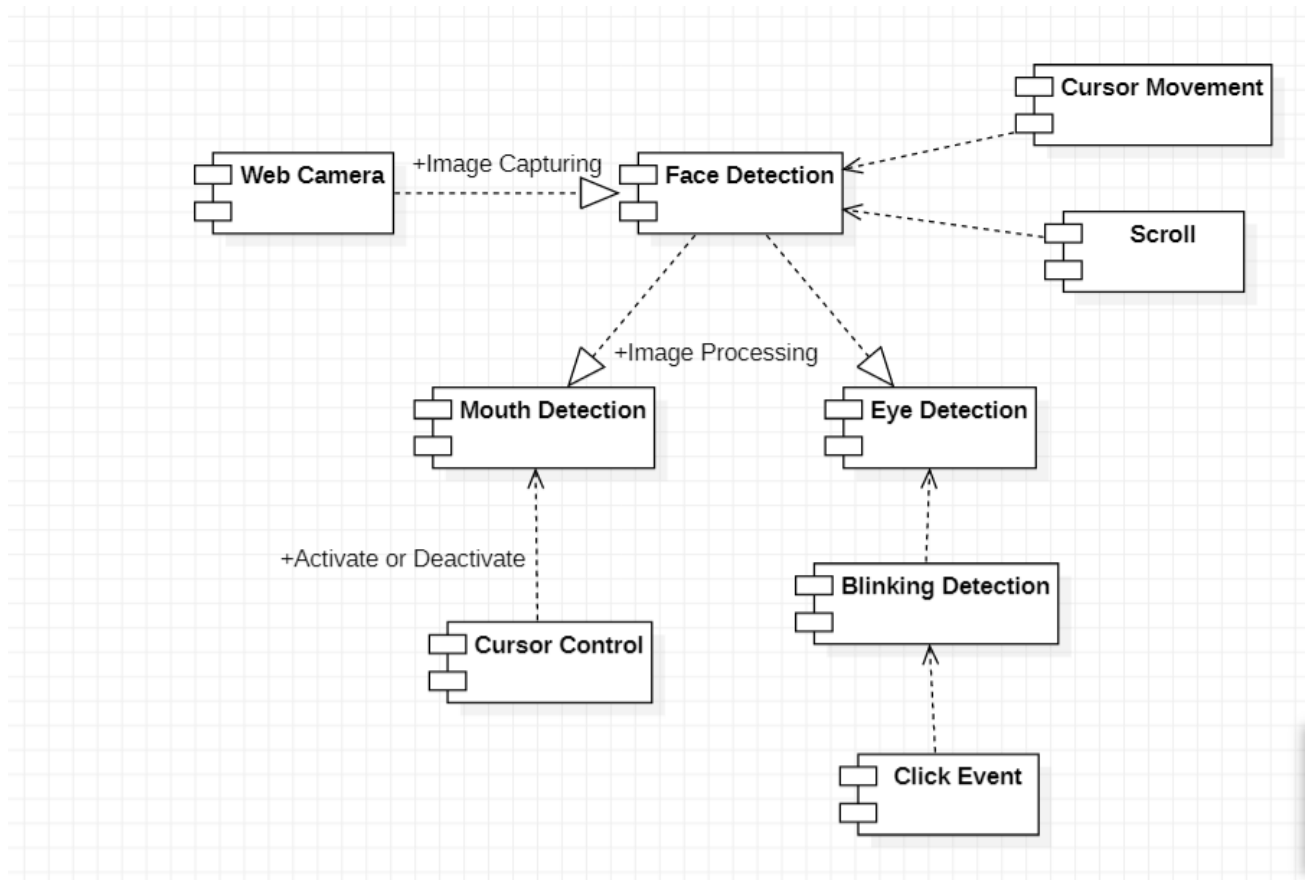


Fig. 5.4 Component Diagram

Fig 5.4 Illustrates the component diagram which is an structural diagram that includes all the components or models used in application are involved.

### State chart Diagram:

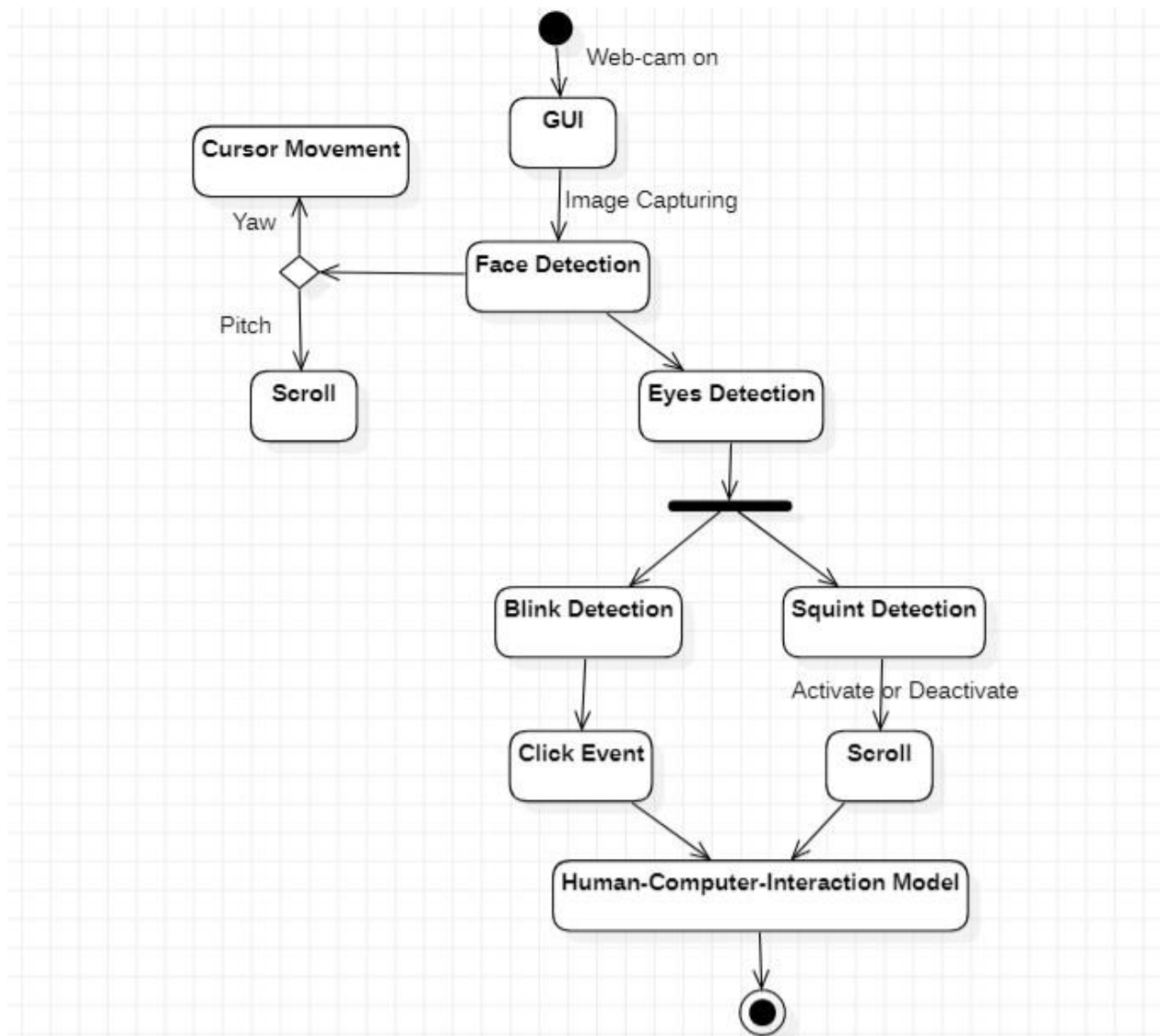


Fig. 5.5 Statechart Diagram

Above Fig. 5.5 illustrates the Statechart diagram which distinguishes between various states of an object

## Activity Diagram:

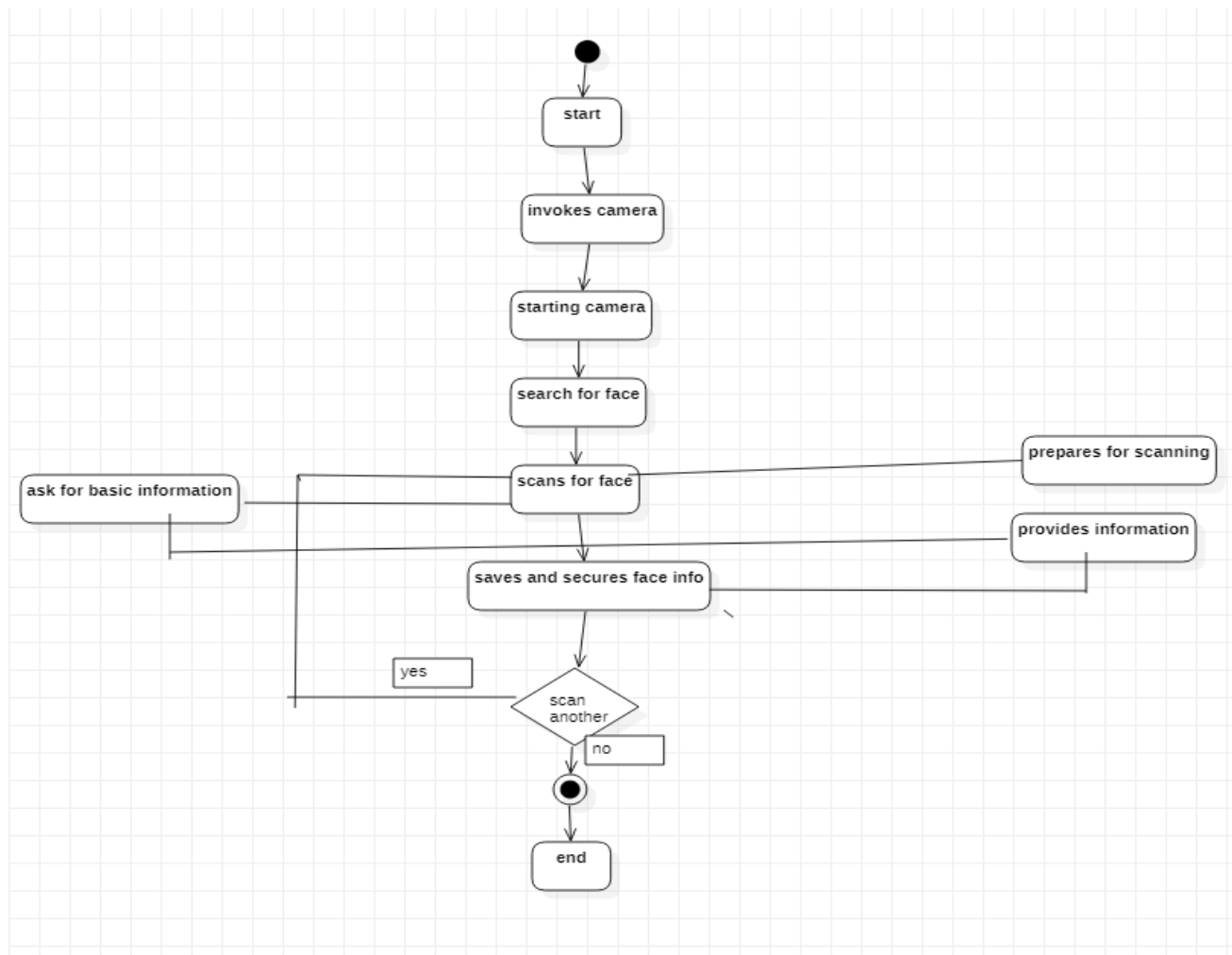


Fig. 5.6 Activity Diagram

Above Fig. 5.6 is an illustration of the activity diagram, which consists of activities and functions of system

## **6. OVERVIEW OF TECHNOLOGIES**

- **COMPUTER VISION**

Computer vision enables computers to honour objects through digital prints or pictures.

The use of Python to apply CV enables inventors to automate processes that need visualisation. Python outperforms the competition despite the fact that other programming languages include computer vision.

- **IMAGE PROCESSING**

Computer vision is a subset of image processing. A computer vision system attempts to pretend vision at the mortal scale using image processing styles. Image processing can be used, for case, if the end is to ameliorate the image for operation in the future. And it qualifies as computer vision if the ideal is to honor effects for automatic driving.

fashion used then for Image processing is Histogram of Oriented Gradients (HOG) and Support Vector Machine (SVM).

### **HISTOGRAM OF ORIENTED GRADIENTS [HOG]**

- This technique is to take out the vector from the features, and sending it into algorithm like a Support Vector Machine to find whether any face is present in a region or not.
- The image or patch of image can be represented by this feature descriptor by and takeout the useful information and eliminate unnecessary information.
- The features are histogram distribution of directions of slants (acquainted slants) of the image. slants are generally large around edges and corners and allow us to descry those regions.

### **SUPPORT VECTOR MACHINE [SVM]**

- A well-defined Binary Classification Algorithm.
- The feature vector produced from this algorithm when send to Support Vector Machine (SVM) outputs a labelled result.

## Step 1: Images Calculating Gradient

a. Horizontal Kernel

b. Vertical Kernel

Fig. 6.1 Filters or kernels or feature detector.

Above Fig. 6.1 shows two different matrices i.e., row matrix (Fig a), column matrix (Fig b) which is used as filter to pass and compare over the image matrix in row wise and column wise respectively.

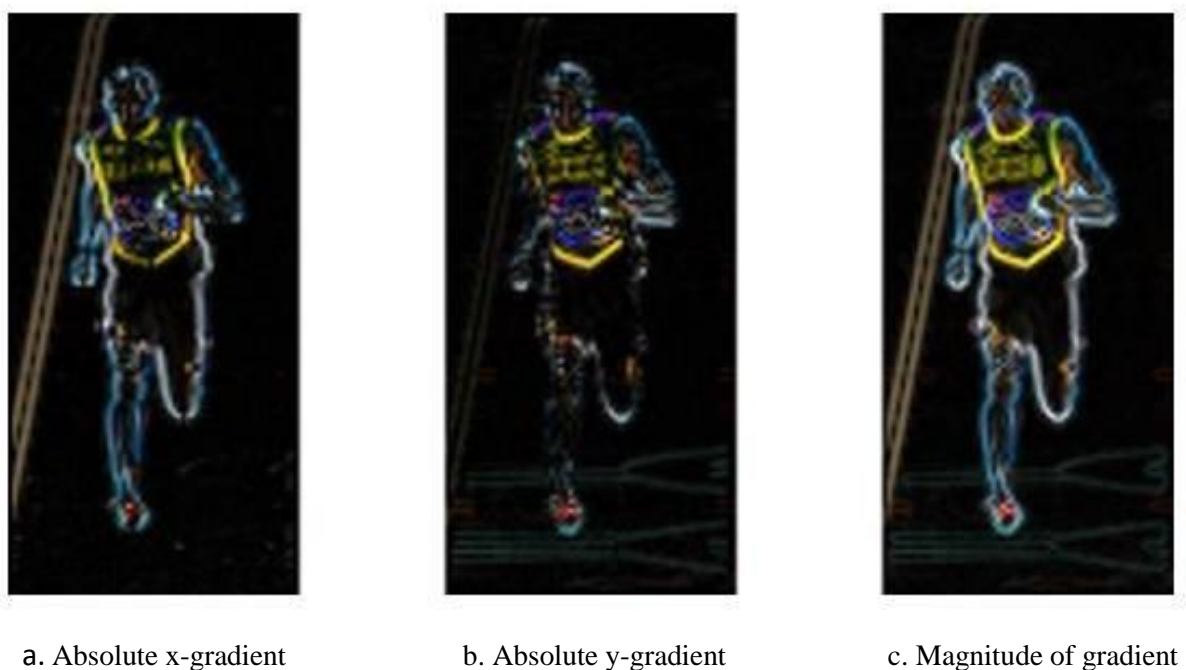


Fig.6.2 After filtering resultant absolute gradients:

Above Fig. 6.2 is an illustration of resultant image in an eliminated non-essential information form (example: background colour), but highlighted outlines.

## Step 2: Histogram of Gradients in 8×8 cells Calculation

- Here angles are taken from 0 to 180 degrees rather 0 to 360 degrees. These are known as “unsigned” gradients since its grade and it’s negative are represented by the same figures.
- The orders of the histogram correspond to angles of the grade, from 0 to 180 degrees
- There are 9 orders overall 0, 20, 40, 160.

We also calculate 2 basic required information

- Direction of the grade
- Magnitude of the grade

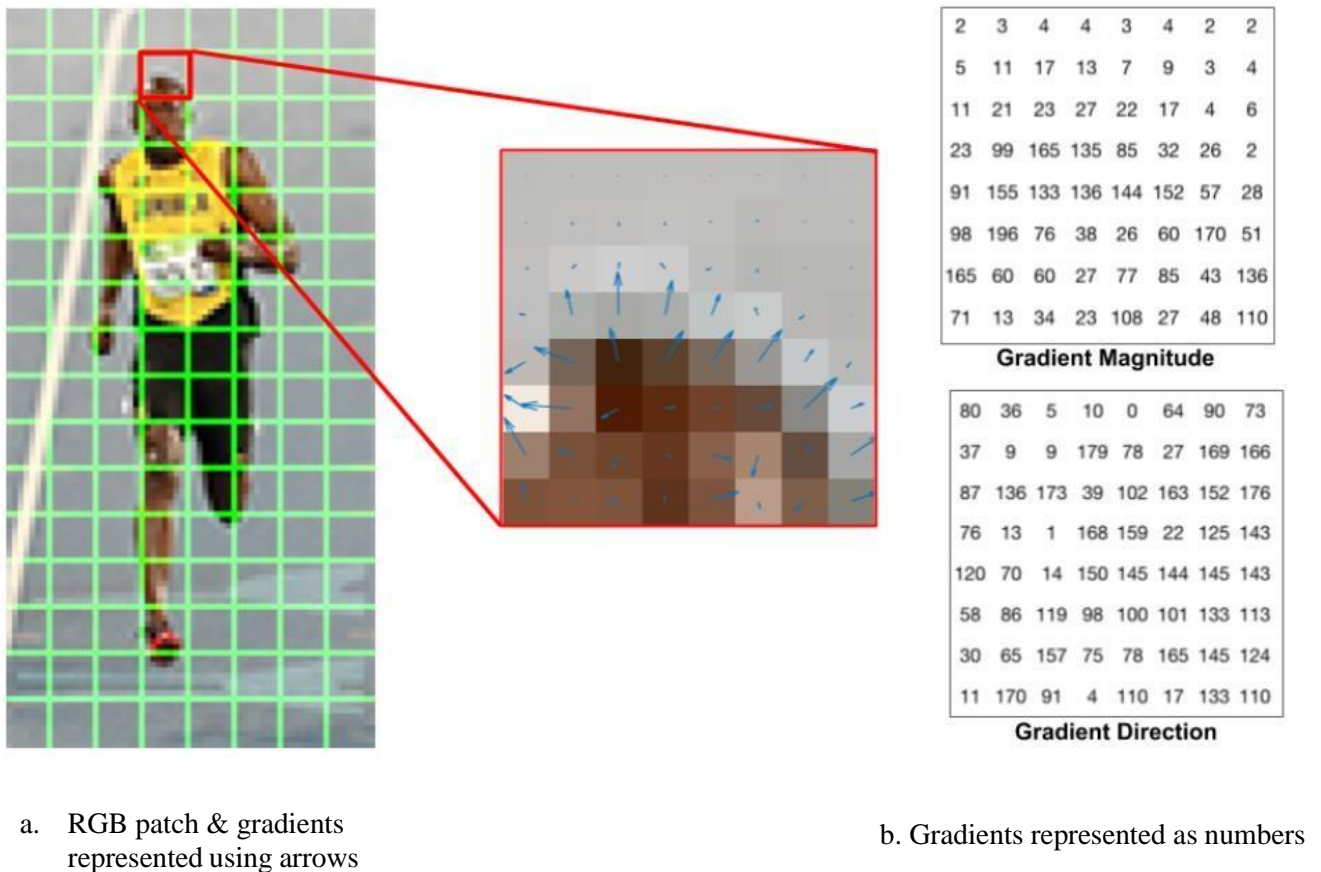


Fig 6.3 Resultant Absolute Gradients

Fig 6.3 Illustrates the clear visualization of the gradient information(a) and its corresponding tabular values of it magnitude and direction (b).

- **Rule 1:**

Angle is lower than 160 degrees and not half in between 2 classes. Then angle will be summed in the right order of the histogram

- **Rule 2:**

Angle lower than 160 degrees & exactly between two classes. Then we treat an equal donation to the two close classes and resolve the magnitude in two.

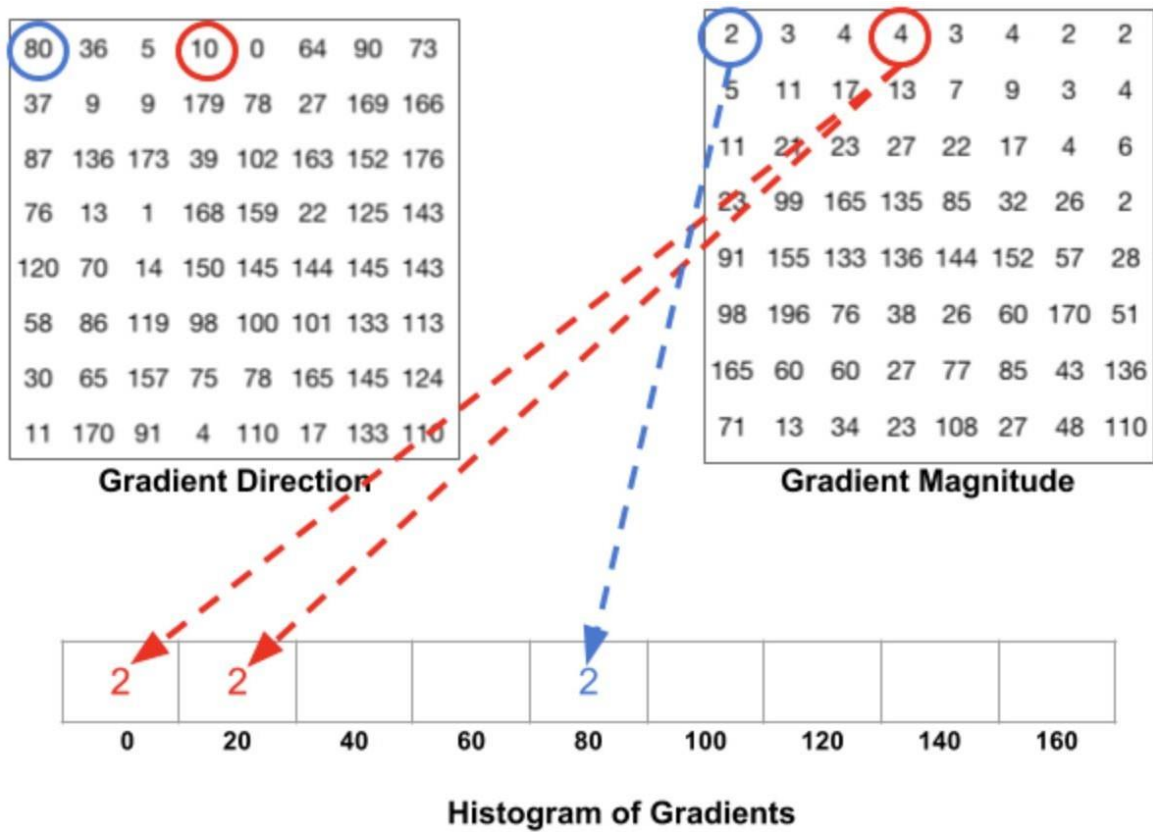


Fig. 6.4 Rule1 and 2 visualization with an example

Above Fig. 6.4 is the illustration of computation as per #rule 1 and #rule 2, the magnitude of each grade is divided and directly assigned to the histogram orders independently.

- **Rule 3:**

If angle larger than 160 degrees. In similar case, we contribute the pixel proportionally to 0° and to 160 °.

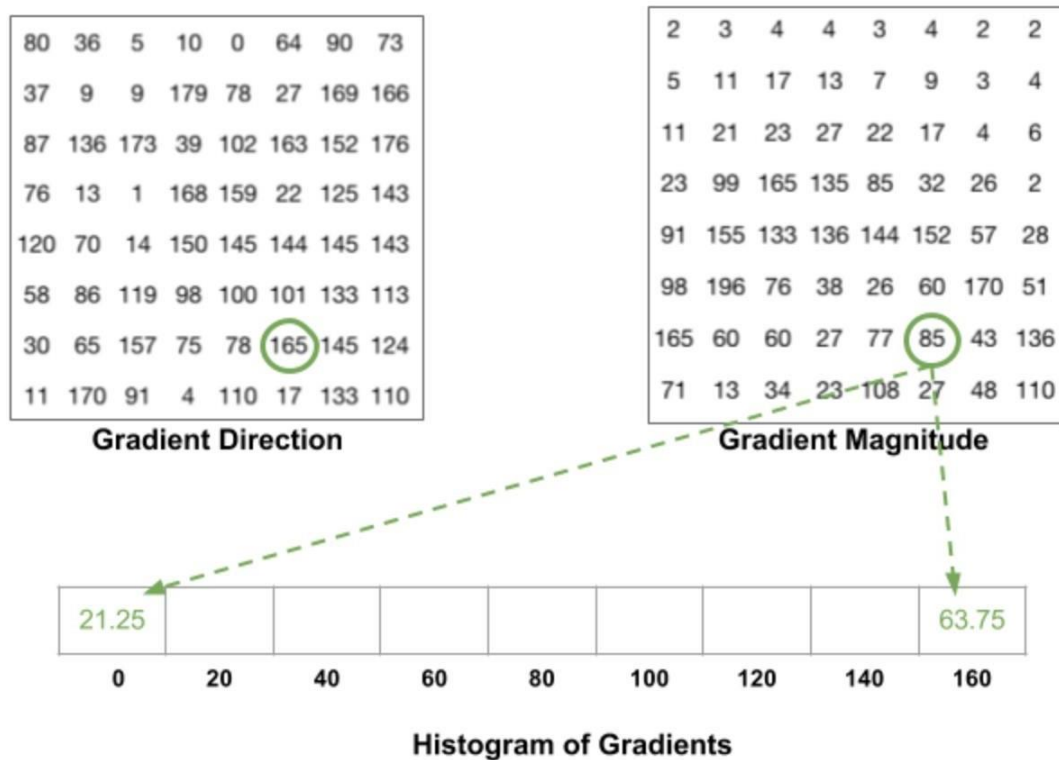


Fig. 6.5

Above Fig 6.5 illustrates the computation of rule 3, which is to divide the magnitude values in proportional to the extreme angles 0 and 160.

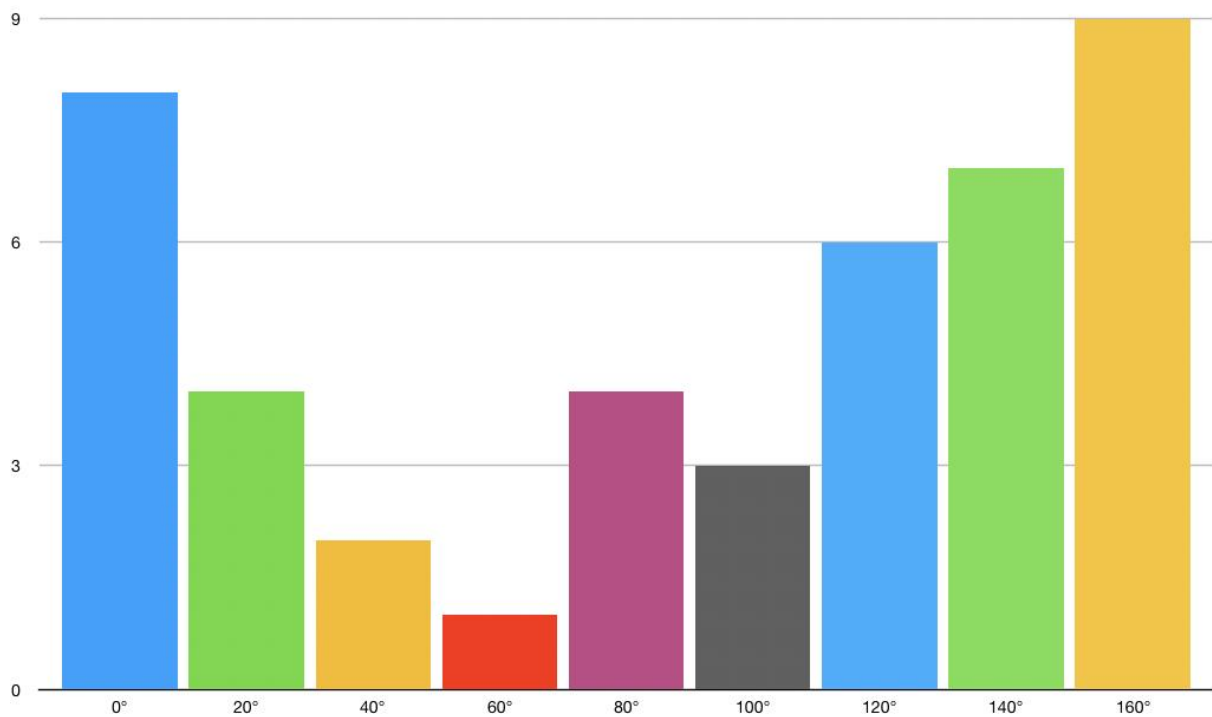


Fig. 6.6 All the pixels of  $8 \times 8$  blocks add up to form 9- bin histogram.



Above Fig. 6.6 illustrates our visualization, the y- axis is 9-bin and x-axis is oriented degrees. Most of the weights are fallen near to the angles 0 and 180, Nothing but another way of presenting that gradients are pointing either up or down.

### Step 3: $16 \times 16$ Cell Normalization

- Immaculately, to get free from light variations of our descriptor. That's to “homogenize” the histogram so that they aren't effected by variations in light.

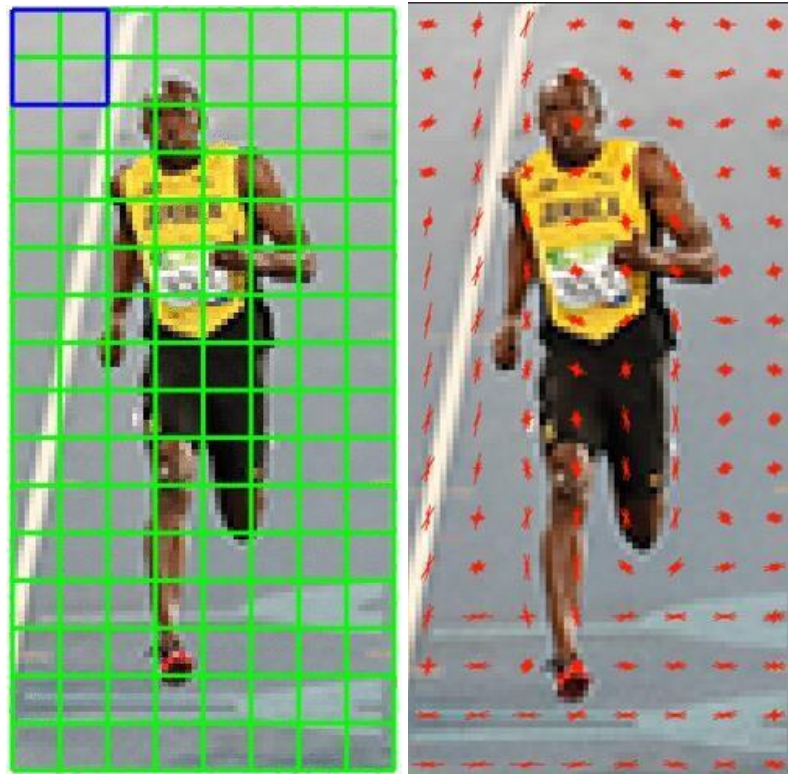


Fig. 6.7 Block normalization visualization

Above Fig. 6.7 is the vector regularization which is done as the  $3 \times 1$  vector regularization, a  $36 \times 1$  element vector is regularized form the concatenation of  $16 \times 16$  block (4 histograms).

- A  $36 \times 1$  vector is calculated by moving the window of 8 pixels and this step is repeated all over the frame.
- Finally we obtain  $36 \times 105 = 3780$  dimensional vector by concatenating all the pool of vectors.

**Final feature vector is transferred as input to the Support Vector Machine (SVM) for clear bracket of given data and return affair as labelled data.**

### Facial Landmarks Prediction Model:

- Predicted marks on face lets us process with face, that include:
  1. Mouth
  2. Eye [right]
  3. Eye [left]
  4. Nose
  5. Jaw

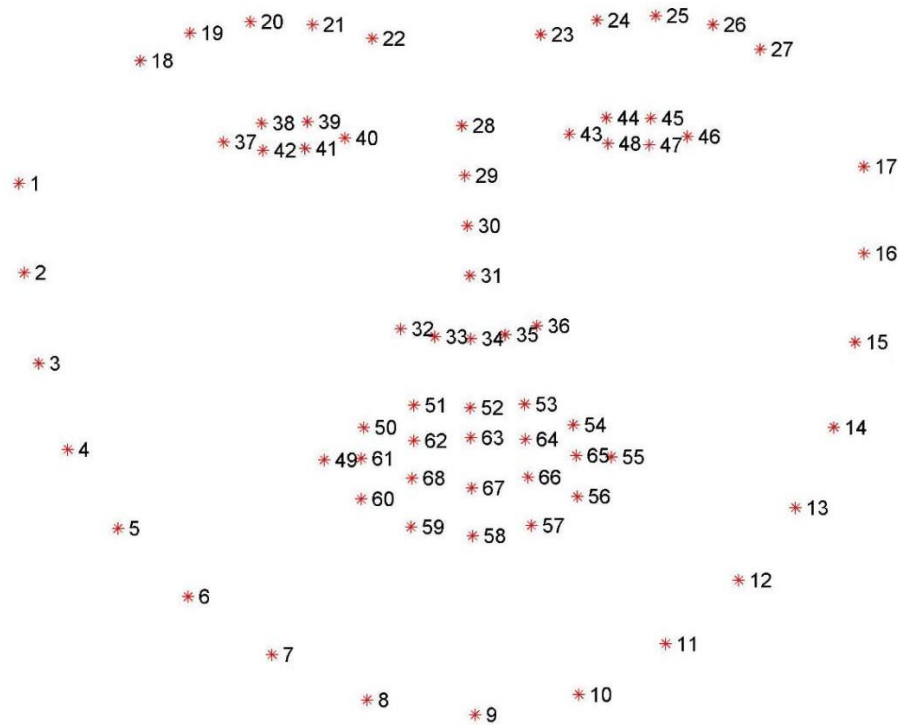
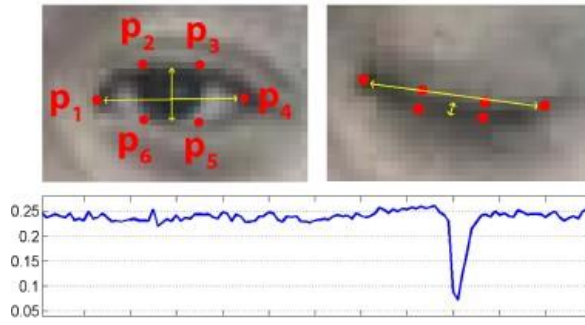


Fig. 6.8 2D Facial landmarks

Above Fig 6.8 illustrates positions 68 2D (x, y) co-ordinates equals to face structure. Which is predicted using previously trained marks detector in dlib library.

- Definite features are obtained from the marks superimposed on face, which lets us to detect blinks using Eye- Aspect-Ratio and Mouth- Aspect- Ratio
- PyAutoGUI module helps to initiate the cursor movements.

- **Eye Aspect Ratio [EAR]:**



$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|},$$

Fig. 6.9 Eye Aspect Ration visualization

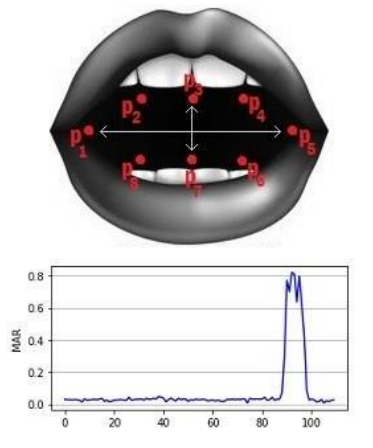
Above Fig. 6.9 illustrates the capturing blinks and winks of the eye. However, also the threshold values fall to zero if eye is in close state.

**Syntax:**

If  $EAR \leq \text{THRESHOLD}$ :

STATUS = 'CLOSE'.

- **Mouth Aspects Ratio [MAR]:**



$$MAR = \frac{\|p_2 - p_8\| + \|p_3 - p_7\| + \|p_4 - p_6\|}{2\|p_1 - p_5\|}$$

Fig. 6.10 Mouth Aspect Ratio

Above Fig. 6.10 illustrates capturing yawn and pout of mouth. When mouth is in opened state the defined threshold increases else, it falls.

**Syntax:**

if  $MAR \leq \text{THRESHOLD}$ :

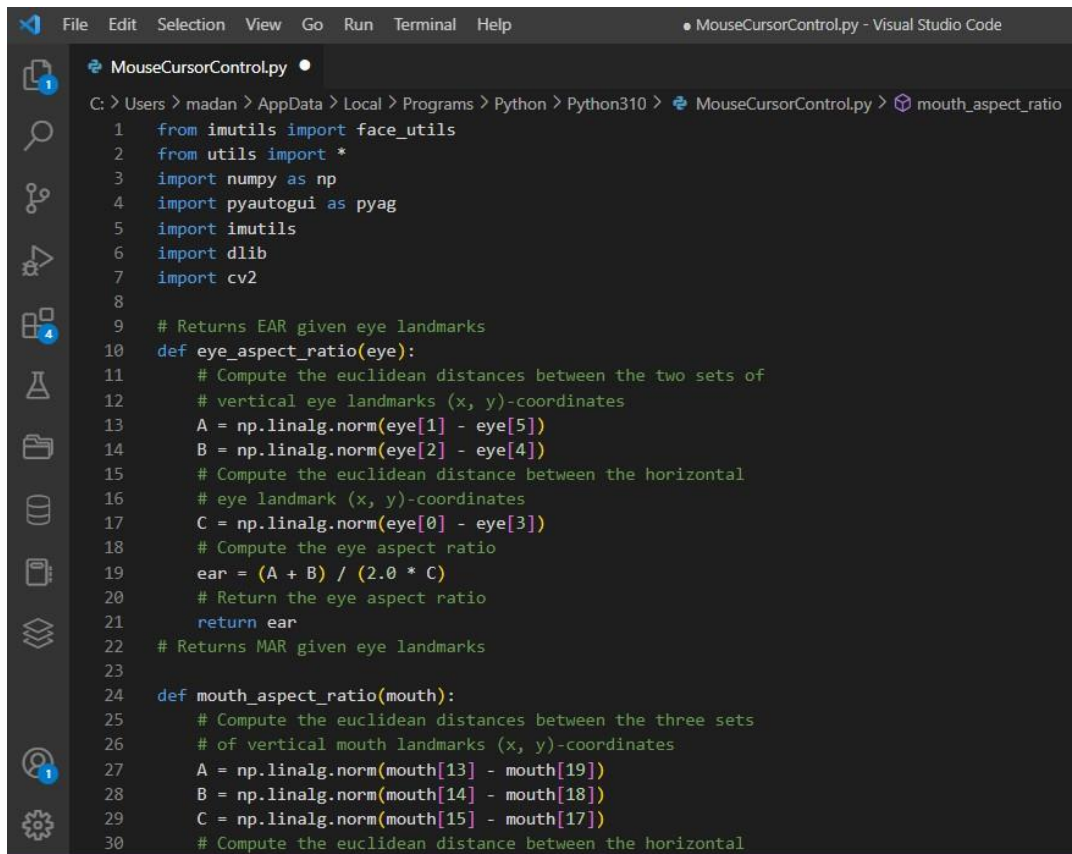
STATUS = 'CLOSE'.

## 7. IMPLEMENTATION

### Steps followed:

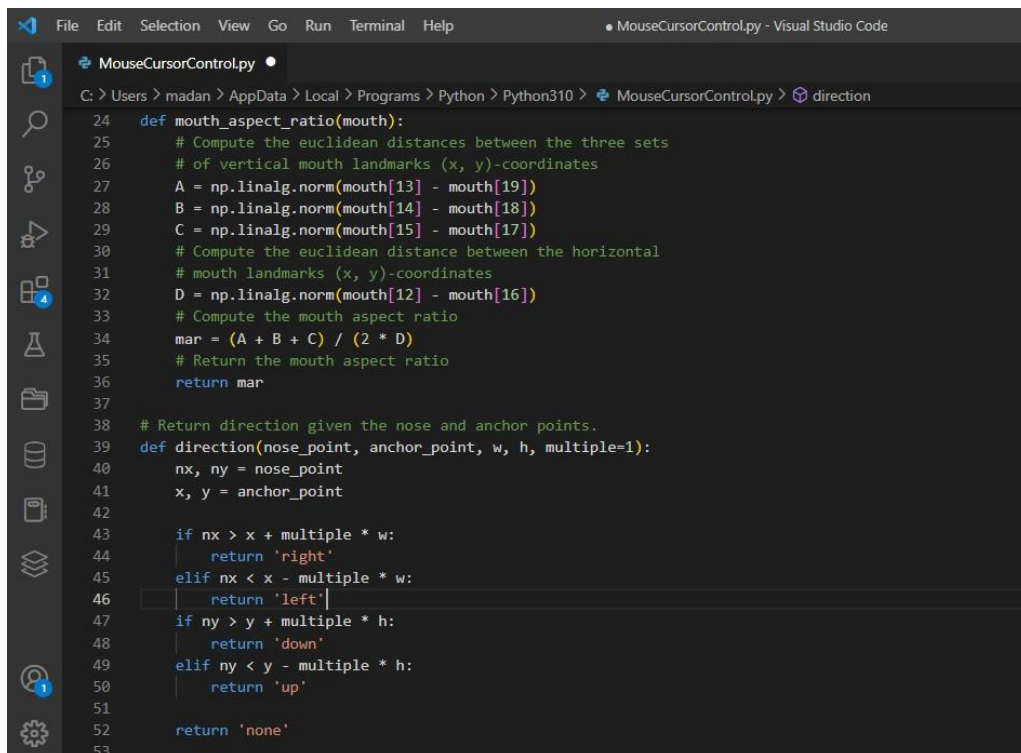
1. Install needed libraries:  
Numpy  
OpenCv  
PyAutoGui  
Dlib  
Imutils
2. Defining all the needed thresholds values, initializing counters of frame, and frame length driving mouse action.
3. From HOG face sensor we obtain facial corners.
4. Gain the indicators of facial milestones for mouth both the eyes and nose.
5. From the face in the image frame is converted into grayscale format.
6. Storing the facial marks co-ordinates (x, y) in the NumPy array.
7. Calculating Eye Aspect rate and Mouth Aspect Ratio.
8. Exactly from the centre of the nose, a bounding box is drawn.
9. Within that bounding box, one can perform the needed funtions.

## 7.1 Code:



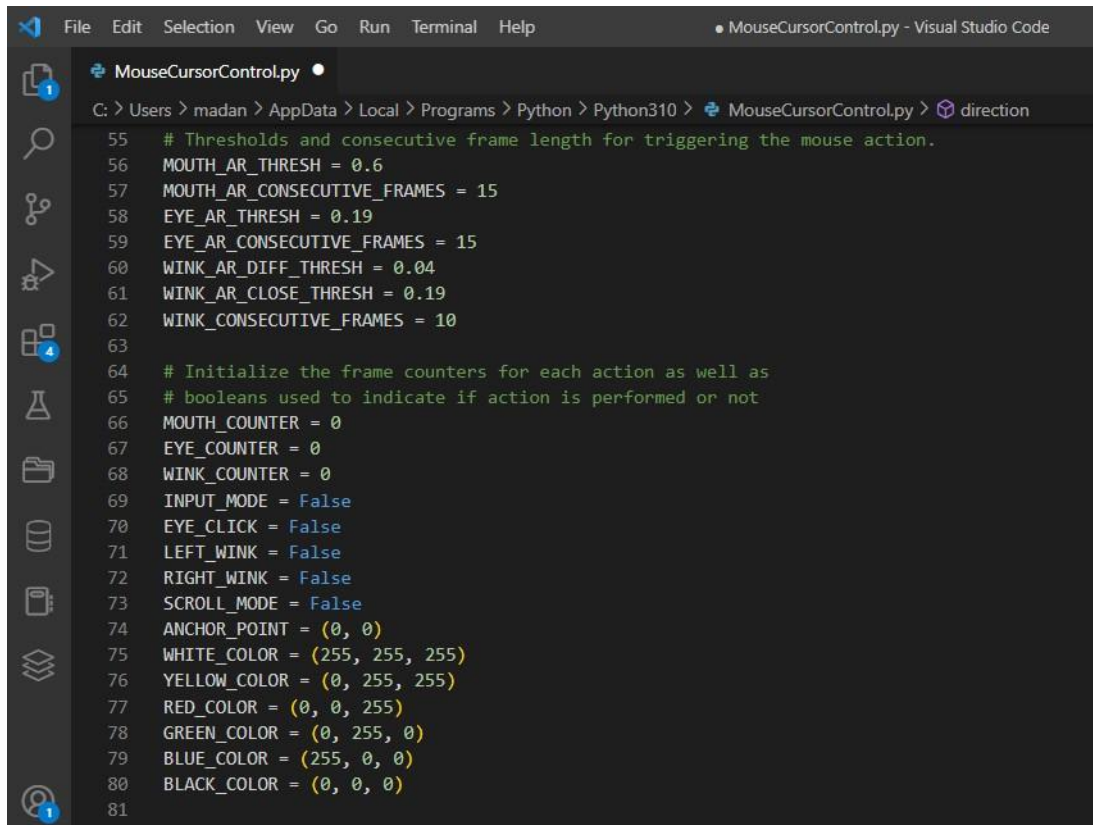
```
File Edit Selection View Go Run Terminal Help
MouseCursorControl.py
C: > Users > madan > AppData > Local > Programs > Python > Python310 > MouseCursorControl.py > mouth_aspect_ratio
1 from imutils import face_utils
2 from utils import *
3 import numpy as np
4 import pyautogui as pyag
5 import imutils
6 import dlib
7 import cv2
8
9 # Returns EAR given eye landmarks
10 def eye_aspect_ratio(eye):
11     # Compute the euclidean distances between the two sets of
12     # vertical eye landmarks (x, y)-coordinates
13     A = np.linalg.norm(eye[1] - eye[5])
14     B = np.linalg.norm(eye[2] - eye[4])
15     # Compute the euclidean distance between the horizontal
16     # eye landmark (x, y)-coordinates
17     C = np.linalg.norm(eye[0] - eye[3])
18     # Compute the eye aspect ratio
19     ear = (A + B) / (2.0 * C)
20     # Return the eye aspect ratio
21     return ear
22 # Returns MAR given eye landmarks
23
24 def mouth_aspect_ratio(mouth):
25     # Compute the euclidean distances between the three sets
26     # of vertical mouth landmarks (x, y)-coordinates
27     A = np.linalg.norm(mouth[13] - mouth[19])
28     B = np.linalg.norm(mouth[14] - mouth[18])
29     C = np.linalg.norm(mouth[15] - mouth[17])
30     # Compute the euclidean distance between the horizontal
```

Fig. 7.1.1



```
File Edit Selection View Go Run Terminal Help
MouseCursorControl.py
C: > Users > madan > AppData > Local > Programs > Python > Python310 > MouseCursorControl.py > direction
24 def mouth_aspect_ratio(mouth):
25     # Compute the euclidean distances between the three sets
26     # of vertical mouth landmarks (x, y)-coordinates
27     A = np.linalg.norm(mouth[13] - mouth[19])
28     B = np.linalg.norm(mouth[14] - mouth[18])
29     C = np.linalg.norm(mouth[15] - mouth[17])
30     # Compute the euclidean distance between the horizontal
31     # mouth landmarks (x, y)-coordinates
32     D = np.linalg.norm(mouth[12] - mouth[16])
33     # Compute the mouth aspect ratio
34     mar = (A + B + C) / (2 * D)
35     # Return the mouth aspect ratio
36     return mar
37
38 # Return direction given the nose and anchor points.
39 def direction(nose_point, anchor_point, w, h, multiple=1):
40     nx, ny = nose_point
41     x, y = anchor_point
42
43     if nx > x + multiple * w:
44         return 'right'
45     elif nx < x - multiple * w:
46         return 'left'
47     if ny > y + multiple * h:
48         return 'down'
49     elif ny < y - multiple * h:
50         return 'up'
51
52     return 'none'
53
```

Fig. 7.1.2



```

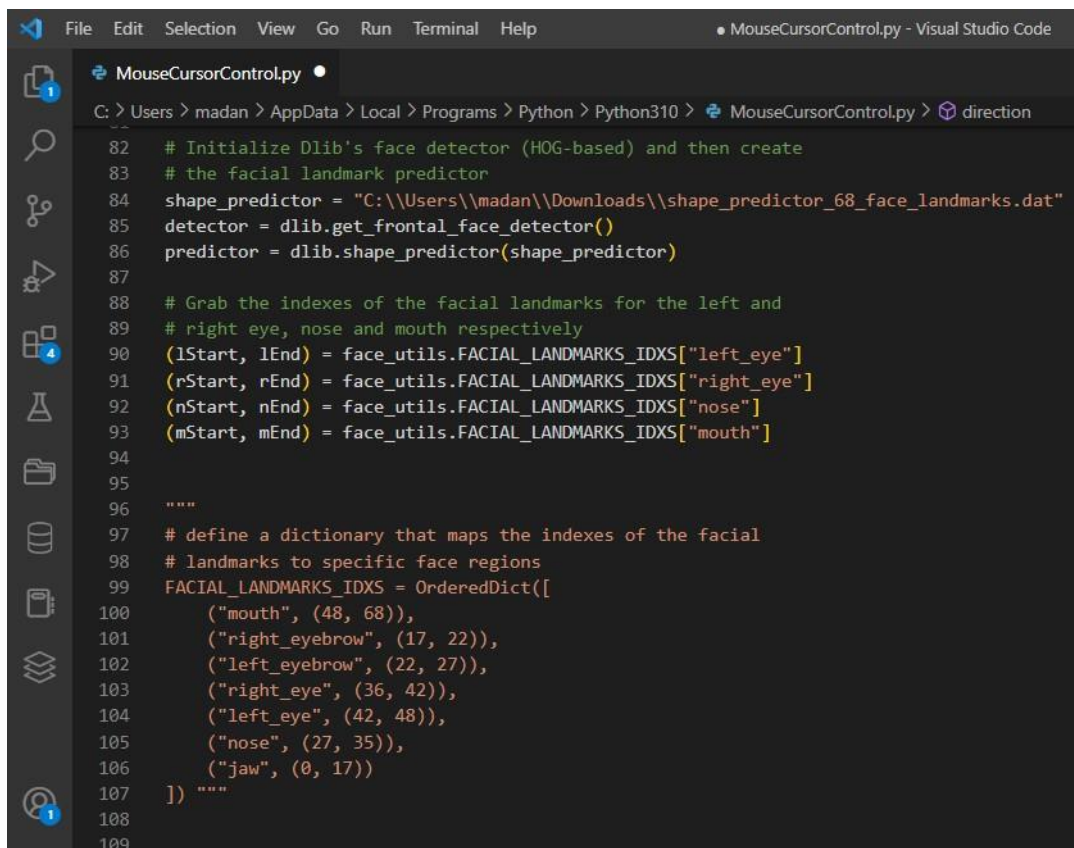
File Edit Selection View Go Run Terminal Help
• MouseCursorControl.py - Visual Studio Code

C: > Users > madan > AppData > Local > Programs > Python > Python310 > MouseCursorControl.py > direction

55 # Thresholds and consecutive frame length for triggering the mouse action.
56 MOUTH_AR_THRESH = 0.6
57 MOUTH_AR_CONSECUTIVE_FRAMES = 15
58 EYE_AR_THRESH = 0.19
59 EYE_AR_CONSECUTIVE_FRAMES = 15
60 WINK_AR_DIFF_THRESH = 0.04
61 WINK_AR_CLOSE_THRESH = 0.19
62 WINK_CONSECUTIVE_FRAMES = 10
63
64 # Initialize the frame counters for each action as well as
65 # booleans used to indicate if action is performed or not
66 MOUTH_COUNTER = 0
67 EYE_COUNTER = 0
68 WINK_COUNTER = 0
69 INPUT_MODE = False
70 EYE_CLICK = False
71 LEFT_WINK = False
72 RIGHT_WINK = False
73 SCROLL_MODE = False
74 ANCHOR_POINT = (0, 0)
75 WHITE_COLOR = (255, 255, 255)
76 YELLOW_COLOR = (0, 255, 255)
77 RED_COLOR = (0, 0, 255)
78 GREEN_COLOR = (0, 255, 0)
79 BLUE_COLOR = (255, 0, 0)
80 BLACK_COLOR = (0, 0, 0)
81

```

Fig. 7.1.3



```

File Edit Selection View Go Run Terminal Help
• MouseCursorControl.py - Visual Studio Code

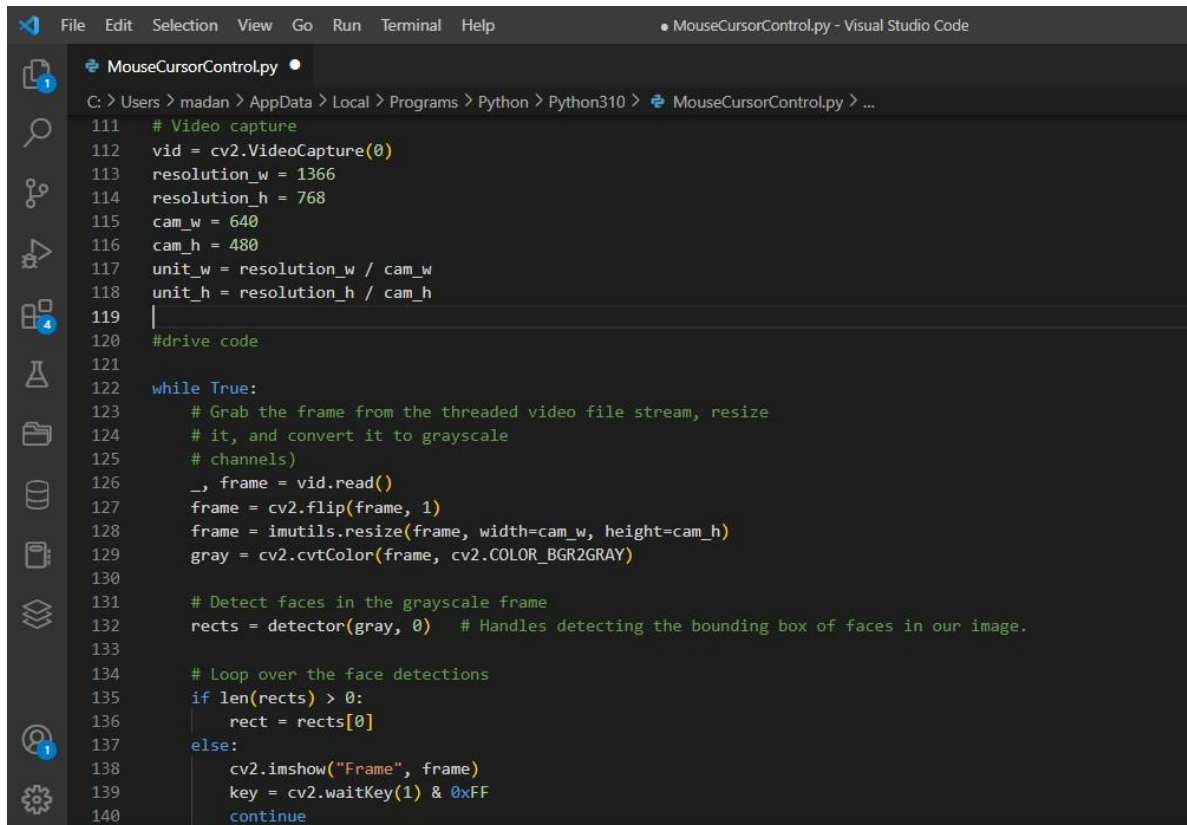
C: > Users > madan > AppData > Local > Programs > Python > Python310 > MouseCursorControl.py > direction

82 # Initialize Dlib's face detector (HOG-based) and then create
83 # the facial landmark predictor
84 shape_predictor = "C:\\Users\\madan\\Downloads\\shape_predictor_68_face_landmarks.dat"
85 detector = dlib.get_frontal_face_detector()
86 predictor = dlib.shape_predictor(shape_predictor)
87
88 # Grab the indexes of the facial landmarks for the left and
89 # right eye, nose and mouth respectively
90 (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
91 (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
92 (nStart, nEnd) = face_utils.FACIAL_LANDMARKS_IDXS["nose"]
93 (mStart, mEnd) = face_utils.FACIAL_LANDMARKS_IDXS["mouth"]
94
95
96 """
97 # define a dictionary that maps the indexes of the facial
98 # landmarks to specific face regions
99 FACIAL_LANDMARKS_IDXS = OrderedDict([
100     ("mouth", (48, 68)),
101     ("right_eyebrow", (17, 22)),
102     ("left_eyebrow", (22, 27)),
103     ("right_eye", (36, 42)),
104     ("left_eye", (42, 48)),
105     ("nose", (27, 35)),
106     ("jaw", (0, 17))
107 ]) """
108
109

```

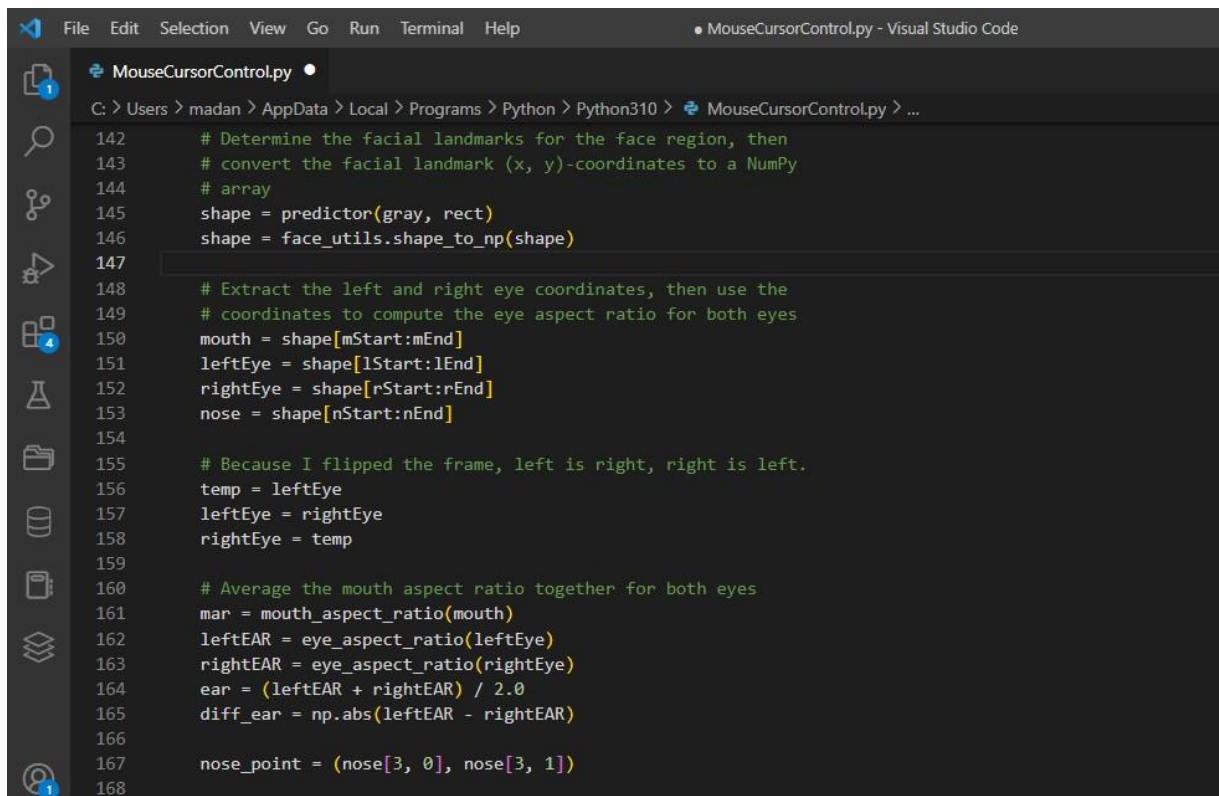
Fig. 7.1.2





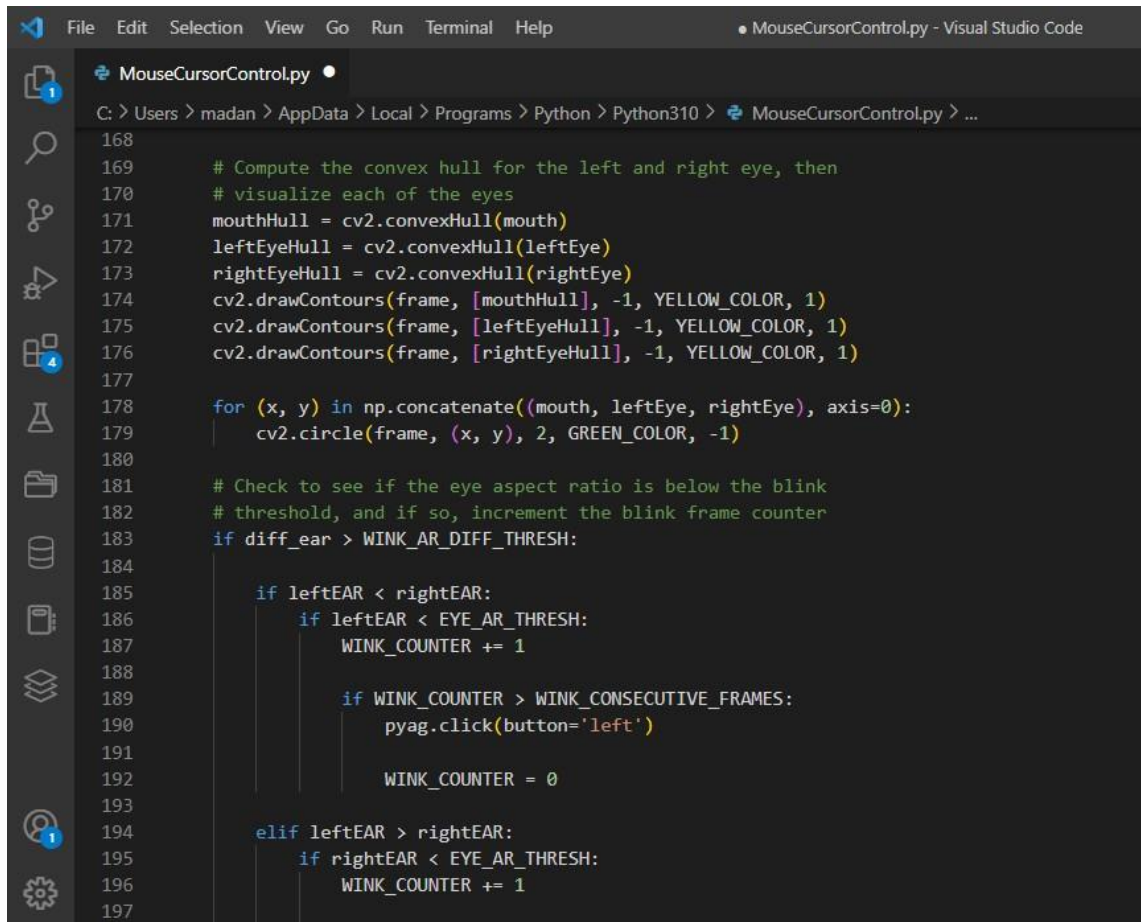
```
111 # Video capture
112 vid = cv2.VideoCapture(0)
113 resolution_w = 1366
114 resolution_h = 768
115 cam_w = 640
116 cam_h = 480
117 unit_w = resolution_w / cam_w
118 unit_h = resolution_h / cam_h
119 |
120 #drive code
121
122 while True:
123     # Grab the frame from the threaded video file stream, resize
124     # it, and convert it to grayscale
125     # channels)
126     _, frame = vid.read()
127     frame = cv2.flip(frame, 1)
128     frame = imutils.resize(frame, width=cam_w, height=cam_h)
129     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
130
131     # Detect faces in the grayscale frame
132     rects = detector(gray, 0) # Handles detecting the bounding box of faces in our image.
133
134     # Loop over the face detections
135     if len(rects) > 0:
136         rect = rects[0]
137     else:
138         cv2.imshow("Frame", frame)
139         key = cv2.waitKey(1) & 0xFF
140         continue
```

Fig. 7.1.5



```
142 # Determine the facial landmarks for the face region, then
143 # convert the facial landmark (x, y)-coordinates to a NumPy
144 # array
145 shape = predictor(gray, rect)
146 shape = face_utils.shape_to_np(shape)
147
148 # Extract the left and right eye coordinates, then use the
149 # coordinates to compute the eye aspect ratio for both eyes
150 mouth = shape[mStart:mEnd]
151 leftEye = shape[lStart:lEnd]
152 rightEye = shape[rStart:rEnd]
153 nose = shape[nStart:nEnd]
154
155 # Because I flipped the frame, left is right, right is left.
156 temp = leftEye
157 leftEye = rightEye
158 rightEye = temp
159
160 # Average the mouth aspect ratio together for both eyes
161 mar = mouth_aspect_ratio(mouth)
162 leftEAR = eye_aspect_ratio(leftEye)
163 rightEAR = eye_aspect_ratio(rightEye)
164 ear = (leftEAR + rightEAR) / 2.0
165 diff_ear = np.abs(leftEAR - rightEAR)
166
167 nose_point = (nose[3, 0], nose[3, 1])
168
```

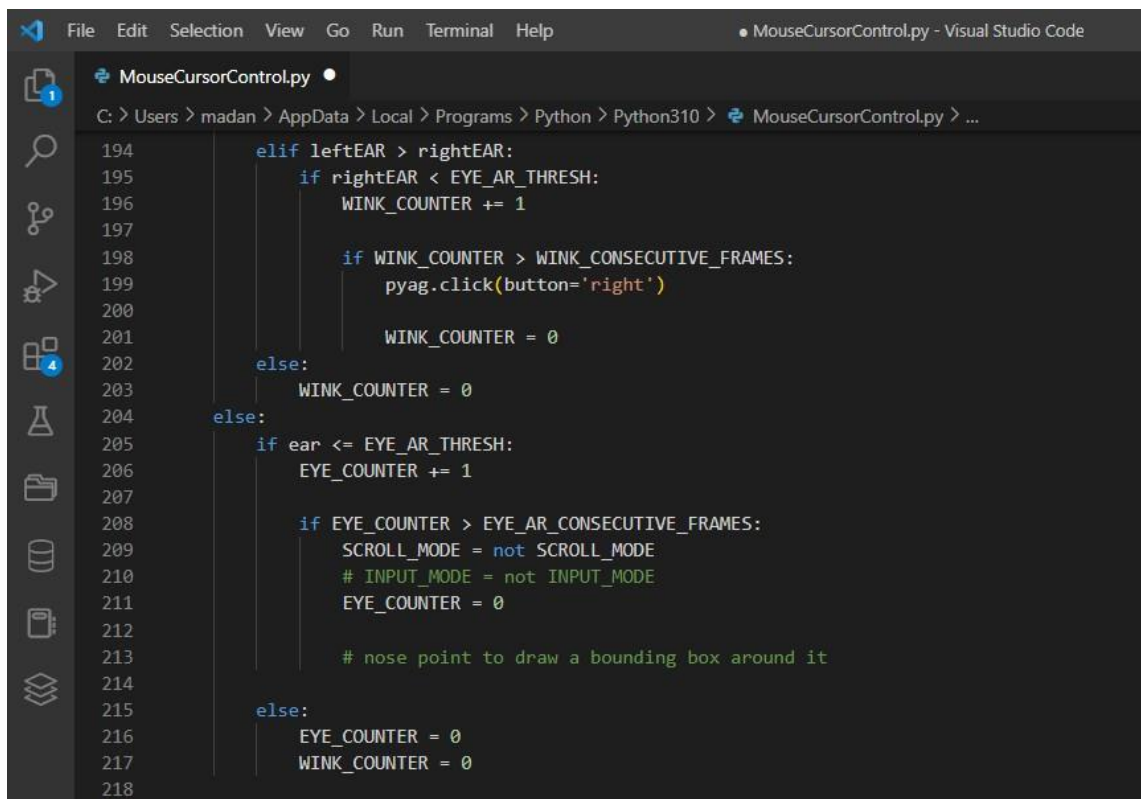
Fig. 7.1.6



```
File Edit Selection View Go Run Terminal Help
MouseCursorControl.py
C: > Users > madan > AppData > Local > Programs > Python > Python310 > MouseCursorControl.py > ...

168
169     # Compute the convex hull for the left and right eye, then
170     # visualize each of the eyes
171     mouthHull = cv2.convexHull(mouth)
172     leftEyeHull = cv2.convexHull(leftEye)
173     rightEyeHull = cv2.convexHull(rightEye)
174     cv2.drawContours(frame, [mouthHull], -1, YELLOW_COLOR, 1)
175     cv2.drawContours(frame, [leftEyeHull], -1, YELLOW_COLOR, 1)
176     cv2.drawContours(frame, [rightEyeHull], -1, YELLOW_COLOR, 1)
177
178     for (x, y) in np.concatenate((mouth, leftEye, rightEye), axis=0):
179         cv2.circle(frame, (x, y), 2, GREEN_COLOR, -1)
180
181     # Check to see if the eye aspect ratio is below the blink
182     # threshold, and if so, increment the blink frame counter
183     if diff_ear > WINK_AR_DIFF_THRESH:
184
185         if leftEAR < rightEAR:
186             if leftEAR < EYE_AR_THRESH:
187                 WINK_COUNTER += 1
188
189             if WINK_COUNTER > WINK_CONSECUTIVE_FRAMES:
190                 pyag.click(button='left')
191
192                 WINK_COUNTER = 0
193
194         elif leftEAR > rightEAR:
195             if rightEAR < EYE_AR_THRESH:
196                 WINK_COUNTER += 1
197
```

Fig. 7.1.7



```
File Edit Selection View Go Run Terminal Help
MouseCursorControl.py
C: > Users > madan > AppData > Local > Programs > Python > Python310 > MouseCursorControl.py > ...

194     elif leftEAR > rightEAR:
195         if rightEAR < EYE_AR_THRESH:
196             WINK_COUNTER += 1
197
198             if WINK_COUNTER > WINK_CONSECUTIVE_FRAMES:
199                 pyag.click(button='right')
200
201                 WINK_COUNTER = 0
202         else:
203             WINK_COUNTER = 0
204     else:
205         if ear <= EYE_AR_THRESH:
206             EYE_COUNTER += 1
207
208             if EYE_COUNTER > EYE_AR_CONSECUTIVE_FRAMES:
209                 SCROLL_MODE = not SCROLL_MODE
210                 # INPUT_MODE = not INPUT_MODE
211                 EYE_COUNTER = 0
212
213                 # nose point to draw a bounding box around it
214
215         else:
216             EYE_COUNTER = 0
217             WINK_COUNTER = 0
218
```

Fig. 7.1.8



```

219     if mar > MOUTH_AR_THRESH:
220         MOUTH_COUNTER += 1
221
222         if MOUTH_COUNTER >= MOUTH_AR_CONSECUTIVE_FRAMES:
223             # if the alarm is not on, turn it on
224             INPUT_MODE = not INPUT_MODE
225             # SCROLL_MODE = not SCROLL_MODE
226             MOUTH_COUNTER = 0
227             ANCHOR_POINT = nose_point
228
229     else:
230         MOUTH_COUNTER = 0
231
232     if INPUT_MODE:
233         cv2.putText(frame, "READING INPUT!", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, RED_COLOR, 2)
234         x, y = ANCHOR_POINT
235         nx, ny = nose_point
236         w, h = 60, 35
237         multiple = 1
238         cv2.rectangle(frame, (x - w, y - h), (x + w, y + h), GREEN_COLOR, 2)
239         cv2.line(frame, ANCHOR_POINT, nose_point, BLUE_COLOR, 2)
240
241         dir = direction(nose_point, ANCHOR_POINT, w, h)
242         cv2.putText(frame, dir.upper(), (10, 90), cv2.FONT_HERSHEY_SIMPLEX, 0.7, RED_COLOR, 2)
243         drag = 18

```

Fig. 7.1.9

```

244     if dir == 'right':
245         pyag.moveRel(drag, 0)
246     elif dir == 'left':
247         pyag.moveRel(-drag, 0)
248     elif dir == 'up':
249         if SCROLL_MODE:
250             pyag.scroll(40)
251         else:
252             pyag.moveRel(0, -drag)
253     elif dir == 'down':
254         if SCROLL_MODE:
255             pyag.scroll(-40)
256         else:
257             pyag.moveRel(0, drag)
258
259     if SCROLL_MODE:
260         cv2.putText(frame, 'SCROLL MODE IS ON!', (10, 60), cv2.FONT_HERSHEY_SIMPLEX, 0.7, RED_COLOR, 2)
261
262     # Show the frame
263     cv2.imshow("Frame", frame)
264     key = cv2.waitKey(1) & 0xFF
265
266     # If the `Esc` key was pressed, break from the loop
267     if key == 27:
268         break
269
270     # Do a bit of cleanup
271     cv2.destroyAllWindows()
272     vid.release()

```

Fig. 7.1.10

Above Fig. 7.1.1 – Fig. 7.1.10 illustrates the code implementation of the application

## 7.2 OUTPUT:

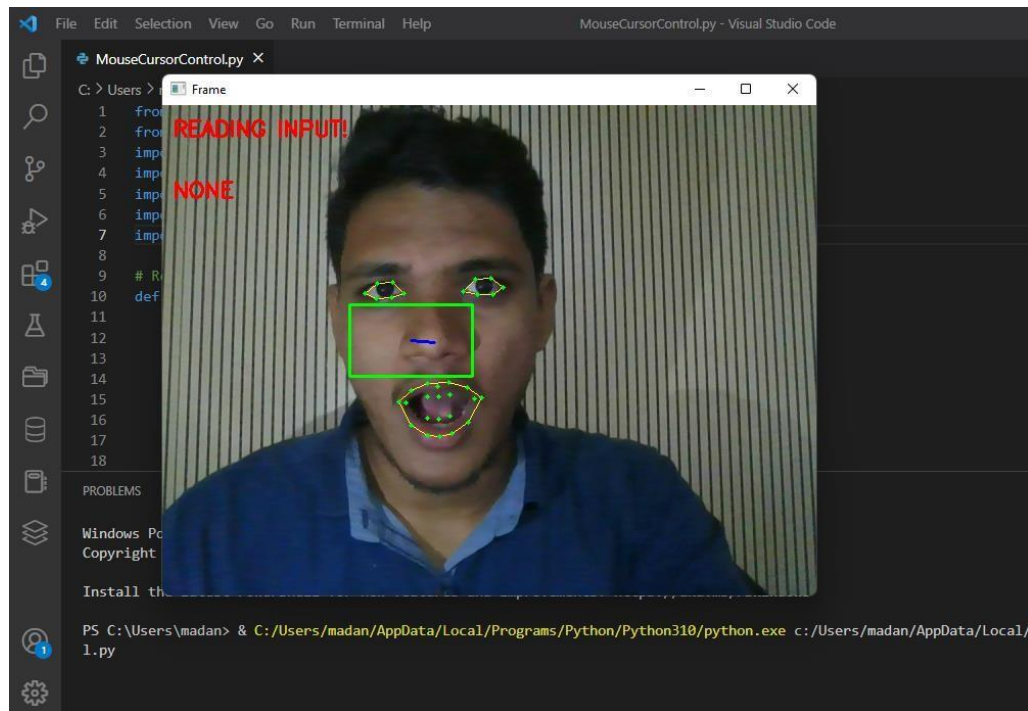


Fig 7.2.1

Fig 7.2.2 illustrates the initiation of the application; one has to open his mouth to turn on to read the input mode.

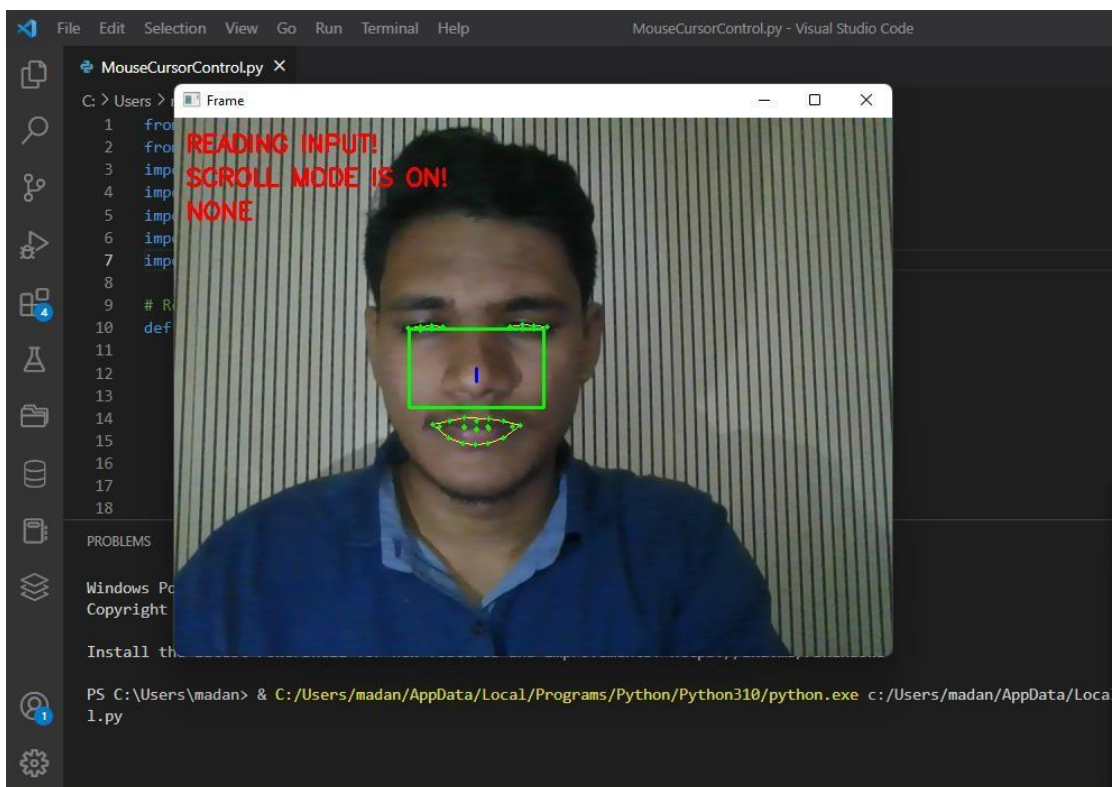


Fig. 7.2.2

Fig 7.2.2 illustrates the action of Squinting of eyes to turn on the scrolling mode on

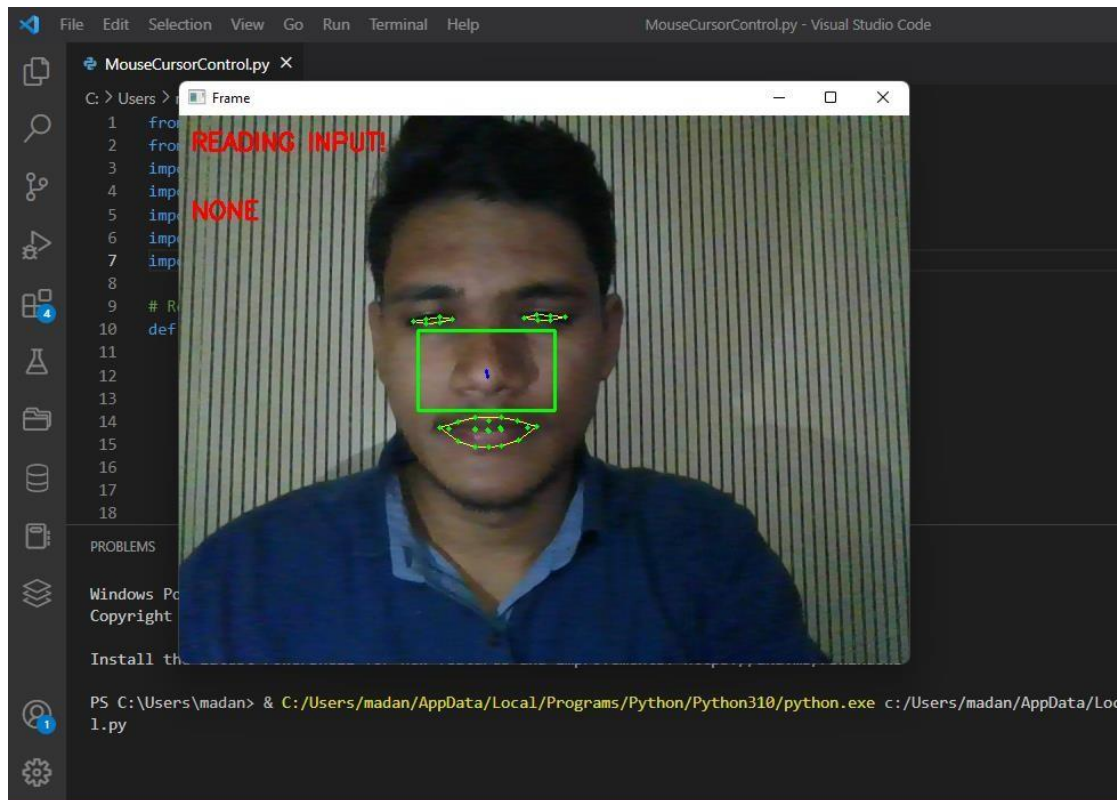


Fig 7.2.3

Fig 7.2.3 illustrates the action of Squinting of both eyes again to turn off the scrolling mode

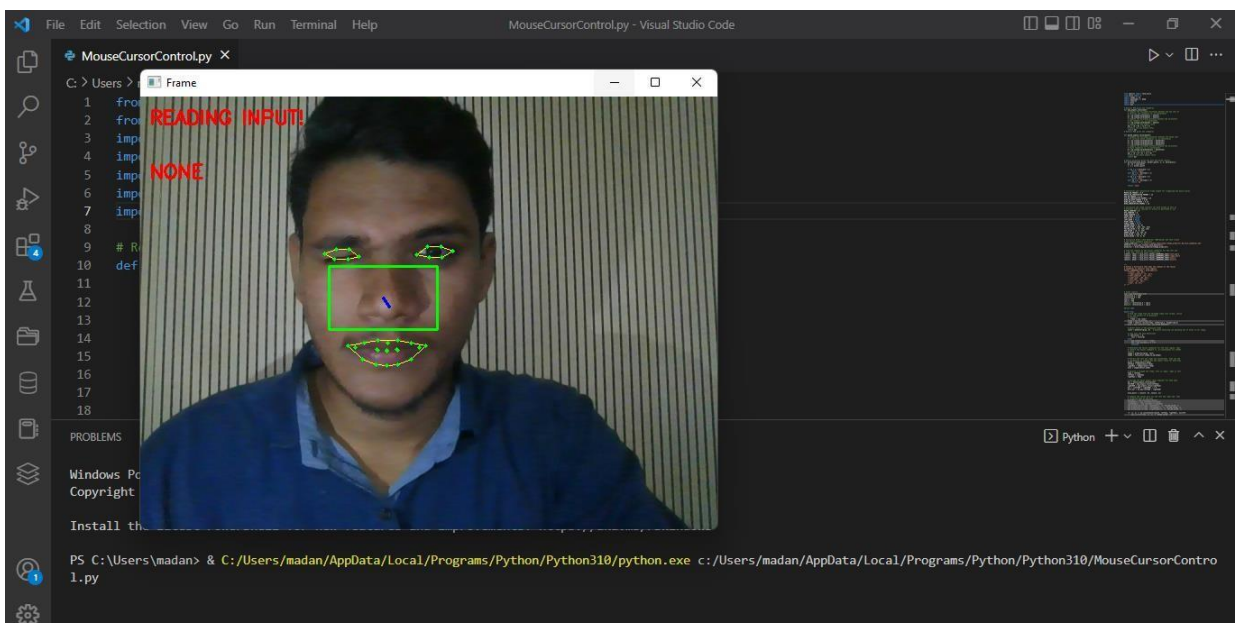


Fig 7.2.3

Fig 7.2.3 illustrates the action of blinking of any of the eye to perform the click action either right or left



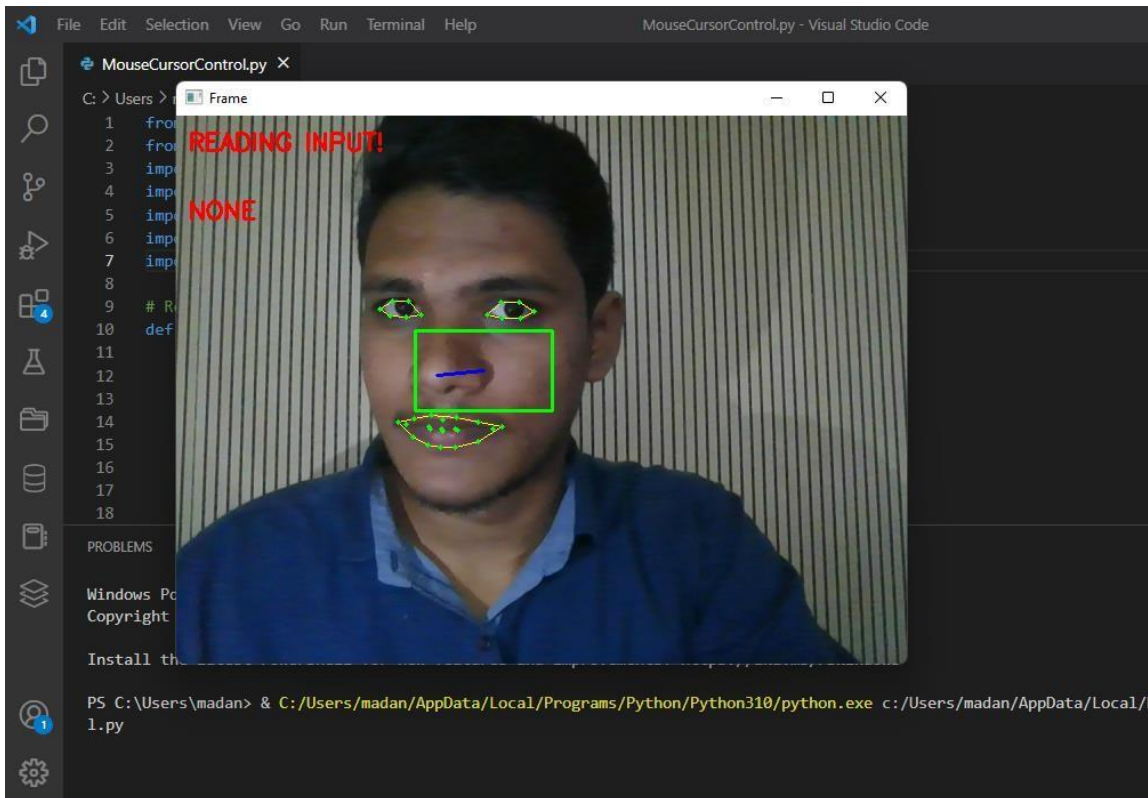


Fig 7.2.5

Fig 7.2.5 illustrates action yawing of face triggers mouse cursor to move back and forth.

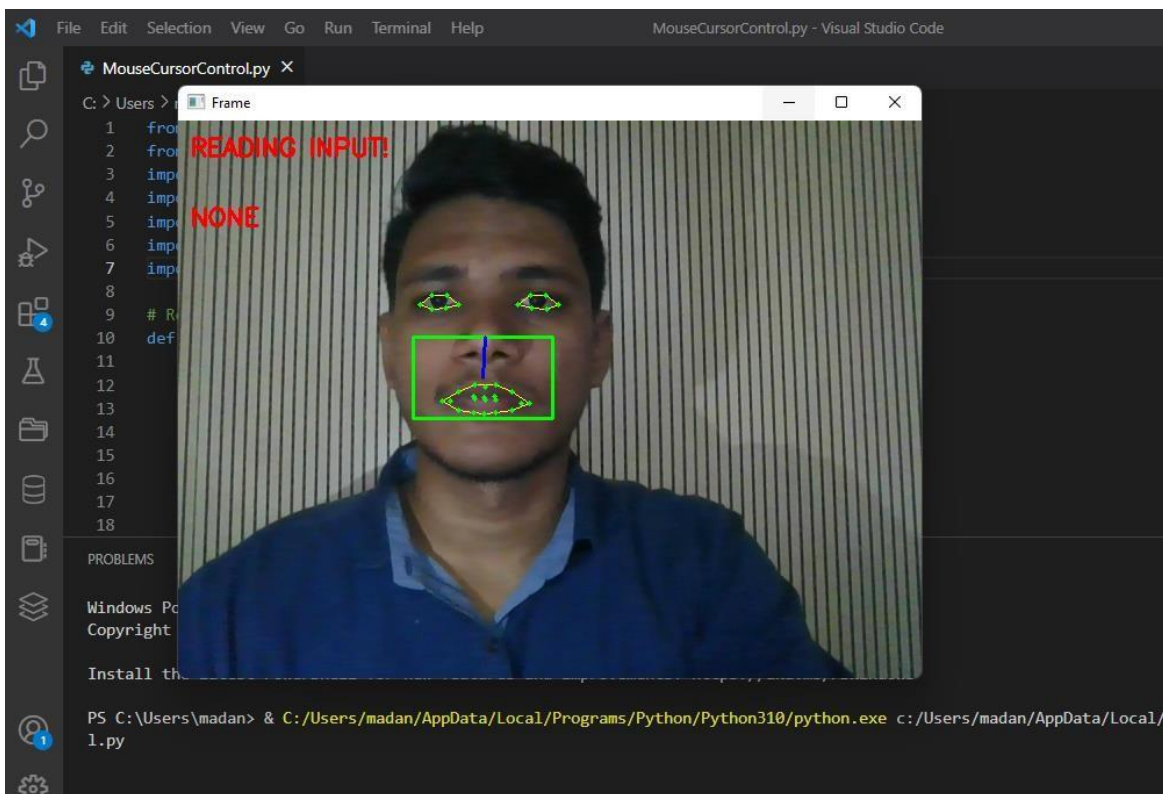
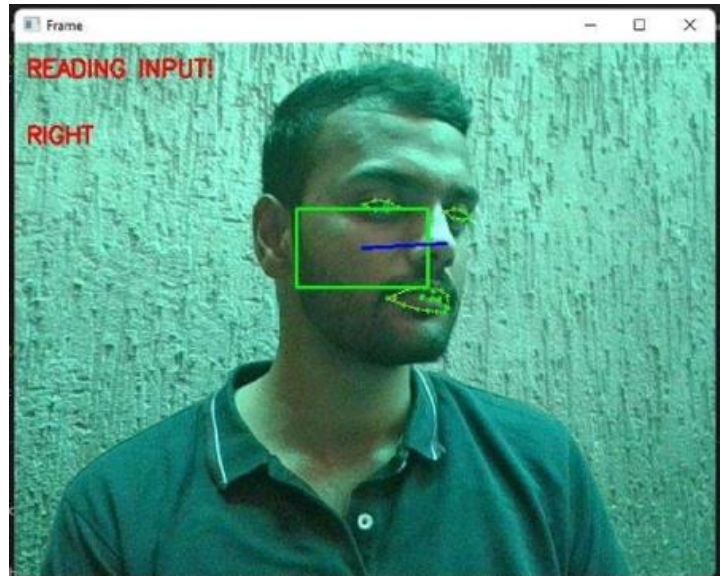
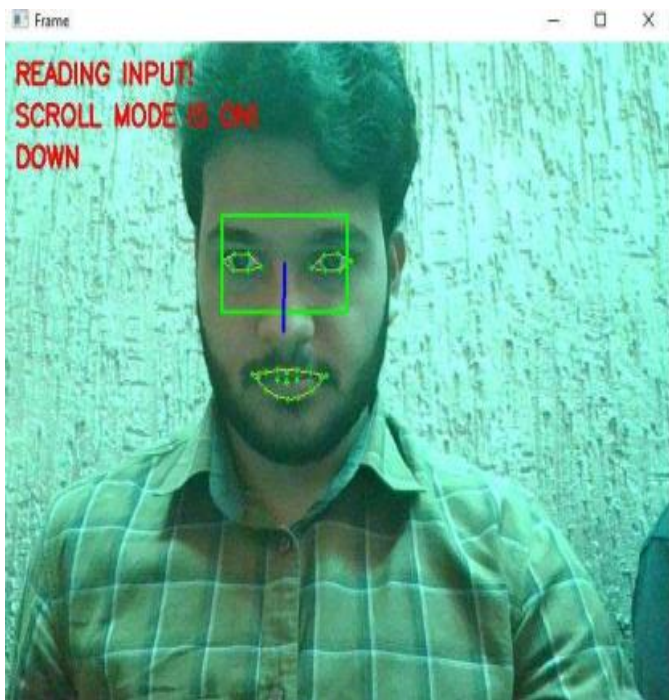


Fig 7.2.6

Fig 7.2.6 illustrates the action of pitching of head triggers the cursor to move Up and Down



7.2.7 Illustrates the action of Left, Right movements of the cursor.



7.2.8 Illustrates the action of Down, up scroll movements of cursor.



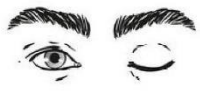

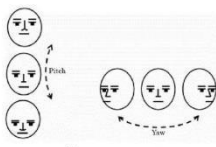
## 8. RESULTS AND DISCUSSION

The main agenda of this work was to increase the understanding, responsiveness and interaction towards machines along with human behaviour. Creating a technology that is portable, inexpensive, and compatible with any normal operating system was the main goal of this article.

The proposed work is to control the mouse pointer by capturing human Face and calculating the eyes and nose positions. While, the moving of pointer will be in the same direction with respect to the Head Movement. the mechanism Control basic mouse actions including moving in all directions, left-clicking, right-clicking and scrolling.

Opening the mouth activates the cursor and movement of the pointer can be controlled by gazing our head. The clicking events is performed by making winks with the eyes i.e., for left-clicking, doing a wink with left eye and for right-clicking, doing a wink with right eye and by squinting the eyes we can activated and start scrolling and again squinting we can deactivate.

Table 8.1 Action and Function pairs

Action	Function
 Opening Mouth	Activate / Deactivate Mouse Control
 Right Eye Wink	Right Click
 Left Eye Wink	Left Click
 Squinting Eyes	Activate / Deactivate Scrolling
 Head Movements (Pitch and Yaw)	Scrolling / Cursor Movement



## **9. CONCLUSION AND FUTURE SCOPE**

We intend to improve the software's functionality, particularly tracking of Head movement which is the main event for cursor movement in the near future. In order to fully replace our traditional mouse, we also want to speed up the software's response time for pointer movement. Mouse operations like dragging, zoom in, zoom out and for typing using voice recognition and text extraction are to be implemented.

We also plan to design an interface where can implement this project in real life applications. The interface includes a user who is physically handicapped can operate a PC/laptop by his movements and facial expressions without any other's help. This work helps them a lot in their life whose goals are towards technology. Handicapped people need not to be dependent on other's for operating a PC/laptop.

## 10. REFERENCES

- [1] Shashidhar R et al. "Mouse Cursor Control Using Facial Movements -An HCL Application" 2022 International Conference on Sustainable Computing and Data Communication System (ICSCDS). IEEE, 2022.
- [2] Pavithra's et al. "Cursor Movement Control using Eyes and Facial Movements for Physically Challenged People." 2021 International Research Journal of Engineering and Technology (IRJET). e-ISSN: 2395-0056, p-ISSN: 2395-0072
- [3] Satyanarayana B., et al. "Eyeball Movement based Cursor Control using Raspberry Pi and OpenCV." 2020 International Journal of Innovative Technology and Exploring Engineering (IJITEE). ISSN: 2278-3075
- [4] Vasanthan M et al. "Face Expression Based computer cursor control system for assisting physically disabled person" ResearchGate Publications, COMNESAT 2012, IEEE 2012.
- [5] Shrunkhala Satish Wankhede, et al. " CONTROLLING MOUSE CURSOR USING EYE MOVEMENT." 2013 International Journal of Application or Innovation in Engineering & Management (IJAIEEM), ISSN 2319 - 4847 Special Issue for National Conference on Recent Advances in Technology and Management for Integrated Growth (RATMIG)
- [6] Margi Sadashiv Narayan, et al. " ENHANCED CURSOR CONTROL USING EYE MOUSE." Dec-2016 International Journal of Advances in Electronics and Computer Science, ISSN: 2393-2835.
- [7] Sovit Ranjan Rath, et al. "Machine Learning and Deep Learning, Face detection with Dlib using HOG and Linear SVM"