MOUSE CURSOR & KEYBOARD CONTROL USING FACIAL MOVEMENTS

Prepared For:

Artificial Intelligence (CSE3013) – J Component

TEAM MEMBERS:

Abhishek Agarwal - 20BDS0070

Aditya Avinash Namjoshi - 20BCE0437

Atharv Mahesh Gote - 20BCT0151

Patel Viraj Shaileshkumar - 20BCE2503

Zeel Rasikbhai Lukhi - 20BCI0206

Submitted to:
Mr. DILIPKUMAR S.

ABSTRACT

To create software that will allow the user to perform mouse cursor control with facial movement with just a regular webcam. It will be hands-free, and also cuts down the cost of wearable hardware or sensors for mouse cursor control.

KEYWORDS

- Mouse control
- Keyboard control
- Facial movements
- Hands-free device
- Face recognition
- Eye blink
- Touchless virtual keyboard
- Introduction

INTRODUCTION

Problem statement

People without hands can’t use electronic gadgets. Face trackers with eye gaze communication have been proposed as a possible communication portal by researchers in the field. Some portals as such have been introduced, but they are quite expensive. In psychology, marketing, and user interfaces, Face trackers have become increasingly essential. They have been around for a while, but their use was mostly limited to laboratory investigations to examine the nature of human eye movements, and were too expensive to consider using in a genuine user-human interface a decade ago.

Motivation

The number of assistive gadgets controlled by computers is growing every day. Physically challenged people can utilize these assistive devices to interact with a variety of applications such as communication with others, education, and entertainment. Vision-based systems have become more popular in recent years for creating and implementing various types of applications. The cursor movement in a computer application is controlled by the movement of lit markers using facial expression in this work. Face detection, eye extraction, and voice recognition are among the image processing techniques used. It captures an input image with a standard webcam. A computer would also be simple to use and learn for a physically challenged individual, which is another reason for constructing this project.

Objectives

In the recent decade, there has been a huge growth in the technology sector. Mobile phones, tablets, and even laptops with Touch screens have been introduced and widely accepted. Eye gaze has been proposed as a possible communication portal by researchers in the field. Some portals as such have been introduced, but they are quite expensive. This system can make the lives of physically challenged individuals significantly easy by giving them access to computers without having to use their limbs. Scope and applications. The system can be used to access a computer without the usage of limbs - from eye movement alone. This can be used by physically challenged individuals. The scope of this paper includes reviewing existing work on the topic, developing and testing a system with a main interface, Mouse/keyboard simulation engine, User action detection module, Halt (sleep) module, mouse and keyboard function modules.

Techniques Used

The mouse cursor can be controlled by facial movements such as moving up and down or left and right, blinking and/or voice. The keyboard can be controlled via keys displayed on the computer screen which get selected when the visual finger is "virtually" inside the area which corresponds to a particular key. A webcam with low resolution attached to a wearable glass frame can be utilized for the same. The iris center is located using the images generated from the webcam and the movement of iris is coordinated to the movement of the mouse pointer.

Literature Review

Ref. No.	Paper Title	Journal name and year of publication	Work done	Techniques used	Gaps found
1	Eye Gaze controlled virtual keyboard	International Journal of Recent Technology and .Engineering (IJRTE), 2019	Developed a virtual keyboard that works by detecting eye gaze and eye Blinking using video captured directly from a PC camera.	The system uses an approach that involves the 68 points of the face which is specific and must exist in every face is used	A maximum of 100 words only can be written using the keyboard. Results are not accurate for users wearing glasses.
2	Real-time eye blink detection using facial landmarks	Centre for Machine Perception, Department of Cybernetics Faculty of Electrical Engineering, Czech Technical University in Prague, 2016	Presented a literature review of driver drowsiness detection based on behavioural measures using machine learning techniques	Estimated the landmark positions, extracted a single scalar quantity eye aspect ratio and characterized the eye-opening in each frame. Finally, an SVM classifier detects eye blinks as a pattern of EAR values in a short temporal window	A fixed blink duration for all subjects was assumed, although everyone's blink lasts differently.
3	Touchless virtual keyboard controlled by eye blinking and EEG signals	Man-Machine Interactions 5.ICMMI 2017. Advances in Intelligent Systems and Computing,2017	Developed a virtual keyboard where each key can be selected by three double-eye blinks registered by EMG sensor	An online A desktop-based tool was created on which EEG signals are used as support that allows the user to change the input mode of single characters to the mode of predicted word selection.	Obtained result (WPM= 1.11) only partially confirms the calculated typing speed (WPM= 1.23)

4	A 3D Approach to Facial Landmarks: Detection, Refinement, and Tracking	ICPR 2014, IEEE 2014	A real-time an algorithm for accurate localization of facial landmarks in a single monocular image is proposed.	The algorithm is formulated as an optimization problem, in which the sum of responses of local classifiers is maximized concerning the camera pose. The algorithm simultaneously estimates a head position and orientation and detects the facial landmarks in the image.	None were identified
5	Eye Tracking Based Control System for Natural Human-Computer Interaction	Hindawi Computational Intelligence and Neuroscience,2017	Developed an eye-tracking-based control system for user-computer dialogue which combines both mouse functions and keyboard functions.	A fixation function is defined to calculate the gaze time of the user and determines the mouse coordinates directed by eye movement. The system then performs directly from the viewpoint of the user.	None were identified
6	Real-Time Eye Tracking and Blink Detection with USB Cameras	Boston University Computer Science Technical Report No. 2005-12	A a human-computer interface (HCI) system designed for use by people with severe disabilities are presented.	The algorithm used by the system for detecting and analysing blinks is initialized automatically, dependent only upon the inevitability of the involuntary blinking of the user.	The model does not yield accurate results for people wearing glasses.

7	Low-cost natural interface based on head movements	Procedia Computer Science, 2015	Developed a human-computer interface that allows for touchless computer control, avoiding the use of a keyboard, mouse, or touchscreen.	An interface that is based on the Kinect sensor, and allows computer control only by the movement of the head, eyebrows (up/down), and the mouth (opening/closing), in addition to available voice commands.	Voice commands were not covered in the initial application goal
8	Gesture Control Using Single Camera For PC	Procedia Computer Science, 2015	Proposed a basic model of integrating Face Recognition method with Hand Tracking where the authenticated user alone can access their system without using a mouse thereby providing access a privilege to the user.	The Face Recognition model uses Viola and Jones method for detection of the face and PCA (Principal Component Analysis) for recognition and identification of algorithms.	The application however fails clearly in darkness which is not the case with a real mouse, so the illumination is always provided.
9	An empirical evaluation of hands-free computer interaction for users with motor disabilities	Journal of Biomedical Informatics, 2016	To design and evaluate a universal solution for a hands-free HCI, based on the emotive EPOC+ device, which, among other capabilities, also enables controlling	The proposed solution is based on the Emotive EPOC+ device. By measuring and analysing EMG signals, facial expressions can be identified, which, in turn, can be used for the implementation	All participants had no experience in using the EPOC+ device, and everyone received the same level of training in using the device.

			the computer with facial expressions and motion sensors.	of computer commands.	
10	Locally Linear Regression for Pose-Invariant Face Recognition	IEEE Transactions on Image Processing, 2007	Proposed a novel locally linear regression (LLR) method, which generates the virtual frontal view from a given non-frontal face image	Formulated the estimation of the linear mapping as a prediction problem and presented the regression-based solution. To improve the prediction accuracy for coarse alignment, LLR is further proposed.	The pose of the input non-frontal face image is assumed to be known. This implies that one has to use a front-end procedure to estimate the pose of the input image.
11	Incremental Face Alignment in the Wild	Proceedings of the IEEE conference on computer vision and pattern recognition, 2014	proposed the possibility to automatically construct robust discriminative person and imaging condition-specific models in the wild that outperform state-of-the-art generic face alignment strategies.	The Par-CLR method is the foundation for the proposed incremental face alignment framework, which has an exact incremental solution per level.	The system is not real-time
12	A System Identification Approach for Video-based Face Recognition	Proceedings of the 17th International Conference on Pattern Recognition, 2004	Presented a structured approach to the problem of video-based face recognition that dealt with the problem of recognizing	A moving face is represented as a linear dynamical system whose appearance changes with time. Subspace angles-based distance metrics are used to get the measure of similarity	None identified

			faces when both gallery and probe consist of face videos	between ARMA models representing moving face sequences.	
13	Automatic 3D Face Reconstruction from Single Images or Video	8th IEEE International Conference on Automatic Face & Gesture Recognition, 2008	Presented a fully automated algorithm for reconstructin g g a textured 3D model of a face from a single photograph or a raw video stream.	The algorithm is based on a combination of Support Vector Machines (SVMs) and a Morphable Model of 3D faces.	Simple ambient illumination is assumed in the images or videos
14	Video-Based Face Recognition Using Adaptive Hidden Markov Models	IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 03), 2003	Proposed to use adaptive Hidden Markov Models (HMM) to perform video-based face recognition	During the recognition process, the temporal characteristics of the test video sequence are analysed over time by the HMM corresponding to each subject. The likelihood scores	Observation probabilities of Hidden Markov Models (HMM) are not taken into consideration.

Related Work

As we can see from the literature survey, Face Recognition is utilized in studying facial characteristics, storing features in a database, using them to discover users. It facilitates in encoding the relationship among the facial features, to discover structure features of face and set of photographs that capture the variants of faces that exist even if lightning situations vary. There are different approaches to recognize a face. But in this work, we extract Eigenvalues from a picture that is transformed to Eigenfaces which consists of Eigen features. The advantage of extracting Eigenvalues is that it offers much more correct outcomes than every other technique. This technique is known as Viola and Jones which is called after them. The advantage of using Viola and Jones is that it no longer requires a complete frontal pose of a face. It makes use of cascading items in place of measuring the ratio of eyes and nose.

Gaps identified

As shown in the literature survey, certain models depicted inaccuracy while dealing with users wearing glasses- this occurred due to the glare caused and while working in darkness. Along with that, certain models had word limitations (up to 100 words only) and took generalized assumptions which may or may not work for all users (fixed blink duration).

Existing Algorithm

- The faces are detected using a Haar Cascade Classifier on an image in conjunction with the cropping of the cardinal section of the face.
- Thresholding is use for image segmentation which is a way to create a binary image from a grayscale or full-color image used to separate object like foreground pixels from background pixels to aid in image processing.
- Mouse cursor control are also done by RGB-D images and fingertip detection using K-cosine Corner Detection algorithm and Moore-Neighbour algorithm.

Disadvantages

- It gives some false result in case of eye gaze detection of those person who wearing a glasses because of reflection of the light in the glass, it cannot detect eye ball accurately.
- This system also has got some limitation due to its environment and lacking of features.
- We can write only 100 word by using this technique, so we can not use this system to write big notes.
- The fingertip detection algorithm used is not user friendly.

Implementation of techniques

Detecting facial landmarks is a subset of the shape prediction problem. Given an input image (and normally an ROI that specifies the object of interest), a shape predictor attempts to localize key points of interest along with the shape.

In the context of facial landmarks, our goal is to detect important facial structures on the face using shape prediction methods.

Detecting facial landmarks is therefore a two-step process:

Step 1: Face Detection

We have used shape_predictor_68 which is already a trained model for facial detection and is available publicly.

pre-trained HOG + Linear SVM object detector specifically for the task of face detection.

In either case, the actual algorithm used to detect the face in the image doesn’t matter. Instead, what’s important is that through some method we obtain the face bounding box (i.e., the (x, y)-coordinates of the face in the image).

Step 2: detecting key facial structures in the face region.

First the camera access will be given when we run the program

Second the facial marks will be analysed and then it will be appointed to eyes, mouth, Nose.

The eyes will have landmarks from 37-48. Nose will have 28-36. mouth 49-68. for our implementation we will be appointing them to the camera.

Then next we will be taking real time input according to the distance between the coordinates appointed for each part.

The EAR (Eye Aspect Ratio) and Mar (Mouth Aspect Ratio)

So, for eyes when the eye aspect ratio is lesser than 0.02 it will be counted as wink.

Similarly for each part we have assigned a threshold and according to that all the functionalities will be done.

PARAMETERS:

- Image or frame currently being scanned
- specific (x, y)-coordinates of regions surrounding each facial structure
- **EAR (Eye Aspect Ratio) :** It is a scalar quantity obtained by detecting a face from an image, finding the Euclidean distance of the corresponding eye coordinates, and substituting it into the following formula.

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

- **MAR(Mouth Aspect Ratio) :** Same as EAR but for mouth coordinates

Algorithm used:

Distance from head to eyes is 50%

Eyes=50% from eyes to nose is 70%

Nose=70%

remaining is mouth, Mouth=100;

Eyes= current_face (1:Eyes);

Nose= current_face(Eyes:Nose);

Mouth= = current_face (Nose:Mouth);

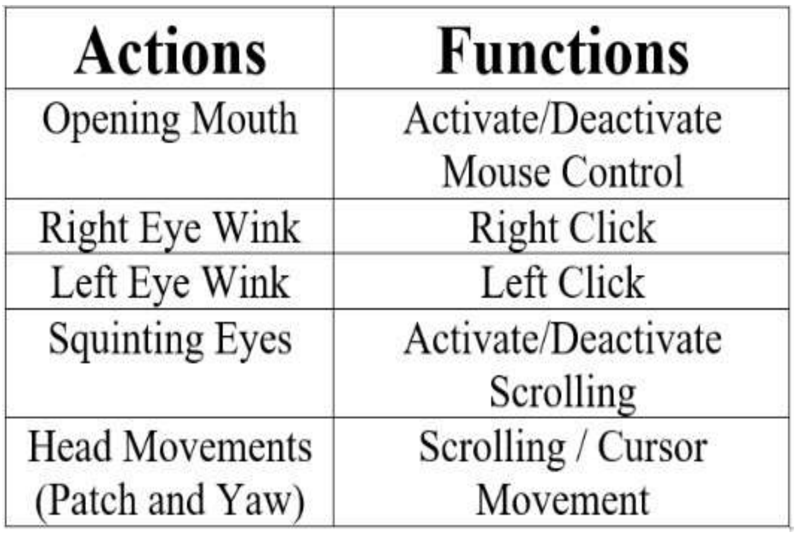
Viola-Jones: The characteristics of Viola–Jones algorithm which make it a good detection algorithm are:

- Robust – very high detection rate (true-positive rate) & very low false-positiverate always.
- Real time – For practical applications at least 2 frames per second must be processed.
- Face detection only (not recognition) - The goal is to distinguish faces fromnon-faces (detection is the first step in the recognition process).

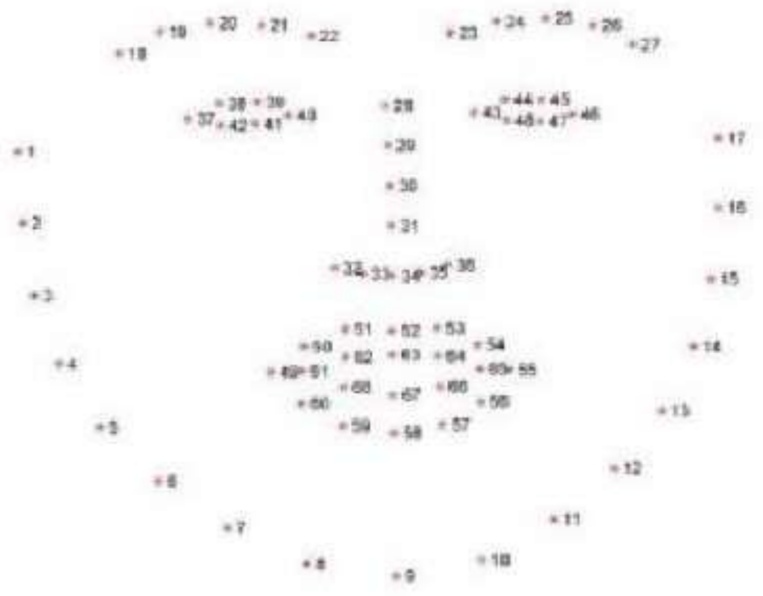
The algorithm has four stages:

- Haar Feature Selection
- Creating an Integral Image
- Adaboost Training
- Cascading Classifiers

The algorithm has linear time complexity o(N) where N is no. of Pixels.



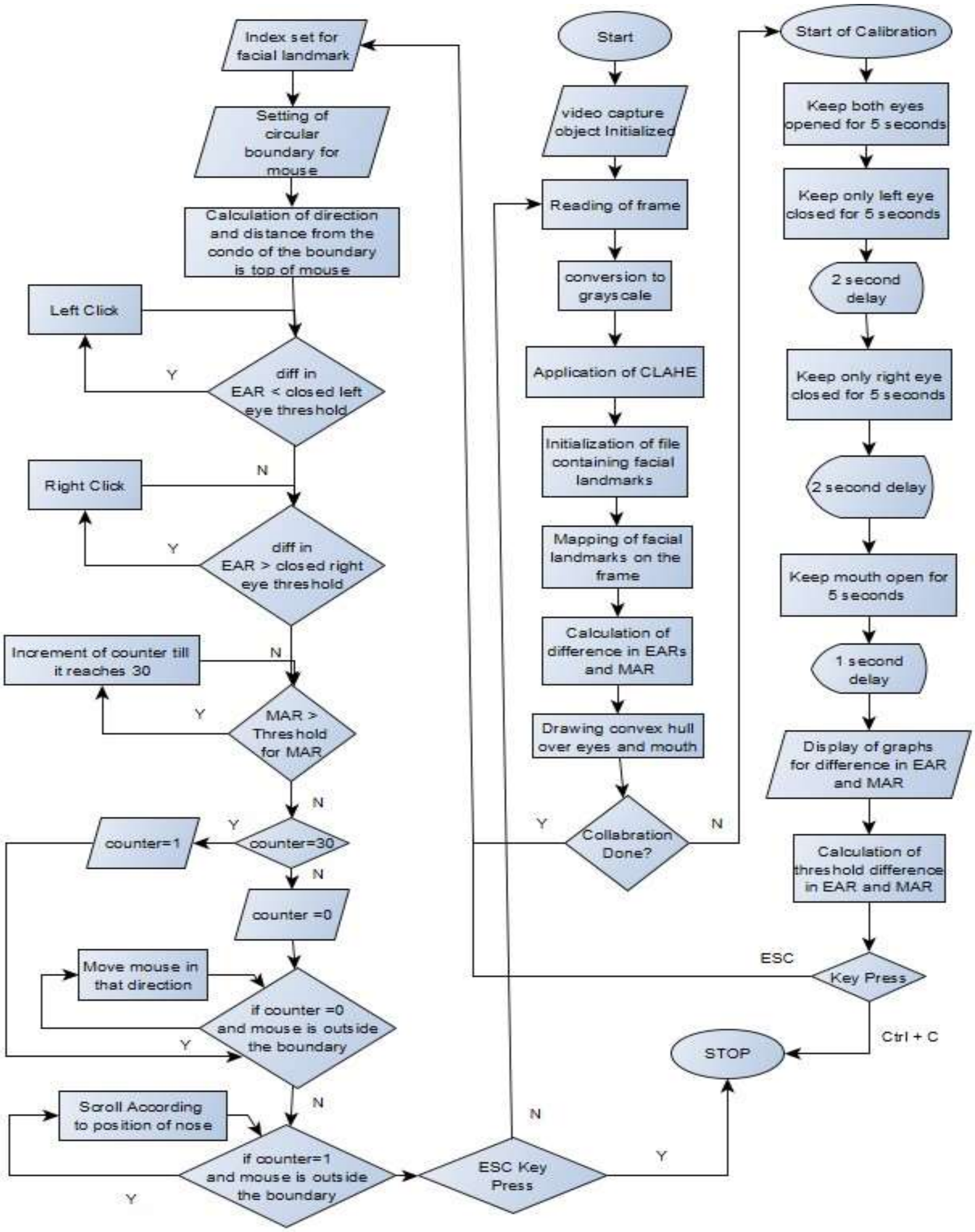
Actions	Functions
Opening Mouth	Activate/Deactivate Mouse Control
Right Eye Wink	Right Click
Left Eye Wink	Left Click
Squinting Eyes	Activate/Deactivate Scrolling
Head Movements (Patch and Yaw)	Scrolling / Cursor Movement



Working Algorithm:

- First the camera access will be given when we run the program
- Second the facial marks will be analysed and then it will be appointed to eyes, mouth, Nose.
- The eyes will have landmarks from 37-48. Nose will have 28-36. mouth 49-68. for our implementation we will be appointing them to the camera.
- Then next we will be taking real time input according to the distance between the coordinates appointed for each part.
- So for eyes when the eye aspect ratio is lesser than 0.02 it will be counted as wink. Similarly for each part we have assigned a threshold and according to that all the functionalities will be done.

Architecture diagram



Index set for facial landmark
Setting of circular boundary for mouse
Calculation of direction and distance from the condo of the boundary is top of mouse
Left Click
diff in EAR < closed left eye threshold
Y
N
Right Click
diff in EAR > closed right eye threshold
Y
N
Increment of counter till it reaches 30
MAR > Threshold for MAR
Y
N
counter=1
counter=30
Y
N
counter =0
Move mouse in that direction
if counter =0 and mouse is outside the boundary
Y
N
Scroll According to position of nose
if counter=1 and mouse is outside the boundary
Y
ESC Key Press
N
Y
Start
video capture object Initialized
Reading of frame
conversion to grayscale
Application of CLAHE
Initialization of file containing facial landmarks
Mapping of facial landmarks on the frame
Calculation of difference in EARs and MAR
Drawing convex hull over eyes and mouth
Collabration Done?
Y
N
STOP
Start of Calibration
Keep both eyes opened for 5 seconds
Keep only left eye closed for 5 seconds
2 second delay
Keep only right eye closed for 5 seconds
2 second delay
Keep mouth open for 5 seconds
1 second delay
Display of graphs for difference in EAR and MAR
Calculation of threshold difference in EAR and MAR
Key Press
ESC
Ctrl + C

Experimmental Setup

Modules Required: utils , numpy , pyautogui , imutils , dlib , cv2

Softwares Requried: VS Code , Jupyter Notebook

Utils: Python Utils is a collection of small Python functions and classes which make common patterns shorter and easier. It is by no means a complete collection but it has served me quite a bit in the past and I will keep extending it.

Numpy: NumPy is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices.NumPy stands for Numerical Python.NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely.

Pyautogui: PyAutoGUI is a cross-platform GUI automation Python module for human beings. Used to programmatically control the mouse & keyboard.

imutils: A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges, and much more easier with OpenCV and both Python 2.7 and Python 3.

Dlib: It‘s a landmark’s facial detector with pre-trained models, the dlib is used to estimate the location of 68 coordinates (x, y) that map the facial points on a person’s face

Cv2: Wrapper package for OpenCV python bindings.Pre-built CPU-only OpenCV packages for Python.Check the manual build section if you wish to compile the bindings from source to enable additional modules such as CUDA.

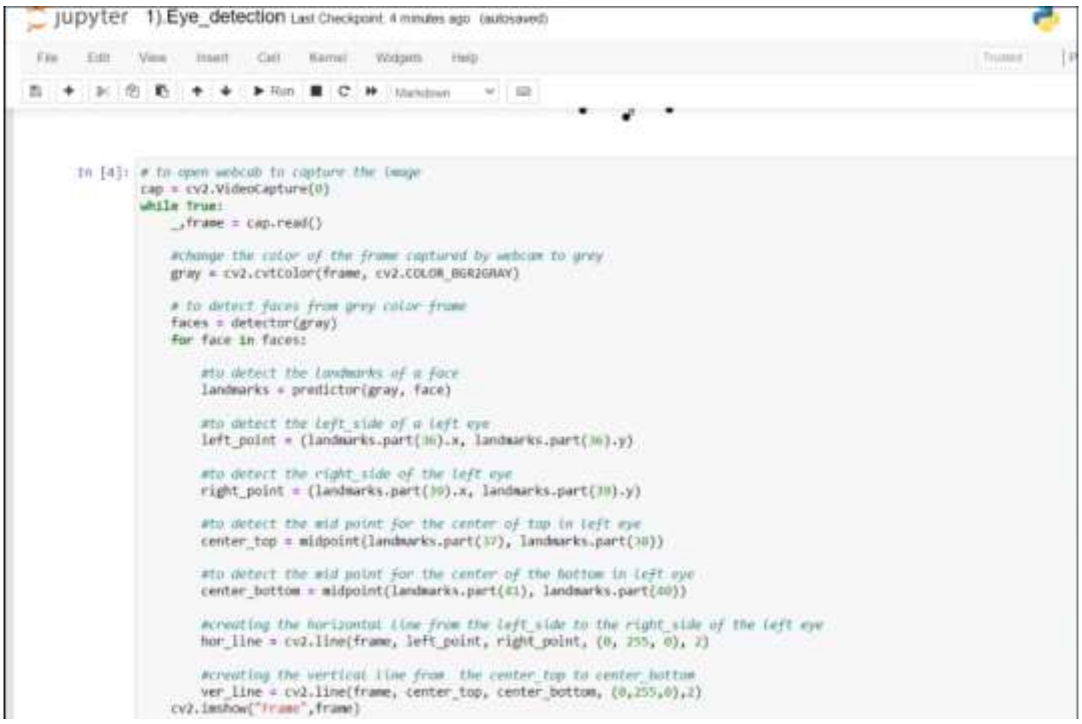
Code:

```
from imutils import face_utils
from utils import *
import numpy as np
import pyautogui as pyag
import imutils
import dlib
import cv2

# Thresholds and consecutive frame length for triggering the mouse action.
MOUTH_AR_THRESH = 0.6
MOUTH_AR_CONSECUTIVE_FRAMES = 15
EYE_AR_THRESH = 0.19
```

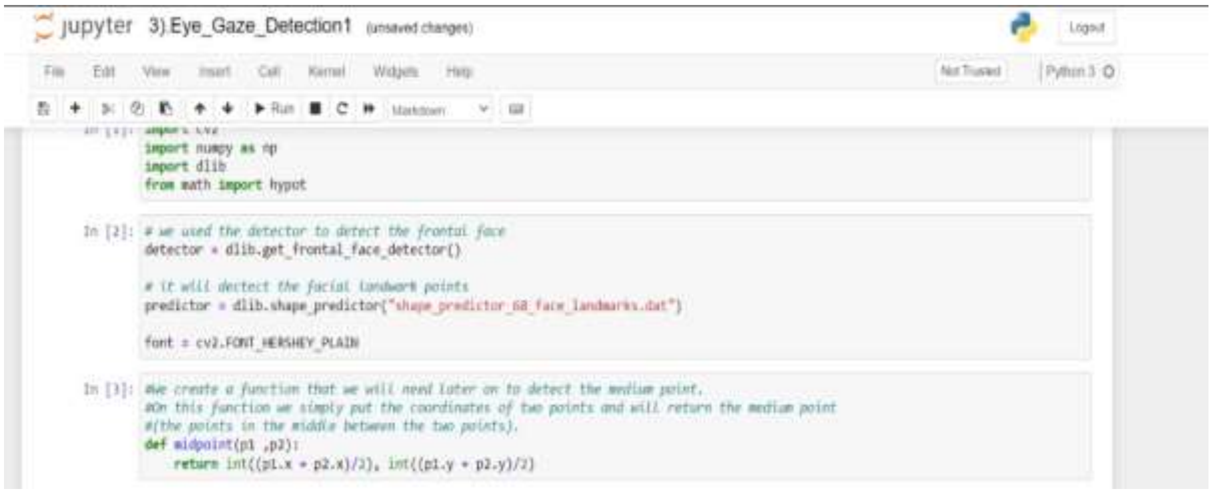
```
EYE_AR_CONSECUTIVE_FRAMES = 15
WINK_AR_DIFF_THRESH = 0.04
WINK_AR_CLOSE_THRESH = 0.19
WINK_CONSECUTIVE_FRAMES = 10

# Initialize the frame counters for each action as well as
# booleans used to indicate if action is performed or not
MOUTH_COUNTER = 0
EYE_COUNTER = 0
WINK_COUNTER = 0
INPUT_MODE = False
EYE_CLICK = False
LEFT_WINK = False
RIGHT_WINK = False
SCROLL_MODE = False
ANCHOR_POINT = (0, 0)
WHITE_COLOR = (255, 255, 255)
YELLOW_COLOR = (0, 255, 255)
RED_COLOR = (0, 0, 255)
GREEN_COLOR = (0, 255, 0)
BLUE_COLOR = (255, 0, 0)
BLACK_COLOR = (0, 0, 0)

# Initialize Dlib's face detector (HOG-based) and then create
# the facial landmark predictor
shape_predictor = "model/shape_predictor_68_face_landmarks.dat"
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(shape_predictor)

# Grab the indexes of the facial landmarks for the left and
# right eye, nose and mouth respectively
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
(nStart, nEnd) = face_utils.FACIAL_LANDMARKS_IDXS["nose"]
(mStart, mEnd) = face_utils.FACIAL_LANDMARKS_IDXS["mouth"]

# Video capture
vid = cv2.VideoCapture(0)
resolution_w = 1366
resolution_h = 768
cam_w = 640
cam_h = 480
unit_w = resolution_w / cam_w
unit_h = resolution_h / cam_h

while True:
    # Grab the frame from the threaded video file stream, resize
    # it, and convert it to grayscale
    # channels)
    _, frame = vid.read()
    frame = cv2.flip(frame, 1)
    frame = imutils.resize(frame, width=cam_w, height=cam_h)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect faces in the grayscale frame
    rects = detector(gray, 0)

    # Loop over the face detections
    if len(rects) > 0:
        rect = rects[0]
```

```
else:
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
    continue

# Determine the facial landmarks for the face region, then
# convert the facial landmark (x, y)-coordinates to a NumPy
# array
shape = predictor(gray, rect)
shape = face_utils.shape_to_np(shape)

# Extract the left and right eye coordinates, then use the
# coordinates to compute the eye aspect ratio for both eyes
mouth = shape[mStart:mEnd]
leftEye = shape[lStart:lEnd]
rightEye = shape[rStart:rEnd]
nose = shape[nStart:nEnd]

# Because I flipped the frame, left is right, right is left.
temp = leftEye
leftEye = rightEye
rightEye = temp

# Average the mouth aspect ratio together for both eyes
mar = mouth_aspect_ratio(mouth)
leftEAR = eye_aspect_ratio(leftEye)
rightEAR = eye_aspect_ratio(rightEye)
ear = (leftEAR + rightEAR) / 2.0
diff_ear = np.abs(leftEAR - rightEAR)

nose_point = (nose[3, 0], nose[3, 1])

# Compute the convex hull for the left and right eye, then
# visualize each of the eyes
mouthHull = cv2.convexHull(mouth)
leftEyeHull = cv2.convexHull(leftEye)
rightEyeHull = cv2.convexHull(rightEye)
cv2.drawContours(frame, [mouthHull], -1, YELLOW_COLOR, 1)
cv2.drawContours(frame, [leftEyeHull], -1, YELLOW_COLOR, 1)
cv2.drawContours(frame, [rightEyeHull], -1, YELLOW_COLOR, 1)

for (x, y) in np.concatenate((mouth, leftEye, rightEye), axis=0):
    cv2.circle(frame, (x, y), 2, GREEN_COLOR, -1)

# Check to see if the eye aspect ratio is below the blink
# threshold, and if so, increment the blink frame counter
if diff_ear > WINK_AR_DIFF_THRESH:

    if leftEAR < rightEAR:
        if leftEAR < EYE_AR_THRESH:
            WINK_COUNTER += 1

            if WINK_COUNTER > WINK_CONSECUTIVE_FRAMES:
                pag.click(button='left')

                WINK_COUNTER = 0

    elif leftEAR > rightEAR:
        if rightEAR < EYE_AR_THRESH:
            WINK_COUNTER += 1
```

```
                if WINK_COUNTER > WINK_CONSECUTIVE_FRAMES:
                    pag.click(button='right')

                    WINK_COUNTER = 0
        else:
            WINK_COUNTER = 0
    else:
        if ear <= EYE_AR_THRESH:
            EYE_COUNTER += 1

            if EYE_COUNTER > EYE_AR_CONSECUTIVE_FRAMES:
                SCROLL_MODE = not SCROLL_MODE
                # INPUT_MODE = not INPUT_MODE
                EYE_COUNTER = 0

                # nose point to draw a bounding box around it

        else:
            EYE_COUNTER = 0
            WINK_COUNTER = 0

    if mar > MOUTH_AR_THRESH:
        MOUTH_COUNTER += 1

        if MOUTH_COUNTER >= MOUTH_AR_CONSECUTIVE_FRAMES:
            # if the alarm is not on, turn it on
            INPUT_MODE = not INPUT_MODE
            # SCROLL_MODE = not SCROLL_MODE
            MOUTH_COUNTER = 0
            ANCHOR_POINT = nose_point

    else:
        MOUTH_COUNTER = 0

    if INPUT_MODE:
        cv2.putText(frame, "READING INPUT!", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, RED_COLOR, 2)
        x, y = ANCHOR_POINT
        nx, ny = nose_point
        w, h = 60, 35
        multiple = 1
        cv2.rectangle(frame, (x - w, y - h), (x + w, y + h), GREEN_COLOR, 2)
        cv2.line(frame, ANCHOR_POINT, nose_point, BLUE_COLOR, 2)

        dir = direction(nose_point, ANCHOR_POINT, w, h)
        cv2.putText(frame, dir.upper(), (10, 90), cv2.FONT_HERSHEY_SIMPLEX,
0.7, RED_COLOR, 2)
        drag = 18
        if dir == 'right':
            pyag.moveRel(drag, 0)
        elif dir == 'left':
            pyag.moveRel(-drag, 0)
        elif dir == 'up':
            if SCROLL_MODE:
                pyag.scroll(40)
            else:
                pyag.moveRel(0, -drag)
        elif dir == 'down':
            if SCROLL_MODE:
                pyag.scroll(-40)
            else:
                pyag.moveRel(0, drag)

    if SCROLL_MODE:
        cv2.putText(frame, 'SCROLL MODE IS ON!', (10, 60),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, RED_COLOR, 2)

    # cv2.putText(frame, "MAR: {:.2f}".format(mar), (500, 30),
    #             cv2.FONT_HERSHEY_SIMPLEX, 0.7, YELLOW_COLOR, 2)
    # cv2.putText(frame, "Right EAR: {:.2f}".format(rightEAR), (460, 80),
    #             cv2.FONT_HERSHEY_SIMPLEX, 0.7, YELLOW_COLOR, 2)
    # cv2.putText(frame, "Left EAR: {:.2f}".format(leftEAR), (460, 130),
    #             cv2.FONT_HERSHEY_SIMPLEX, 0.7, YELLOW_COLOR, 2)
    # cv2.putText(frame, "Diff EAR: {:.2f}".format(np.abs(leftEAR -
rightEAR)), (460, 80),
    #             cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

    # Show the frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # If the `Esc` key was pressed, break from the loop
    if key == 27:
        break

# Do a bit of cleanup
cv2.destroyAllWindows()
vid.release()
```

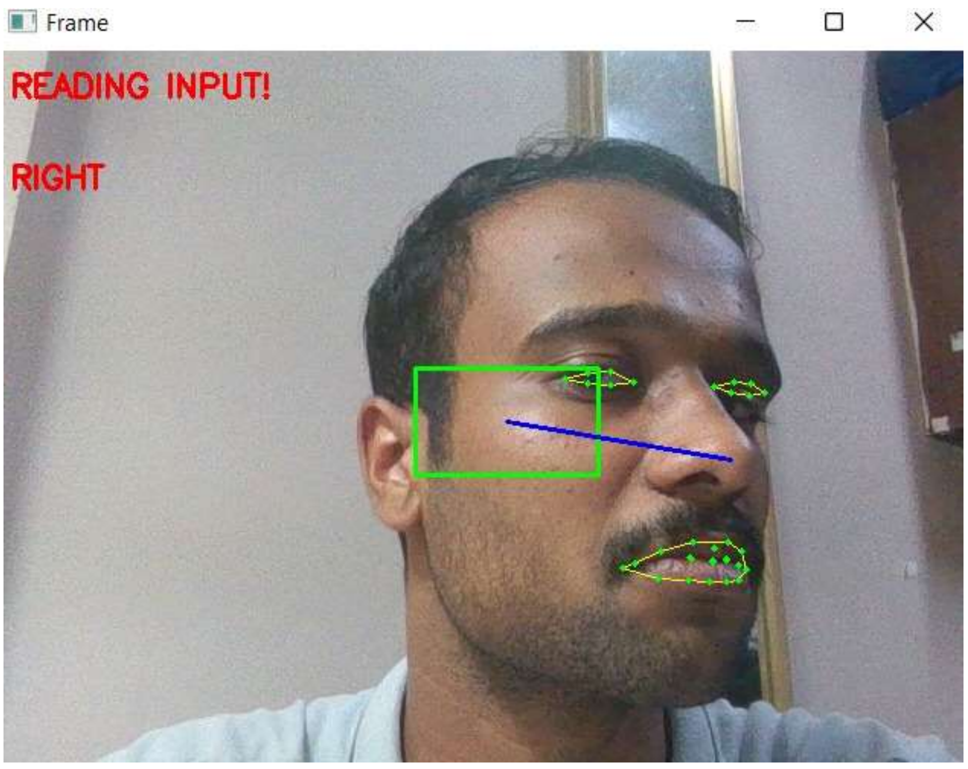

jupyter 1).Eye_detection Last Checkpoint: 4 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Run
Markdown

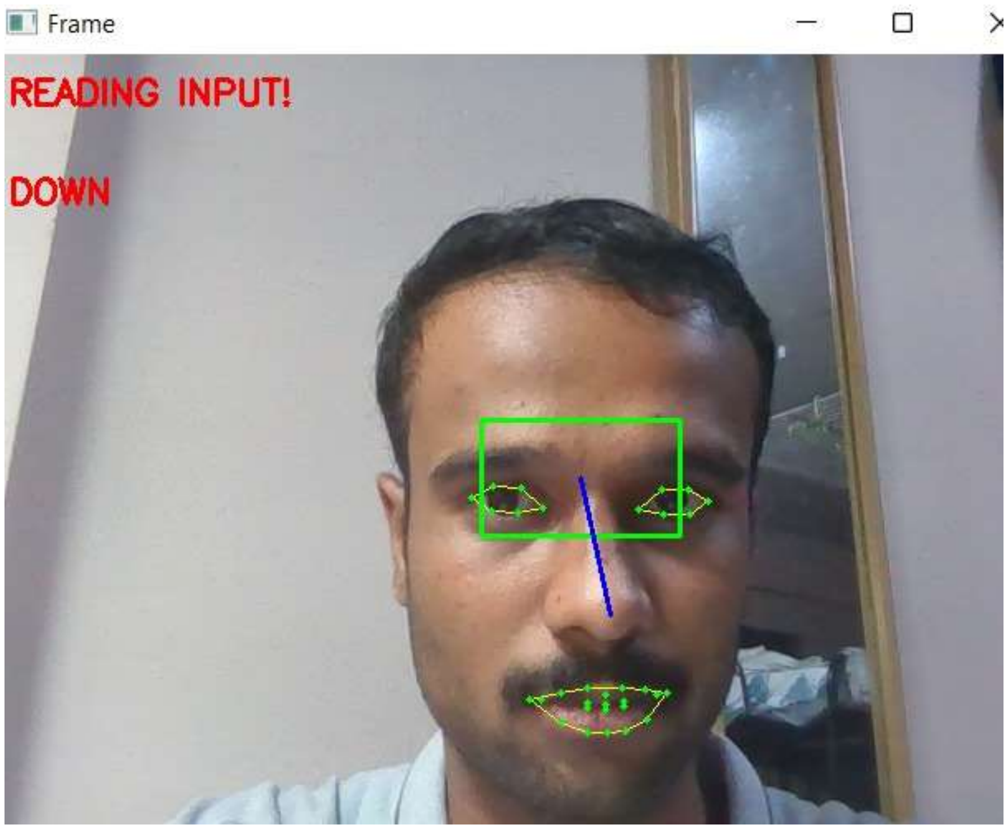
```
In [4]: # to open webcub to capture the image
        cap = cv2.VideoCapture(0)
        while True:
            _,frame = cap.read()

            #change the color of the frame captured by webcam to grey
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

            # to detect faces from grey color frame
            faces = detector(gray)
            for face in faces:

                #to detect the landmarks of a face
                landmarks = predictor(gray, face)

                #to detect the left_side of a left eye
                left_point = (landmarks.part(36).x, landmarks.part(36).y)

                #to detect the right_side of the left eye
                right_point = (landmarks.part(39).x, landmarks.part(39).y)

                #to detect the mid point for the center of top in left eye
                center_top = midpoint(landmarks.part(37), landmarks.part(38))

                #to detect the mid point for the center of the bottom in left eye
                center_bottom = midpoint(landmarks.part(41), landmarks.part(40))

                #creating the horizontal line from the left_side to the right_side of the left eye
                hor_line = cv2.line(frame, left_point, right_point, (0, 255, 0), 2)

                #creating the vertical line from  the center_top to center_bottom
                ver_line = cv2.line(frame, center_top, center_bottom, (0,255,0),2)
            cv2.imshow("frame",frame)
```

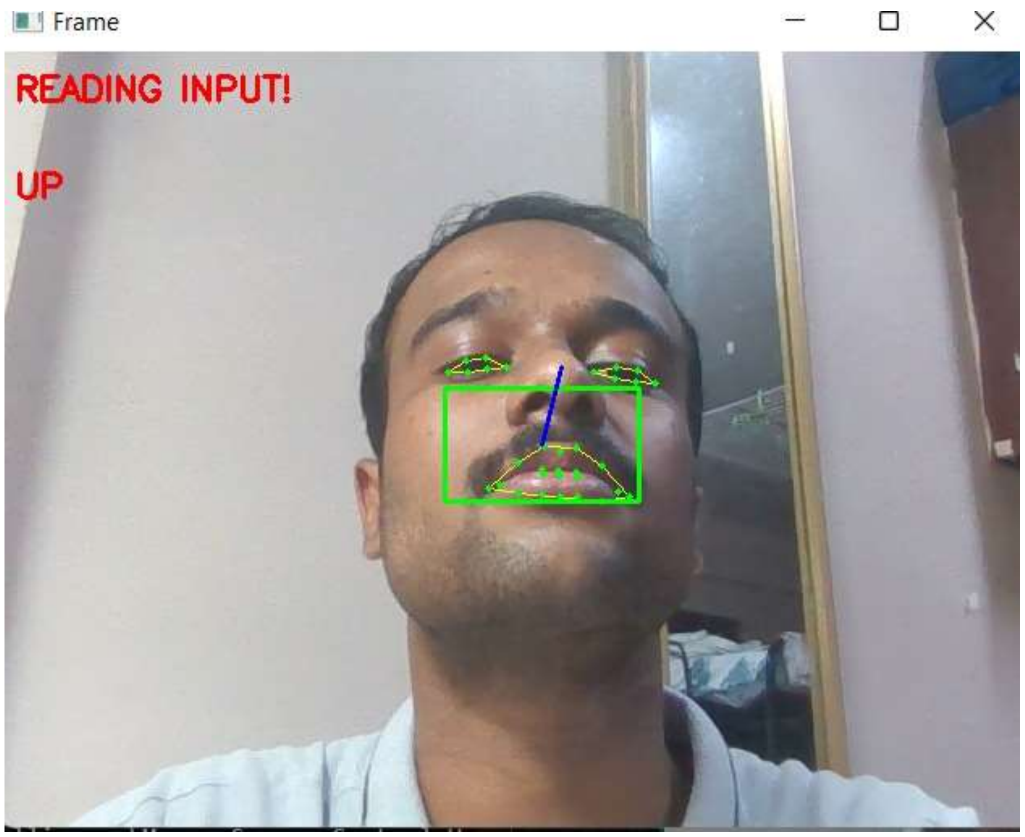

jupyter 3).Eye_Gaze_Detection1 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown

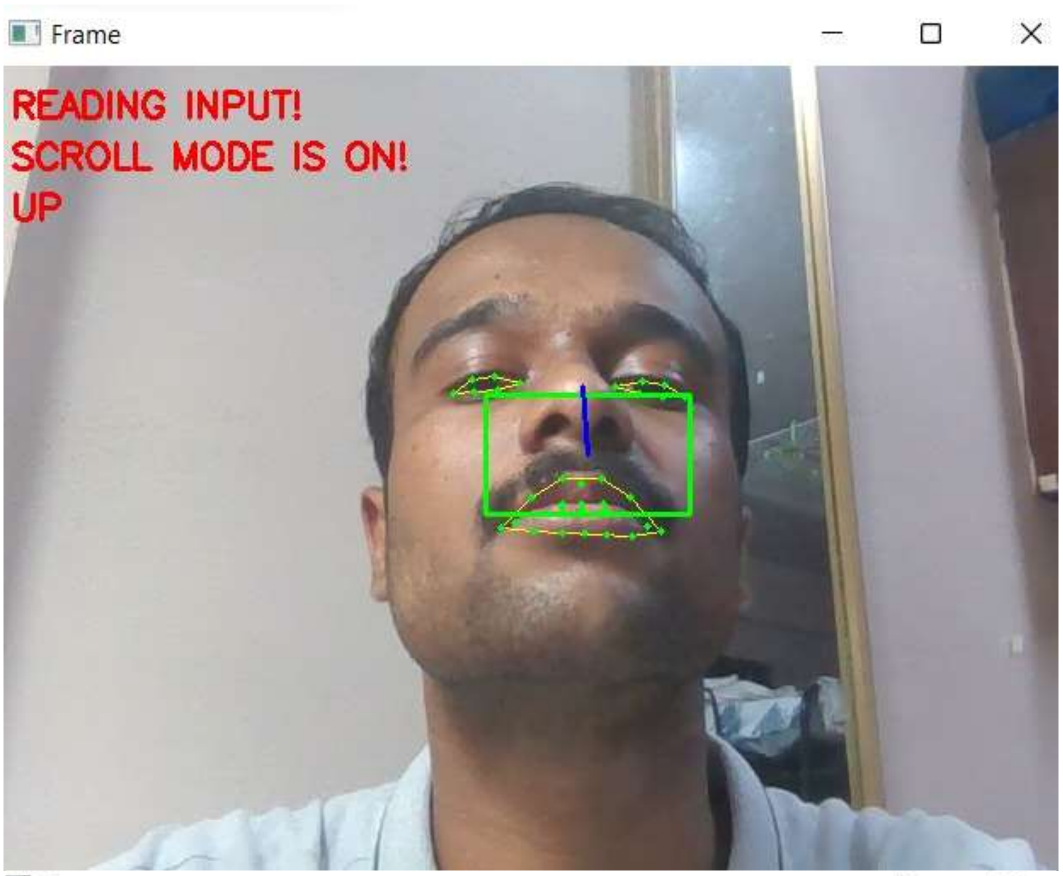
```
In [1]: import cv2
        import numpy as np
        import dlib
        from math import hypot

In [2]: # we used the detector to detect the frontal face
        detector = dlib.get_frontal_face_detector()

        # it will dectect the facial landwork points
        predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

        font = cv2.FONT_HERSHEY_PLAIN

In [3]: #we create a function that we will need later on to detect the medium point.
        #On this function we simply put the coordinates of two points and will return the medium point
        #(the points in the middle between the two points).
        def midpoint(p1 ,p2):
            return int((p1.x + p2.x)/2), int((p1.y + p2.y)/2)
```

<u>Output:</u>

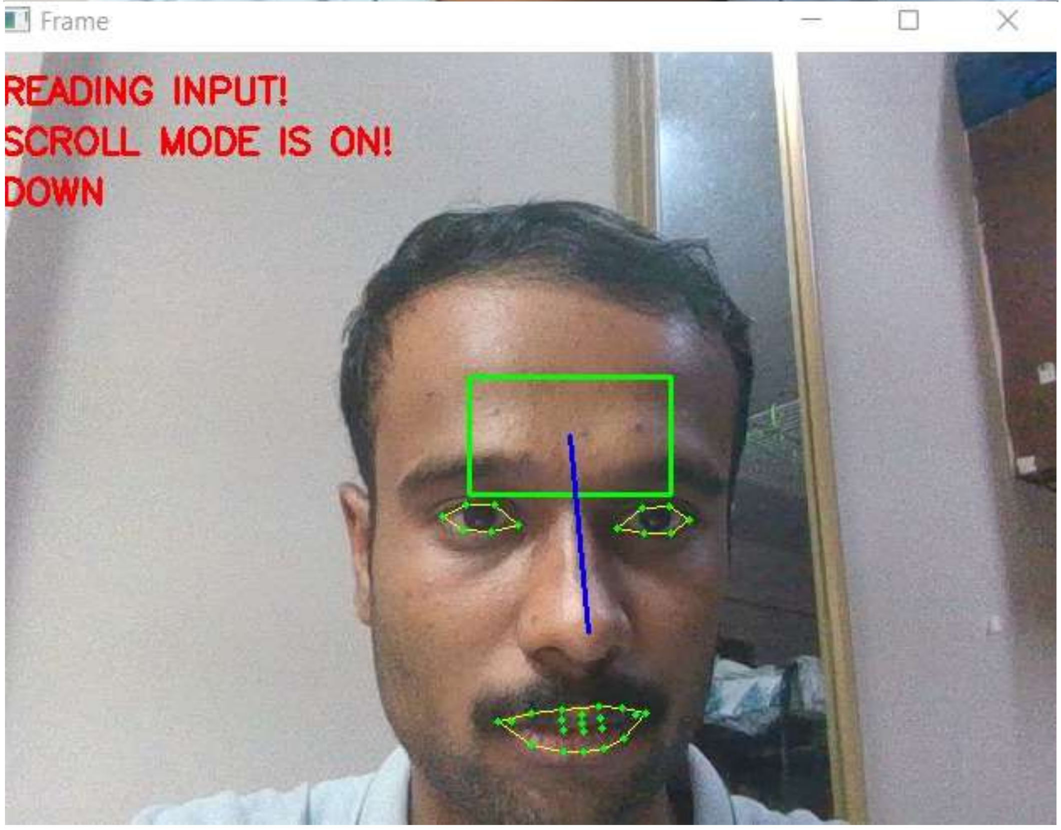



Frame
READING INPUT!
DOWN

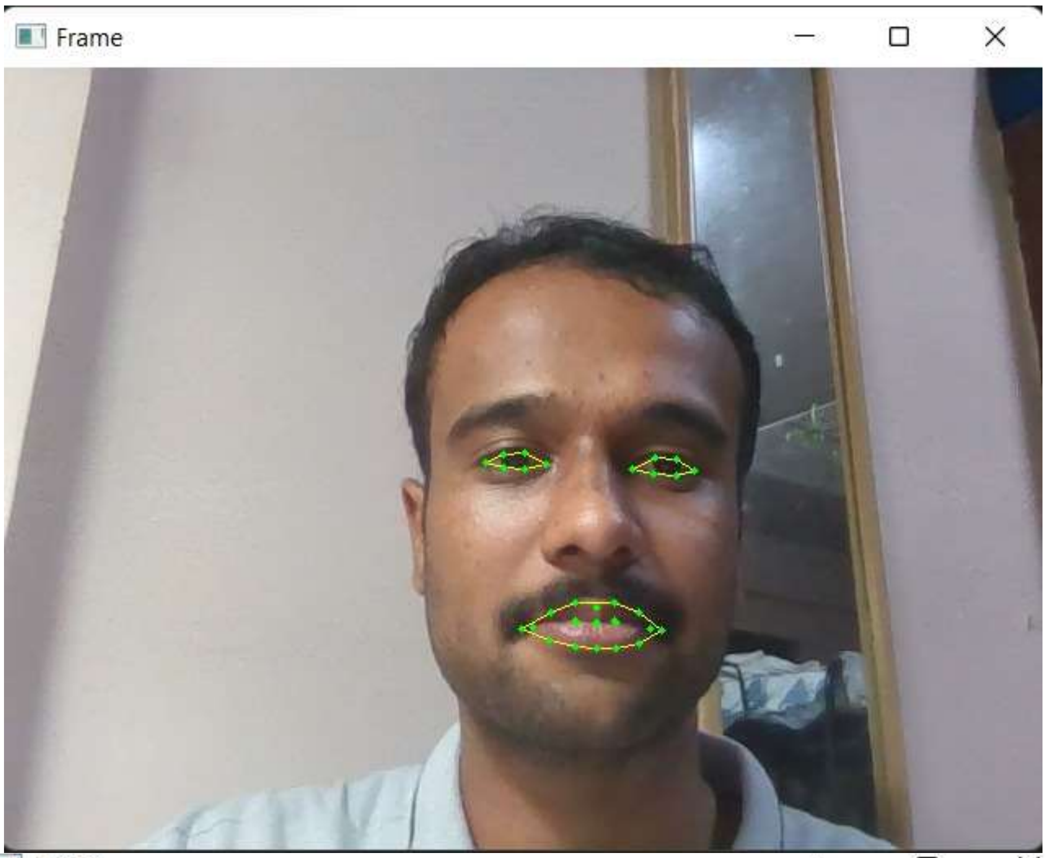

Frame
READING INPUT!
UP

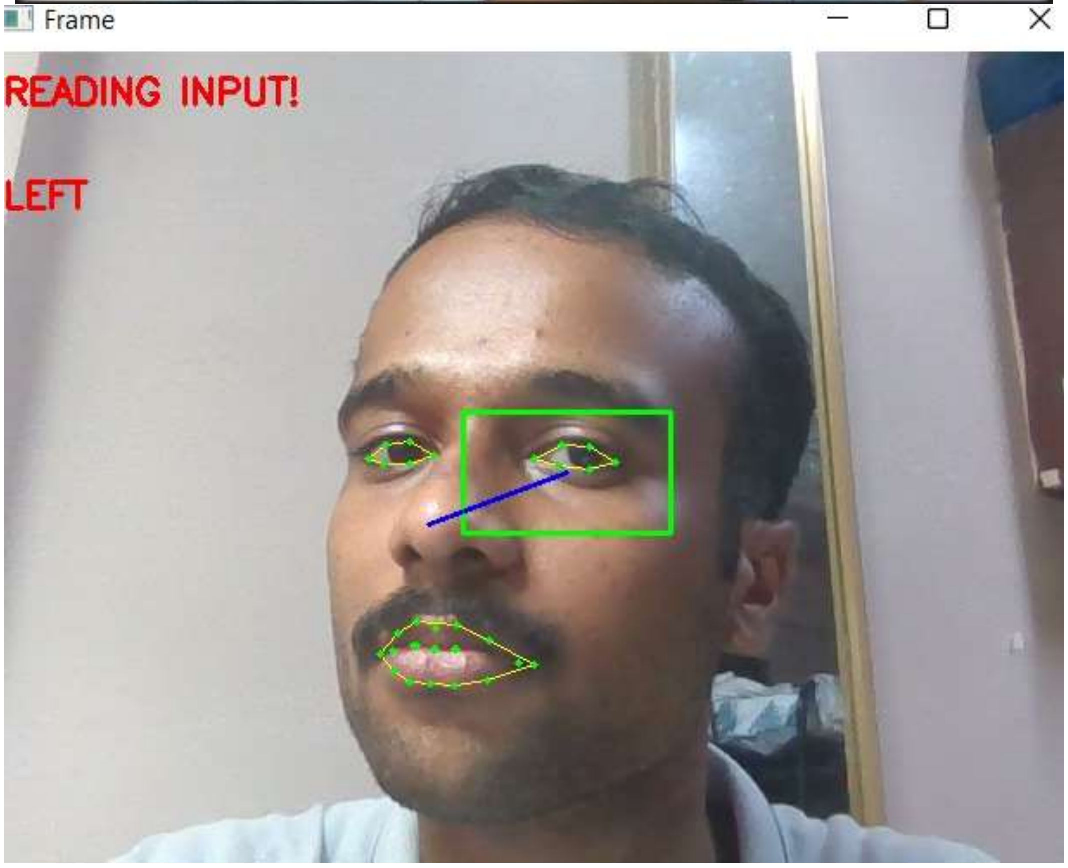


READING INPUT!
SCROLL MODE IS ON!
UP



READING INPUT!
SCROLL MODE IS ON!
DOWN


Frame


Frame
READING INPUT!
LEFT

Advantages:

- All the actions that are shown work according to their desired function.
- The Project also clearly depicts the face and its facial movements and thefunctions are performed accordingly.
- The detection of eye gaze and the nose also works with good accuracy.
- The application is user-friendly, and users are given clear instructions, feedback, and system visibility status.

Disadvantages:

- The accuracy decreases in darkness.
- Users might need some time to get accustomed to the facial controls.

Realtime Application:

Useful for physically handicap individuals who are not capable of physically using a PC.

Future Enhancements:

- Being able to use other apps and features using gesture control.
- Can be used for movement on mobile phone.

References:

[1] Cech, J. and Soukupova, T., 2016. Real-time eye blink detection using facial landmarks.Cent. Mach. Perception, Dep. Cybern. Fac. Electr. Eng. Czech Tech. Univ. Prague, pp.1-8.
[2] Chau, M. and Betke, M., 2005. Real time eye tracking and blink detection with usb cameras.
Boston University Computer Science Department.
[3] Zhang, X., Liu, X., Yuan, S.M. and Lin, S.F., 2017. Eye tracking based control system for natural human-computer interaction. Computational intelligence and neuroscience, 2017.
[4] Chakraborty, P., Roy, D., Zahid, Z.R. and Rahman, S., 2019. Eye gaze controlled virtual keyboard. Int. J. Rec. Technol. Eng.(IJRTE), 8(4), pp.3264-3269.

[5] Martins, J.M., Rodrigues, J.M. and Martins, J.A., 2015. Low-cost natural interface based on head movements. Procedia Computer Science, 67, pp.312-321.

[6] Lalithamani, N., 2016. Gesture control using single camera for PC. Procedia Computer Science, 78, pp.146-152.

[7] Šumak, B., Špindler, M., Debeljak, M., Heričko, M. and Pušnik, M., 2019. An empirical evaluation of a hands-free computer interaction for users with motor disabilities. Journal of biomedical informatics, 96, p.103249.

[8] Dobosz, K. and Stawski, K., 2017, October. Touchless virtual keyboard controlled by eye blinking and EEG signals. In International Conference on Man–Machine Interactions (pp. 52-61). Springer, Cham.

[9] Chai, X., Shan, S., Chen, X. and Gao, W., 2007. Locally linear regression for pose-invariant face recognition. IEEE Transactions on image processing, 16(7), pp.1716-1725.

[10] Asthana, A., Zafeiriou, S., Cheng, S. and Pantic, M., 2014. Incremental face alignment in the wild. In Proceedings of the IEEE conference on computer vision and pattern recognition
(pp. 1859-1866).

[11] Aggarwal, G., Chowdhury, A.R. and Chellappa, R., 2004, August. A system identification approach for video-based face recognition. In Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004. (Vol. 4, pp. 175-178). IEEE.

[12] Breuer, P., Kim, K.I., Kienzle, W., Scholkopf, B. and Blanz, V., 2008, September. Automatic 3D face reconstruction from single images or video. In 2008 8th IEEE International Conference on Automatic Face & Gesture Recognition (pp. 1-8). IEEE.

[13] Cech, J., Franc, V. and Matas, J., 2014, August. A 3D approach to facial landmarks: Detection, refinement, and tracking. In 2014 22nd International Conference on Pattern Recognition (pp. 2173-2178). IEEE.

[14] Liu, X. and Cheng, T., 2003, June. Video-based face recognition using adaptive hidden markov models. In 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings. (Vol. 1, pp. I-I). IEEE.

[15] Li, S., Liu, X., Chai, X., Zhang, H., Lao, S. and Shan, S., 2012, October. Morphable displacement field based image matching for face recognition across pose. In European conference on computer vision (pp. 102-115). Springer, Berlin, Heidelberg.

[16] Aggarwal, G., Chowdhury, A.R. and Chellappa, R., 2004, August. A system identification approach for video-based face recognition. In Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004. (Vol. 4, pp. 175-178). IEEE.

[17] Pentland, A., Moghaddam, B. and Starner, T., 1994. View-based and modular Eigenspaces for face recognition.
[18] Cai, Q., Sankaranarayanan, A., Zhang, Q., Zhang, Z. and Liu, Z., 2010, June. Real time head pose tracking from multiple cameras with a generic model. In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops (pp. 25-32). IEEE.
[19] Bergasa, L.M., Nuevo, J., Sotelo, M.A., Barea, R. and Lopez, M.E., 2006. Real-time system for monitoring driver vigilance. IEEE Transactions on Intelligent Transportation Systems, 7(1), pp.63-77.
[20] Lee, H.S. and Kim, D., 2006. Generating frontal view face image for pose invariant face recognition. Pattern recognition letters, 27(7), pp.747-754.