



# **HANDS FREE MOUSE CONTROL FOR PARAPLEGICS AND PARALYTICS**

*J – Component for the course*

CSE4015 – Human Computer Interaction

*By*

Yashvardhan Singh Bhadauria-19BCE2129

Gunik Luthra - 19BCE2285

Aryaman Sandeep Jaisinghani-19BCE0853

*Under the Guidance of*

Dr. GUNAVATHI C

### **DECLARATION BY THE MEMBERS**

We hereby declare that the project report entitled “**HANDS FREE MOUSE CONTROL FOR PARAPLEGICS AND PARALYTICS**” submitted by us to VIT University, Vellore is a record of J - component project work carried out *by us under the guidance of Dr. GUNAVATHI C for the course CSE4015 – Human Computer Interaction.* We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

## **ABSTRACT**

In this generation, computers are the most used devices. Their uses are innumerable. They are used for communication, entertainment research, and education. Paraplegics and paralytics are people who are unable to move parts of their body from areas below their waist. Paraplegia is also called leg paralysis, a condition that affects all or part of the torso, legs and pelvic organs, which is due to loss of muscle function. Paraplegia can occur after a spinal cord injury. It's caused by damage to the vertebrae, ligaments or disks of the spinal column.

Paraplegia is the loss of muscle function in the lower half of the body, including both legs.

Rehabilitation, medication and medical devices allow many people with spinal cord injuries to lead productive, independent lives. Which is the reason why this project for cursor movement for paraplegics and paralytics is deemed suitable for the project, because it is a step toward a real-life problem that is faced by people who are physically challenged in this manner. In fact, when typed “how do paraplegics” is typed in the Google search bar, it auto completes to many results such as “how do paraplegics use the restroom”. Such is their condition. If it is this challenging to perform basic tasks such as going to the washroom, it would be impossible to imagine how they would lead professional lives, which involves using computers, smartphones, and other similar devices. In today’s day and age, we cannot live without technology. Hence it is essential that we find a convenient method that such physically challenged people can easily access such technological devices. The topic involves using a hands-free method to control the device. A webcam or camera or any device which can detect the face of the person, will detect the eye and its movements, and accordingly move the cursor on the screen of the computer. This seems like a convenient way to control a laptop or computer as the person does not have to move much to control the computer, it can be done just with eye movements. This is done with the help of Python programming language, a few libraries for it, and then this program interfaces with the webcam to detect the eye and move the cursor accordingly.

## **INTRODUCTION**

We are using Computer Vision concepts to try and solve the functionality gap between existing systems and the differently abled people who might want to use said systems, whether it be for personal use, leisure, work, or even a communication device.

We will make use of *Facial Detection* to create a prototype of an application which allows users who don't have the use of their arms, to be able to access their computers. This will be done by enabling the mouse cursor to act according to certain sequences and actions performed by the person using their face and head to move said cursor and click on things. Since most modern operating systems have an inbuilt virtual keyboard, this also solves the problem of not being able to use the physical keyboard.

While this is not a perfect system, and has certain drawbacks, we feel it is somewhat of a starting point towards developing more such assistants for the less fortunate people. Oftentimes, people with special needs are ignored and that not only hinders their development and inclusion into society, but also stops them from being as productive as they could be or achieve their full potential. Moreover, a large section of society and a potential demographic is being left out due to lack of development in this particular domain.

With the advent of Artificial Intelligence into our daily lives, and it becoming more and more ingrained with the fabric of society, it only makes sense that such systems be also used for the betterment of an undervalued and ignored sect.

What we are hoping to achieve in this project is to use existing theoretical concepts of AI and use them in a practical and productive manner. The application which we propose does exactly that. Based on high level research work of remarkable people, we hope to do something innovative.

## LITERATURE SURVEY

Author	Title/date	Abstract	publishing	Review
Tereza Soukupova' and Jan C	Real-Time Eye Blink Detection using Facial Landmarks	A real-time algorithm to detect eye blinks in a video sequence from a standard camera is proposed. Recent landmark detectors, trained on in-the wild datasets exhibit excellent robustness against a head orientation with respect to a camera, varying illumination and facial expressions. We show that the landmarks are detected precisely enough to reliably estimate the level of the eye opening. The proposed algorithm therefore	21st Computer Vision Winter Workshop Luka Cehovin, Rok Mandelj c, Vitomir Struc (eds.)  Rimske Toplice, Slovenia, February 3–5, 2016	This paper gives an idea of how to take up with the blinking patterns, facial expressions and the lighting conditions .

		<p>estimates the landmark positions, extracts a single scalar quantity – eye aspect ratio (EAR) – characterizing the eye opening in each frame. Finally, an SVM classifier detects eye blinks as a pattern of EAR values in a short temporal window. The simple algorithm outperforms the state-of-the-art results on two standard datasets</p>		
--	--	---	--	--

Vahid Kazemi, Josephine Sullivan	One millisecond face alignment with an ensemble of regression trees	This paper addresses the problem of Face Alignment for a single image. We show how an ensemble of regression trees can be used to estimate the face's landmark positions directly from a sparse subset of pixel intensities, achieving super real-time performance with high quality predictions. We present a general framework based on gradient boosting for learning an ensemble of regression trees that optimizes the sum of square error loss and naturally handles missing or	2014 IEEE Conference on Computer Vision and Pattern Recognition	It shows how the ensemble of regression trees can be used to estimate the face's landmark positions directly from a sparse subset of pixel intensities, achieving super real-time performance with high quality predictions.
-------------------------------------	---	---	---	--

		<p>partially labelled data. We show how using appropriate priors exploiting the structure of image data helps with efficient feature selection. Different regularization strategies and its importance to combat overfitting are also investigated. In addition, we analyse the effect of the quantity of training data on the accuracy of the predictions and explore the effect of data augmentation using synthesized data.</p>		
--	--	--	--	--



C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, M. Pantic	300 Faces in-the- Wild Challenge: The first facial landmark localization Challenge	Automatic facial point detection plays arguably the most important role in face analysis. Several methods have been proposed which reported their results on databases of both constrained and unconstrained conditions. Most of these databases provide annotations with different mark-ups and in some cases there are problems related to the accuracy of the fiducial points. The aforementioned issues as well as the lack of a evaluation protocol makes it difficult to compare performance between different	IEEE Int'l Conf. on Computer Vision (ICCV- W), 300 Faces in-the-Wild Challenge (300-W). Sydney, Australia , December 2013	This paper presents a challenge on showing how the facial point detection plays the most important role in face analysis.
--	---	---	--	---

		<p>systems. In this paper, we present the 300 Faces in-the-Wild Challenge: The first facial landmark localization Challenge which is held in conjunction with the International Conference on Computer Vision 2013, Sydney, Australia. The main goal of this challenge is to compare the performance of different methods on a new-collected dataset using the same evaluation protocol and the same mark-up and hence to develop the first standardized benchmark for facial landmark.</p>		
--	--	---	--	--

M. Chau and M. Betke	Real Time Eye Tracking and Blink Detection with USB Cameras	A human-computer interface (HCI) system designed for use by people with severe disabilities is presented. People that are severely paralyzed or afflicted with diseases such as ALS (Lou Gehrig's disease) or multiple sclerosis are unable to move or control any parts of their bodies except for their eyes. The system presented here detects the user's eye blinks and analyses the pattern and duration of the blinks, using them to provide input to the computer in the form of a mouse click. After the automatic	Technical Report 2005-12, Boston University Computer Science, May 2005.	One of the most important features as how to track the movement and blinking of eyes in real-time. The system in play here analyses the pattern and duration of the blinks, using them to provide input to the computer in the form of a mouse click.
----------------------	---	--	---	---

		<p>initialization of the system occurs from the processing of the user's involuntary eye blinks in the first few seconds of use, the eye is tracked in real time using correlation with an online template. If the user's depth changes significantly or rapid head movement occurs, the system is automatically reinitialized. There are no lighting requirements nor offline templates needed for the proper functioning of the system. The system works with inexpensive USB cameras and runs at a frame rate of 30 frames per second. Extensive</p>		
--	--	---	--	--

		experiments were conducted to determine both the system's accuracy in classifying voluntary and involuntary blinks, as well as the system's fitness in varying environment conditions, such as alternative camera placements and different lighting conditions.		
--	--	---	--	--

Taner Danisman, Ioan Marius Bilasco, Chaabane Djeraba, Nacim Ihaddadene	Drowsy Driver Detection System Using Eye Blink Patterns	<p>This paper presents an automatic drowsy driver monitoring and accident prevention system that is based on monitoring the changes in the eye blink duration. Our proposed method detects visual changes in eye locations using the proposed horizontal symmetry feature of the eyes. Our new method detects eye blinks via a standard webcam in real-time at 110fps for a 320×240 resolution. Experimental results in the JZU [3] eye-blink database showed that the proposed</p>	In Machine and Web Intelligence (ICMWI), Oct 2010.	<p>This paper talks about how to tackle if a person is drowsy. The proposed method detects visual changes in eye locations using the proposed horizontal symmetry feature of the eyes.</p>
---	---	---	--	--





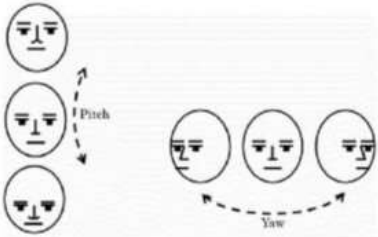
		system detects eye blinks with a 94% accuracy with a 1% false positive rate.		
--	--	--	--	--

## **PROPOSED METHODOLOGY**

The project is centred around using facial landmarks to control the cursor of the mouse. The idea was to use existing facial landmark models to create a system which functions well and requires no external sensors or gadgets. The primary function is to create a method for hands-free movement of the cursor, thus enabling the differently challenged to be able to use the computer for basic needs.

By using specific actions or expressions, we give commands to the computer to move the cursor. Various actions elicit various responses:

- Mouth Open - Activate/Deactivate Mouse Control
- Right Eye Wink - Right Click
- Left Eye Wink - Left Click
- Squinting - Activate and Deactivate scrolling
- Head Movement - Cursor movement or Scroll Movement

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

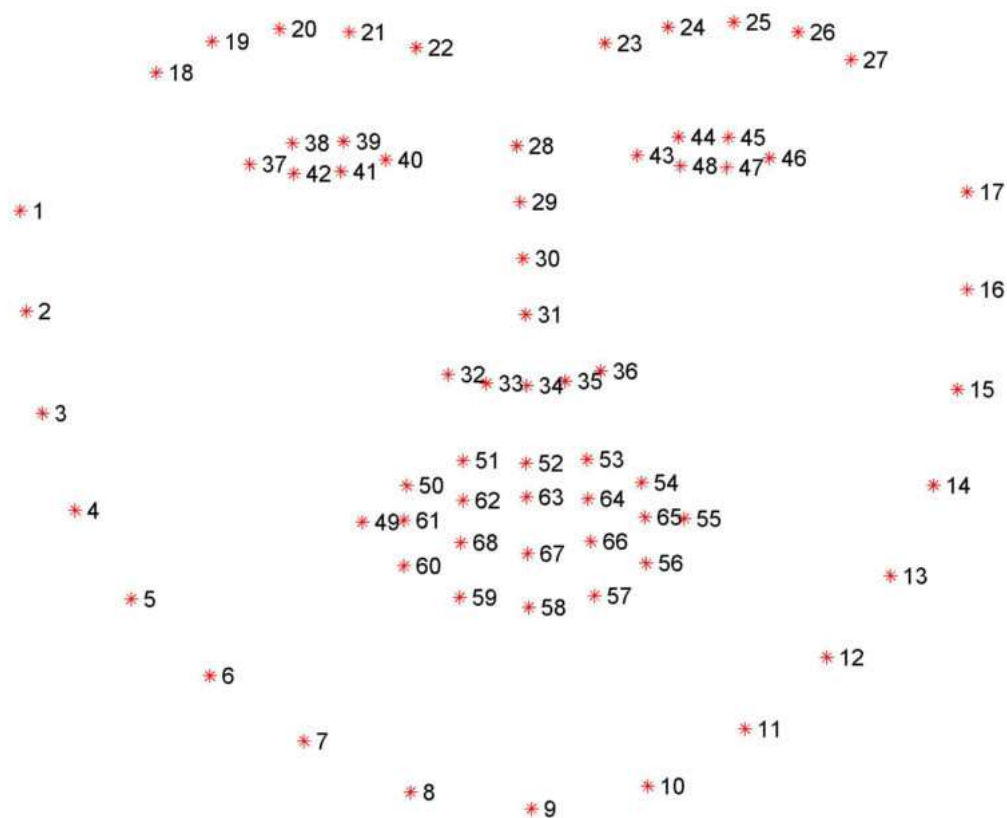
Source: [towardsdatascience.com](https://towardsdatascience.com)



## Working (Metrics, Data Analysis and Base Theory)

As mentioned, the project is based on detecting facial landmarks. A lot of things can be accomplished using these landmarks. We are, however, focusing on the changes in these landmarks as triggers for our application.

For the purpose of detection, we are using a pre-built dlib library, which not only does quick face detection, but also 68 points on the x-y plane of the face, which comes in very handy.



*Source: pyimagesearch.com*

One can exploit these landmarks to detect the various actions needed for our application.

## **-To detect Eyes, Nose, Lip and Jaw**

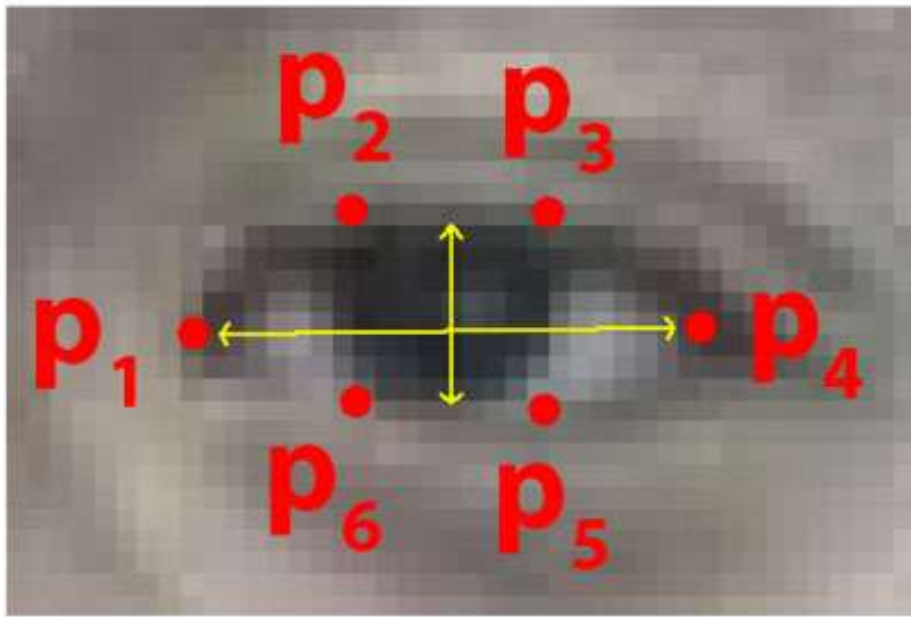
By examining the above diagram, we can see that by using standard Python indexing, one can recognize facial features.

- The *mouth* can be accessed through points [48, 68].
- The *right eyebrow* through points [17, 22].
- The *left eyebrow* through points [22, 27].
- The *right eye* using [36, 42].
- The *left eye* with [42, 48].
- The *nose* using [27, 35].
- And the *jaw* via [0, 17].

One can extract these features and make use of it using the dlib library to detect along with OpenCV (real-time Computer Vision library), imutils (Python library) and code in Python 3.6 and above.

## **-Eye Blinking**

In terms of blink detection, only two sets of facial structures matter - the eyes. Each eye is represented by 6 x-y coordinates from p1 to p6 starting from the left corner of the eye and working its way around clockwise.



*Source: pyimagesearch.com*

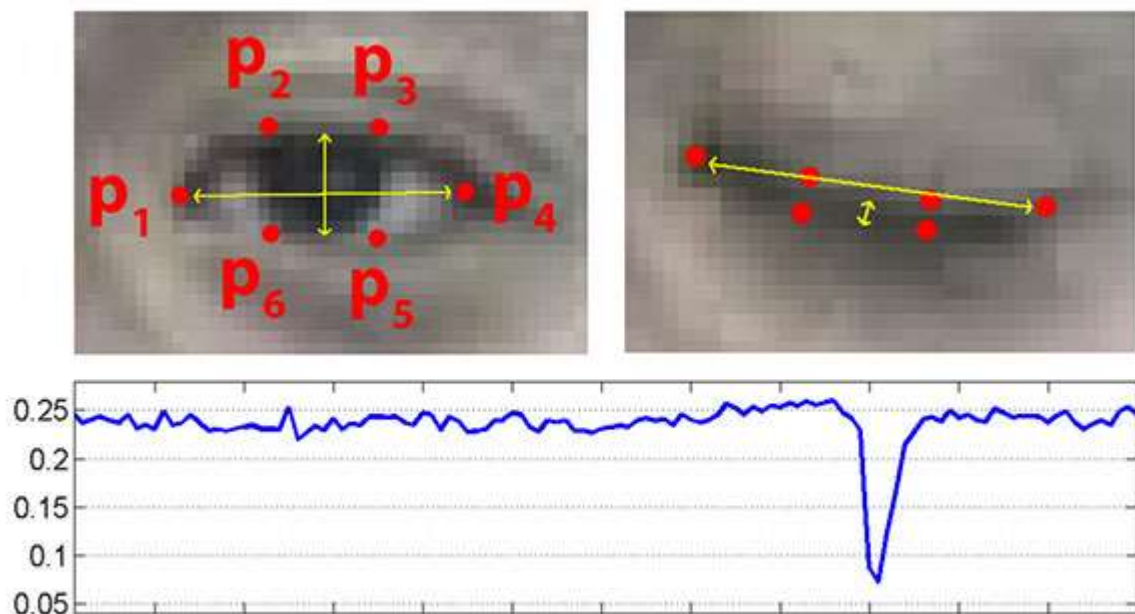
For every video frame,  $p_1$  to  $p_6$  are detected. Based on that, *Eye Aspect Ratio* (EAR) is computed.

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

*Source: pyimagesearch.com*

The numerator of this equation computes the distance between the vertical eye landmarks while the denominator computes the distance between horizontal eye landmarks, weighting the denominator appropriately since there is only *one* set of horizontal points but *two* sets of vertical points.

We observe that the EAR remains almost constant, but rapidly falls when blinking takes place. This can be observed using the following figure by Soukupová and Čech



*Top-left:* A visualization of eye landmarks when the eye is open. *Top-right:* Eye landmarks when the eye is closed. *Bottom:* Plotting the eye aspect ratio over time. The dip in the eye aspect ratio indicates a blink (Figure 1 of Soukupová and Čech).

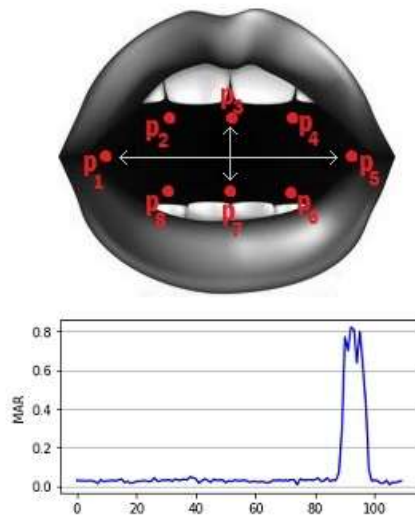
While one can train a classifier to detect the drop, it is easier to use a simple if-condition to detect the blink.

```
if EAR <= SOME_THRESHOLD:
    EYE_STATUS = 'CLOSE'
```

Depending on which eye blinks, different responses will be triggered.

### **-Mouth Closing and Opening**

On the same principle as EAR, a *Mouth Aspect Ratio (MAR)* can be calculated, which will function the same way for all intents and purposes. Instead of 6 points, here 8 points are used

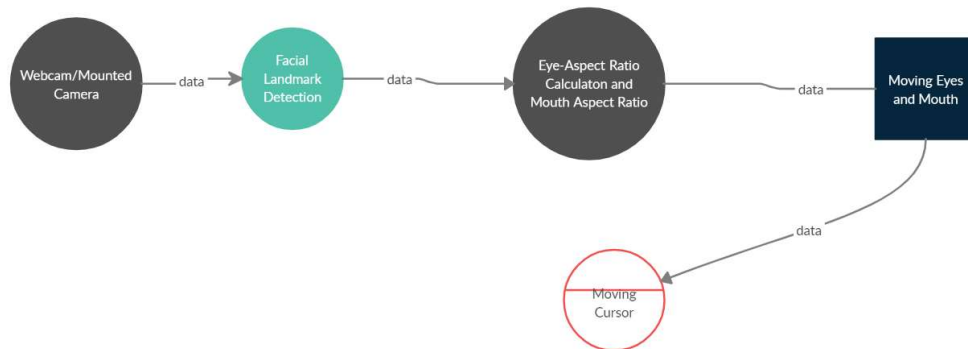


$$\text{MAR} = \frac{\|p_2 - p_8\| + \|p_3 - p_7\| + \|p_4 - p_6\|}{2 \|p_1 - p_5\|}$$

Source: [towardsdatascience.com](https://towardsdatascience.com)

Using these predicted landmarks of the face, we can build appropriate features that will further allow us to detect certain actions, like using the eye-aspect-ratio (more on this below) to detect a blink or a wink, using the mouth-aspect-ratio to detect a yawn etc or maybe even a pout. In this project, these actions are programmed as triggers to control the mouse cursor. *PyAutoGUI* library was used to move the cursor around.

## Context Diagram



## CODE

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

# Thresholds and consecutive frame length for triggering the mouse action.
MOUTH_AR_THRESH = 0.3
MOUTH_AR_CONSECUTIVE_FRAMES = 5
EYE_AR_THRESH = 0.20
EYE_AR_CONSECUTIVE_FRAMES = 5
WINK_AR_DIFF_THRESH = 0.001
WINK_AR_CLOSE_THRESH = 0.20
WINK_CONSECUTIVE_FRAMES = 4
```

```
# 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':

        pag.moveRel(drag, 0)

    elif dir == 'left':

        pag.moveRel(-drag, 0)

    elif dir == 'up':

        if SCROLL_MODE:

            pag.scroll(40)

        else:

            pag.moveRel(0, -drag)

    elif dir == 'down':

        if SCROLL_MODE:

            pag.scroll(-40)

        else:

            pag.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)

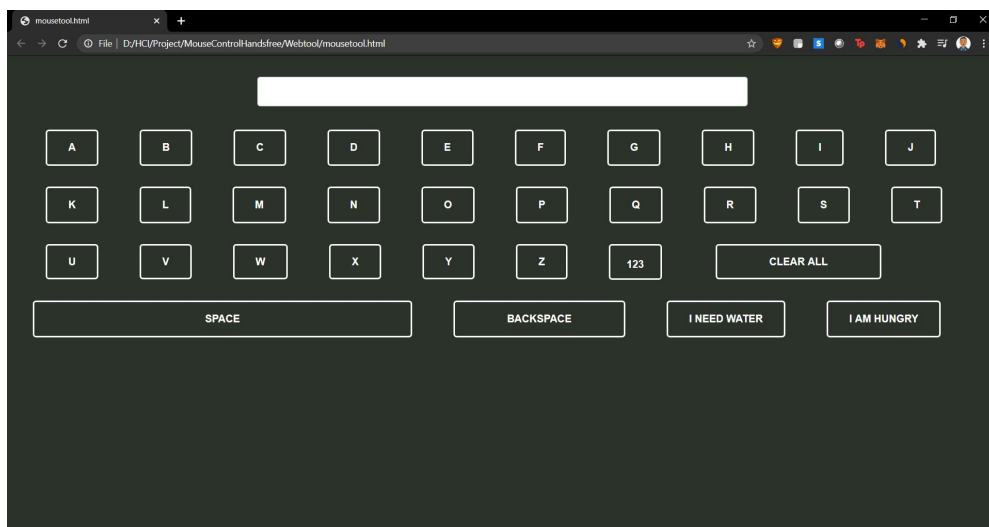
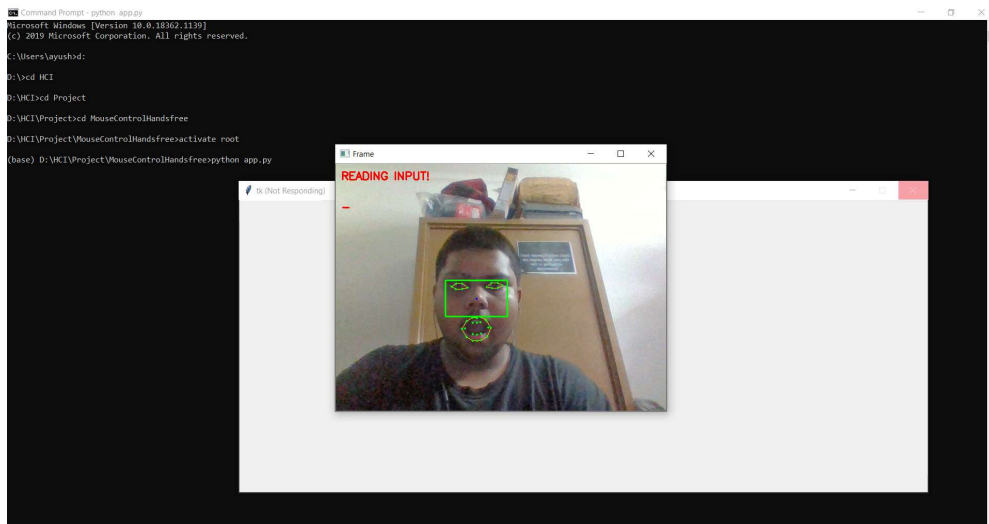
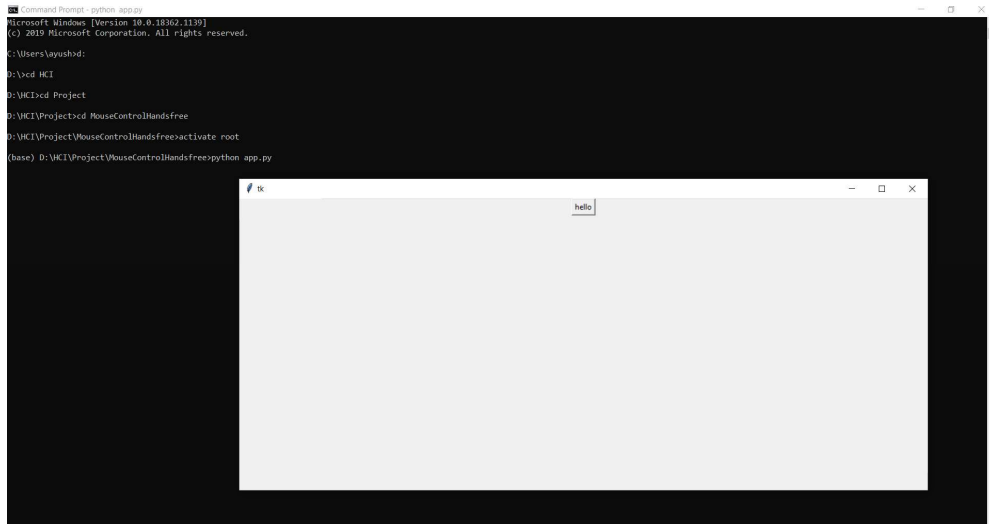
key = cv2.waitKey(1) & 0xFF

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

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

```

# RESULT



The above system works just as well as expected. Average response time is not more than 15-20 milliseconds, in a system with normal specs and no external graphic card. This shows that the system is highly responsive and compatible, with negligible to no lag between input and successive output.

Furthermore, the tkinter app makes it easier to access the system using a GUI interface, instead of a CLI interface. This means that it can be easily used by the technologically challenged. Paired with the webtool interface, this system prototype covers all the bases it aimed to cover.

## **NOVELTY AND INNOVATION**

This project uses existing technology of Computer Vision and Facial Detection to create a system which enables differently abled people to perform the same tasks that we do so easily. Not only does it solve a technical problem, but a social one as well.

The novelty is that unlike existing solutions, this system does not require any extra hardware or attachments. That makes it highly portable and affordable.

This prototype also involves a web-based interface and a tkinter app so that users can easily type, an additional feature to just moving the cursor.

## **CONCLUSION**

This project is going to have a great impact on current worlds situation as there are not many software's to cope with the problems faced by people with some kind of disability. Our software allows many of these people with ease of using the computers without physically touching the mouse as we have tried to make the software in a way that it allows them to interact with the pc with only small movements of their face in place of moving the mouse physically. This is basically focused around using facial landmarks to control the cursor of the mouse. By using specific actions or expressions, we give commands to the computer to move the cursor. Various actions elicit various responses like Mouth Open - Activate/Deactivate Mouse Control, Right Eye Wink - Right Click, Left Eye Wink - Left Click, Squinting - Activate and Deactivate scrolling Head Movement - Cursor movement or Scroll Movement. These were the main aspects of our project.