CHAPTER 1

INTRODUCTION

1.1. INTRODUCTION

Personal computers were initially used for solving mathematical problems and word-processing. In recent years, however, computers have become a necessity for every aspect of our daily activities. These activities range from professional applications to personal uses such as internet browsing, shopping, socializing and entertainment.

In a simple computer system, the interaction between user and computer takes place using devices additional peripheral devices such as mouse and keyboard for inputs and monitors and loudspeakers for output and displaying of information. But the input devices are useless in case of people who are motor-impaired, who cannot move their hands or other parts of their body. But in cases such as amyotrophic lateral sclerosis (ALS) patients have progressive paralysis of the voluntary muscles develops due to loss of motor neurons. This tends to make the patients unable to speak and communication is very much hard for them. Furthermore, this disease may also involve dysphonia or aphonia. Another one such scenario is patients with cerebral palsy where the patient's muscle movement and coordination is disrupted. They are unable to speak and are generally bound to a wheelchair. They also are unable to speak. The computer can be considered as an ideal tool for enabling communication with such people, improving it and making more reliable. The major challenge would be to facilitate the interaction between the user and the computer as they can neither give voice commands nor move their hands. So, we can develop a system where we can track the eye movement of the user to interact with the computer and use it to move the mouse around, which can be used to communicate with the computer or with others. There are some possibilities to acquire information starting from the electrical signal generated by brain (EEG), muscles (EMG) or from the detection of partially active limb micro-movements. Research has been undertaken to use the EEG signal as control of PC operation or other equipment (like an electric wheelchair) which can be combined with PC. There are also some projects involving multiple technologies like EMG and EGT (eye-gaze tracking). Although it is very interesting to operate the machine or computer with thoughts only,

this kind of system demands very stable electromagnetic conditions and they are very sensitive to noise. Detecting eye placement can be used to correct eye movement artefacts in electroencephalogram data. In this case, data coming from the eye tracking device cannot be corrupted by any electrophysiological signal.

The eye gaze mouse enables these motor-impaired patients to control the mouse of a computer by using their gaze. The program uses the head movements of the user to map the movement direction of the mouse and uses left and right winks for the respective clicks. The speed of the mouse also varies as we move away from the centre anchor point. The farther we go the faster the mouse moves. A special scroll mode is also available where the user can scroll easily through a page without the need for clicking the down button on the scrollbar.

1.2. OBJECTIVE

1. To develop a system to recognise the face and eyes of the user and track the eye gaze of the user and map it to the system.
2. Develop an algorithm to map the movements of the eye of the user to the cursor in a more efficiently and accurately.
3. Detect Blinking of the user which would symbolise clicking of the mouse
4. Set up additional hardware, if needed, to enhance the ability and improve the accuracy of the system

CHAPTER 2
LITERATURE SURVEY

Z. Orman, A. Battal, and E. Kemer [1], describes that Face and eye detection is one of the most challenging problems in computer vision area. The goal of this paper is to present a study over the existing literature on face and eye detection and gaze estimation. With the uptrend of systems based on face and eye detection in many different areas of life in recent years, this subject has gained much more attention by academic and industrial area. Many different studies have been performed about-face and eye detection. Besides having many challenging problems like having different lighting conditions, having glasses, facial hair or moustache on face, different orientation pose, or occlusion of face, face and eye detection methods performed great progress. this paper first categorizes face detection models and examine the basic algorithms for face detection. Then they evaluate methods for eye detection and gaze estimation.

V. Raudonis, R. Simutis, and G. Narvydas [2] used principal component analysis (PCA) to find the first six principal components of the eye image to reduce dimensionality problems, which arise when using all image pixels to compare images. Then, an Artificial Neural Network (ANN) is used to classify the pupil position. The training data for ANN is gathered during calibration where the user is required to observe five points indicating five different pupil positions. The system requires special hardware which consists of glasses and a single head-mounted camera and thus might be disturbing to the patient as it is in their field of view. The use of classification slows the system and hence it requires some enhancements to be applicable. Also, the system is not considered a real-time eye-tracking system. The proposed algorithm was not tested on a known database which means the quality of the system might be affected by changes in lighting conditions, shadows, distance of the camera, the exact position in which the camera is mounted, etc. The algorithm required processing which cannot be performed by low computational hardware such as a microcontroller.

M. Mehrubeoglu, L. M. Pham, H. T. Le, R. Muddu, and D. Ryu [3], introduced an eye detection and tracking system that detects the eyes using template matching. The system uses a special customized smart camera which is programmed to continuously track the

user's eye movements until the user stops it. Once the eye is detected, the region of interest (ROI) containing only the eye is extracted to reduce the processed region. From their work, it can be concluded that it is a fast eye-tracking algorithm with acceptable performance. The algorithm could be a nice feature to be added to modern cameras. A drawback is that the experiments were not performed using a database containing different test subjects and conditions, which reduces the reliability of the results. Also, the algorithm located the coordinates but did not classify the eye gaze direction to be left, right, up or down.

Fu and Yang [4], proposed a high-performance eye-tracking algorithm in which two eye templates, one for each eye, are manually extracted from the first video frame for system calibration. The face region in a captured frame is detected and a normalized 2-D cross-correlation is performed for matching the template with the image. Eye gaze direction is estimated by iris detection using edge detection and Hough circle detection. They used their algorithm to implement a display control application. However, it has an inflexible calibration process. The algorithm was not tested on a variety of test subjects and the results were not reported which requires the algorithm to be investigated carefully before choosing to implement it.

Kuo et al. [5], proposed an eye-tracking system that uses the particle filter which estimates a sequence of hidden parameters depending on the data observed. After detecting possible eyes positions, the process of eye tracking starts. For effective and reliable eye tracking, the grey level histogram is selected as the characteristics of the particle filter. Using low-level features in the image makes it a fast algorithm. High accuracy is obtained from the system; however, the real-time performance was not evaluated, the algorithm was tested on images, not videos and the images were not taken from a known database and, thus, the accuracy and performance of the algorithm may decrease when utilized in a real-world application.

A. Kudale et al. [6], explains that as there is great advancement in the technology in recent years, there has been much improvement in various fields of computing such as Human-Computer Interface (HCI), Computer Vision and Perceptual User Interface (PUI). Input to the computers has sensed information about the physical properties of user, places, or things. For example, a computer mouse operates by motion imparted by the user’s hand. Many active researchers have been working on finding an alternative

solution to control computer mouse-like by using finger, eye, hand/palm, marked gloves etc. These techniques may not be suitable to physically disabled people. This paper introduces how the head motion of the user can be used to control the mouse cursor. A way to create an application which replaces the input device mouse by using the face of the user is proposed. Since face/head is a primary part of the human body. They introduced a face detecting system which precisely record the motion parameters from video at real-time. According to the head movement, the cursor is navigated on a computer screen. The mouse clicks are performed only if the user does not move the cursor for a specified time. This method thus ignored the use of other facial features which are difficult to track under certain conditions. For example, in eye-tracking, if the user is wearing spectacles it becomes difficult for the camera to track the movement of eyes in low light condition. This method can be considered as an alternative way of controlling mouse for users with hand disability and can also be used for motion control games and simulation.

J. Tu et al. [7], introduces a novel camera mouse driven by 3D model-based visual face tracking technique. While the camera becomes standard configuration for personal computer (PC) and computer speed becomes faster and faster, achieving human-machine interaction through visual face tracking becomes a feasible solution to hand-free control. The human facial movement can be decomposed into rigid movement, e.g. rotation and translation, and non-rigid movement, such as the open/close of mouth, eyes, and facial expressions, etc. They introduced a visual face tracking system that can robustly and accurately retrieve these motion parameters from video at real-time. After calibration, the retrieved head orientation and translation can be employed to navigate the mouse cursor, and the detection of mouth movement can be utilized to trigger mouse events. 3 mouse control modes are investigated and compared. Experiments in the Windows XP environment verifies the convenience of navigation and operations using our face mouse. This technique can be an alternative input device for people with hand and speech disability and futuristic vision-based game and interface.

M. Betke, J. Gips, and P. Fleming [8] have developed the “Camera Mouse” system to provide computer access for people with severe disabilities. The system tracks the computer user’s movements with a video camera and translates them into the movements of the mouse pointer on the screen. Body features such as the tip of the

user’s nose or finger can be tracked. The authors have used a visual tracking algorithm, based on cropping an online template of the tracked feature from the current image frame and testing where this template correlates in the subsequent frame. The location of the highest correlation was interpreted as the new location of the feature in the subsequent frame. Various body features were examined for tracking robustness and user convenience. A group of 20 people without disabilities tested the Camera Mouse and quickly learned how to use it to spell out messages or play games. Twelve people with severe cerebral palsy or traumatic brain injury tried the system, nine of whom showed success. They interacted with their environment by spelling out messages and exploring the Internet.

Fu and Huang [9], developed novel head tracking driven camera mouse system, called "hMouse" for manipulating hand-free perceptual user interfaces. The system consists of a robust real-time head tracker, head pose/motion estimator, and a virtual mouse control module. For the hMouse tracker, the author proposed a 2D detection/tracking complementary switching strategy with an interactive loop. Based on the reliable tracking results, hMouse calculates the user's head roll, tilt, yaw, scaling, horizontal, and vertical motion for further mouse control. Cursor position is navigated and fine-tuned by calculating the relative position of tracking window in image space and the user's head tilt or yaw rotation. After mouse cursor is navigated to the desired location, head roll rotation triggers virtual mouse button clicks. Experimental results demonstrated that hMouse succeeds under the circumstances of user jumping, extreme movement, large degree rotation, turning around, hand/object occlusion, part face out of camera shooting region, and multiuser occlusion. It provided an alternative solution for convenient device control, which encourages the application of interactive computer games, machine guidance, robot control, and machine access for disabilities and elders.

Siriteerakul et al. [10] proposed an effective method that tracks changes in head direction with texture detection in orientation space using low-resolution video. The head direction was determined using a Local Binary Pattern (LBP) to compare between the texture in the current video frame representing the head image and several textures estimated by rotating the head in the previous video frame by some known angles. The method was used in real-time applications. It could only find the rotation about one axis

while the others are fixed and determined the rotation about the y-axis (left and right rotations) but not neck flexion.

Song et al. [11] introduced the head and mouth tracking method using image processing techniques where the face is first detected using an Adaboost algorithm. Then, head movements were detected by analysing the location of the face. 5 head motions were defined as the basis of head movements. The geometric centre of the detected face area was calculated and considered to be the head's central coordinates. These coordinates could be analysed over time to trace the motion of the head. The used camera was not head-mounted. The method was found to be fast, which makes it applicable in simple applications for people with disabilities. However, the accuracy and performance of the method were not reported.

In a study conducted by Ball, Laura J., David R. Beukelman, and Gary L. Pattee [12] it was revealed that perceptions of communication effectiveness for speakers with ALS were quite similar for the speakers and their frequent listeners across 10 different social situations. ALS speakers and their listeners reported a range of communication effectiveness depending upon the adversity of specific social situations. Which means that they can communicate and understand what the other person is saying.

Zhao et al. [13] presented another head movement detection method based on image processing which also used the Lucas-Kanade algorithm. To identify head movements accurately, the face is first detected, then the nostrils are located and the Lucas-Kanade algorithm is used to track the optical flow of the nostrils. However, the face position in the video may be approximate and, thus, the coordinates of the feature points might not reflect the head movement accurately. Therefore, they used inter-frame difference in the coordinate of feature points to reflect the head movement. This approach is applicable in head motion recognition systems and it is fast. More experiments are required using datasets which contain different subjects and cases to be considered as an option for real applications, especially since it ignored the challenges of different camera angles of view and normal noise challenges.

Xu et al. [14] presented a method which is used to recover 3-D head pose video with a 3-D cross model to track continuous head movement. The model is projected to an initial template to approximate the head and has been used as a reference. Full head

motion can then be detected in video frames using the optical flow method. It has used a camera that is not head-mounted. It can be considered a high complexity algorithm which has been useful in academic research fields and further research can be done based on the findings of the work and possible applications may be suggested.

Cuong and Hoang [15] proposed an efficient approach for real-time eye-gaze detection from images acquired from a web camera. The measured data was enough to describe the eye movement because the web camera is stationary to the head. First, the image has been binarized with a dynamic threshold. Then geometry features of the eye image have been extracted from binary image. Next using an estimation method based on the geometry structure of the eye, the positions of two eye corners have been detected. After that, the centre of iris was detected by matching between an iris boundary model and image contours. Finally, using the relative position information between the centre of iris and the eye corners, based on the relationship between image coordinate and monitor coordinate, the position where the eye is looking at the monitor is calculated. This system required only a low-cost web camera and a personal computer. Experimental results showed that the proposed system could detect accurately eye movements in real-time.

L. E. MacLellan [16], has proposed that Individuals with disabilities, who do not have reliable motor control to manipulate a standard computer mouse, required alternate access methods for complete computer access and communication as well. The Camera Mouse system visually tracks the movements of selected facial features using a camera to directly control the mouse pointer of a computer. Current research has suggested that the system can successfully provide a means of computer access and communication for individuals with motor impairments. However, there were no existing data on the efficacy of the software's communication output capabilities. The goal of the case study was the provision of a comprehensive evaluation of Camera Mouse as a computer access method for Augmentative and Alternative Communication (AAC) for an individual with cerebral palsy, which has been preferred to use unintelligible dysarthric speech to communicate the desires and thoughts despite having access to a traditional AAC system.

Królak and Stumillo [17], proposed a vision-based human-computer interface. The interface detected voluntary eye-blinks and interpreted them as control commands. The

employed image processing methods include Haar-like features for automatic face detection, and template matching based eye tracking and eye-blink detection. Interface performance has been tested by 49 users (of which 12 were with physical disabilities). Test results indicated interface usefulness in offering an alternative means of communication with computers. The users have entered English and Polish text (with an average time of less than 12s per character) and have been able to browse the Internet. The interface was based on a notebook equipped with a typical web camera and requires no extra light sources. The interface application is available on-line as open-source software.

A. Erdem, E. Erdem, Yardimci, Atalaya and Cetin [18], describes a computer vision-based mouse, which can control and command the cursor of a computer or a computerized system using a camera. In order to move the cursor on the computer screen, the user simply moves the mouse-shaped passive device placed on a surface within the viewing area of the camera. The video generated by the camera is analyzed using computer vision techniques and the computer moves the cursor according to mouse movements. The computer vision-based mouse has regions corresponding to buttons for clicking. To click a button the user simply covers one of these regions with his/her finger.

Robertson, Laddaga and Kleek [19], developed a vision-based virtual mouse interface is described that utilizes a robotic head, visual tracking of the users head and hand positions and recognition of user hand signs to control an intelligent kiosk. The user interface supports, among other things, smooth control of the mouse pointer and buttons using hand signs and movements. The algorithms and architecture of real-time vision and robot controller are described.

Lupu, Ungureanu and Siriteanu [20], explains that in many cases, persons with neuro-locomotor disabilities have a good level of understanding and should use their eyes for communication. In this paper, a reliable, mobile and low-cost system based on eye-tracking mouse is presented. The eye movement is detected by a head-mounted device and consequently, the mouse cursor is moved on the screen. A click event denoting a pictogram selection is performed if the patient gazes a certain time the corresponding image on the screen.

Rotariu, Costin, Cehan and Morancca [21], developed a system to build an electro-informatics system that allows a special communication (through a video interface and radio links) with severely handicapped persons who cannot interfere with real-world by normal ways like speech, writing or language signs. Yet, these persons can see and/or can hear and they understand the meaning of different messages. The TELPROT system is made by two main functional components: the patient equipment and the caretaker equipment. Both sub-systems are composed of a control unit and a communication unit. The patient and nurse communicate through radio waves, within a dedicated LAN.

Agustin, Mollenbach, Barret, Tall and Hansen [22], presents a low-cost gaze tracking system that is based on a webcam mounted close to the user's eye. The performance of the gaze tracker was evaluated in an eye-typing task using two different typing applications. Participants could type between 3.56 and 6.78 words per minute, depending on the typing system used. A pilot study to assess the usability of the system was also carried out in the home of a user with severe motor impairments. The user successfully typed on a wall-projected interface using his eye movements.

Lupu, Ungureanu and Bozomitu [23], presented a new technology to communicate with people with neuro-locomotor disabilities using embedded systems and eye-tracking approach. The eye movement is detected by a special device and the voluntary eye blinking is correlated with a pictogram or keyword selection reflecting patient's needs. The implemented eye-tracking method uses image processing technique based on the binarization algorithm.

Bozomitu, Barabasa, Cehan and Lupu [24], the hardware component of a new technology used to communicate with people with major neuro-locomotor disability using ocular electromyogram is presented. The signals produced by the ocular muscle, provided by five electromyogram sensors are amplified used a new instrumentation amplifier and then processed by the software component of the proposed communication technology. The information obtained after these signals are processed is used in order to move a pointer on a display in concordance to the patient gaze direction. In this way, the needs of the patient are sent to the caretaker by detection of voluntary blinking.

Hori, Sakano and Saitoh [25], presented a communication interface controlled by eye movements and voluntary eye blink has been developed for disabled individuals who have motor paralysis and therefore cannot speak. Horizontal and vertical electrooculogram were measured using two surface electrodes referring to an earlobe electrode. Four directional cursor movements and one selection were realized by logically combining the detected two-channel signals. Virtual input experiments were conducted on a virtual screen keyboard. Its usability and accuracy were improved using our proposed method.

Betke, Gips and Fleming [26], presented the "Camera Mouse" system has been developed to provide computer access for people with severe disabilities. The system tracks the computer user's movements with a video camera and translates them into the movements of the mouse pointer on the screen. Body features such as the tip of the user's nose or finger can be tracked. The visual tracking algorithm is based on cropping an online template of the tracked feature from the current image frame and testing where this template correlates in the subsequent frame. The location of the highest correlation is interpreted as the new location of the feature in the subsequent frame. Various body features are examined for tracking robustness and user convenience. A group of 20 people without disabilities tested the Camera Mouse and quickly learned how to use it to spell out messages or play games. Twelve people with severe cerebral palsy or traumatic brain injury have tried the system, nine of whom have shown success. They interacted with their environment by spelling out messages and exploring the Internet.

Lupu, Bozomitu, V. Cehan and D. Cehan [27], presents a new technology used for communicating with people with major neuro-locomotor disability by using ocular electromyogram (electrooculogram). The electromyogram is obtained with 5 electrodes placed on the ocular muscles which catch the signals produced by the muscles when the eyes move up and down or left and right. The proposed communication technology ensures communication with the patient as follows: the patient sees on a screen key words or ideograms displayed in different areas; the focusing of the patient's gaze involves eye movement which is seized by the electromyogram; the selection of the ideogram is done by voluntary blinking and captured by the electromyogram as an

impulse. The wish or need of the patient is then sent to the caretaker using a wire or a wireless transmission system. The absence of a selection within a certain time period causes the displaying of a new set of ideograms.

Lupu, Bozomitu, Ungureanu and Cehan [28], presents a new technology used for communicating with people with major neuro-locomotor disability by determining gaze direction on a monitor screen. Gaze direction is determined by pupil position through image analysis. The proposed communication technology ensures communication with the patient as follows: (1) on a monitor screen divided into four sections are displayed keywords and corresponding ideograms; (2) the patient sees the keywords and focuses his/her gaze on a certain section - a pointer allows the control of gaze direction; (3) an infra-red mini/micro-video camera placed on the patient's head sends the image of the eye to a computer; (4) the focusing of the patient's gaze on a certain keyword i.e. a certain section of the screen is identified by image analysis; (5) voluntary blinking, also objectified by image analysis, shows the selection of a certain word. The patient's wish or need, represented by the keyword, is then sent to a server and from there to a caretaker by means of a wire or a wireless transmission system. The absence of a selection within a certain time period causes the displaying of a new set of ideograms. The new technology is useful in the case of the patients who cannot communicate verbally, through signs or in writing and is based on their ability to control eye movement, which is often the case with people with major locomotor disabilities.

P. Viola and M. Jones [29] describe a machine learning approach for visual object detection which is capable of processing images extremely rapidly and achieving high detection rates. This work is distinguished by three key contributions. The first is the introduction of a new image representation called the “Integral Image” which allows the features used by our detector to be computed very quickly. The second is a learning algorithm, based on AdaBoost, which selects a small number of critical visual features from a larger set and yields extremely efficient classifiers. The third contribution is a method for combining increasingly more complex classifiers in a “cascade” which allows background regions of the image to be quickly discarded while spending more computation on promising object-like regions. The cascade can be viewed as an object-specific focus-of-attention mechanism which unlike previous approaches provides statistical guarantees that discarded regions are unlikely to contain the object of interest.

In the domain of face detection, the system yields detection rates comparable to the best previous systems. Used in real-time applications, the detector runs at 15 frames per second without resorting to image differencing or skin colour detection.

A real-time algorithm to detect eye blinks in a video sequence from a standard camera is proposed by Soukupova, Tereza, and Jan Cech [30]. Recent landmark detectors trained on in-the-wild datasets exhibit excellent robustness against face resolution, varying illumination and facial expressions. We show that the landmarks are detected precisely enough to reliably estimate the level of the eye openness. The proposed algorithm, therefore, estimates the facial landmark positions, extracts a single scalar quantity – eye aspect ratio (EAR) – characterizing the eye openness in each frame. Finally, blinks are detected either by an SVM classifier detecting eye blinks as a pattern of EAR values in a short temporal window or by hidden Markov model that estimates the eye states followed by a simple state machine recognizing the blinks according to the eye closure lengths. The proposed algorithm has comparable results with the state-of-the-art methods on three standard datasets.

Table 2.1: Analysis Table

Author	Work Done	Method
Raudonis, Simutis, and Narvydas [2]	Find the first 6 principal components of the eye image to reduce dimensionality problems, which arise when using all image pixels to compare images	Principle Component Analysis (PCA)
Mehrubeoglu et al [3]	Introduced an eye detection and tracking system that detects the eyes.	Template Matching using ROI
Fu and Yang [4]	High-performance eye-tracking algorithm in which two eye templates are extracted from the video frame.	Edge detection and Hough circle detection on video frames.
Kuo et al [5]	Eye-tracking system that which estimates a sequence of hidden	Using low level features and particle filters.

	parameters depending on the data observed.	
J. Tu et al. [7]	Developed a Camera Mouse	3D model based visual face tracking technique
Fu and Huang [9]	"hMouse", a camera driven mouse system was developed.	2D detection complementary switching strategy with an interactive loop
Siriteerakul et al. [10]	an effective method that tracks changes in head direction with texture detection in orientation space using low-resolution video	Local Binary Pattern to compare between the texture in the current and the previous video frame.
Song et al [11]	The head and mouth tracking method using image processing techniques where the face is first detected	Adaboost Algorithm
Zhao et al [12]	Head movement detection method based on image processing	Lucas-Kanade algorithm
Królak and Stumillo [17]	A vision-based human-computer interface is presented	Haar-like and template match eye tracking and blink detection.
Viola and Jones [29]	Described a machine learning approach for visual object detection which processed images at high speed and achieving high detection rates.	Integreal Image, Adaboost and classifiers in a cascade.
Soukupova, Tereza, and Jan Cech [30]	A real-time algorithm to detect eye blinks in a video sequence from a standard camera is proposed.	Estimate Facial Landmarks and extract the Eye Aspect Ratio and Blinks by SVM classifier and Markov Model.

CHAPTER 3

EYE GAZE MOUSE

The eye gaze mouse enables these motor-impaired patients to control the mouse of a computer by using their gaze. The program uses the head movements of the user to map the movement direction of the mouse and uses left and right winks for the respective clicks. The speed of the mouse also varies as we move away from the centre anchor point. The farther we go the faster the mouse moves. A special scroll mode is also available where the user can scroll easily through a page without the need for clicking the down button on the scrollbar. The working of the eye gaze mouse is explained below.

3.1. HARDWARE REQUIREMENTS

The eye gaze mouse requires only a webcam for detection of the head movement. The webcam can be inbuilt into the computer or an external webcam (fig 3.1.) can also be used. External webcams are more preferred as they provide a clearer and higher quality images with better contrast as compared to the inbuilt webcams of the computers.

Fig 3.1. External Webcam

Apart from the webcam, no additional hardware is needed for the working of this program on a computer

3.2. Libraries Used

To make this project the following python libraries were used:

3.2.1. OPENCV

Fig 3.2. OpenCV

OpenCV-Python (fig 3.2.) is a library of Python bindings designed to solve computer vision problems.

OpenCV-Python makes use of NumPy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from NumPy arrays. This also makes it easier to integrate with other libraries that use NumPy such as SciPy and Matplotlib.

OpenCV-Python provides various functions which allows us to make models and train them to recognise special objects or work upon images. OpenCV-Python is used in this project to capture and operate on images from the webcam. The OpenCV functions allowed us to separate each part of an image one by one and select the eye out of a face and based on that we could figure out how much the eye has moved and how much the mouse must be moved.

3.2.2. DLIB

Fig 3.3. Dlib

Dlib (Fig 3.3.) is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real-world problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high-performance computing environments.

Dlib is used in this project to identify faces and filter out eyes using its inbuilt algorithms. The face detector is made using ‘get_frontal_face_detector()’ function and to predict the location of the eyes ‘shape_predictor()’ function is used along with "shape_predictor_68_face_landmarks.dat" which is a pre-trained model which can separate the facial features on a face easily with 90% accuracy.

Since dlib is a C++ based library Cmake is needed to convert the python code into C++.

3.2.3. SciPy.Spatial

Fig 3.4. SciPy

The scipy.spatial (Fig 3.4.) package can compute Triangulations, Voronoi Diagrams and Convex Hulls of a set of points, by leveraging the Qhull library. Moreover, it contains KDTree implementations for nearest-neighbour point queries and utilities for distance computations in various metrics.

We are using the distance package of the SciPy.Spatial to find how much distance the mouse cursor is to be moved from our anchor point.

3.2.4. Pyglet

Fig 3.5. Pyglet

Pyglet (Fig 3.5.) is a powerful, yet easy to use Python library for developing games and other visually-rich applications on Windows, Mac OS X and Linux. It supports windowing, user interface event handling, Joysticks, OpenGL graphics, loading images and videos, and playing sounds and music.

Pyglet is used in this project to provide sound assistance to the user while using the mouse. Such as which mode it is in or a click is registered or not.

3.2.5. Pynput

Pynput is the most important part of this program as pynput is the library which allows us to control the mouse. Pynput helps us to control any of the compatible input devices which are normally connected to a computer such as a mouse and keyboard.

3.3. LIMITATION OF PREVIOUS METHODS

The previous methods used to implement this kind of applications were either very much complicated, inefficient or required too much hardware. There were two kinds of existing methods for developing an eye gaze mouse:

3.3.1. Mouse movement using pupil detection

In this method, a webcam mounted on a computer screen or the inbuilt webcams are used to detect the eye movement and map them to the mouse. But this method is inefficient when it comes to upwards or downwards movement as no white region of the eye can be detected properly and we cannot determine the movement of the eye.

3.3.2. Additional headgear

In this method, an additional headgear is mounted on the user to track their eye movement up close to avoid the issues from the first method. But this additional hardware can put strain on the user's eye, can block the vision and brings up the cast of the product as well.

Our method aims to eliminate these two problems by incorporating the head movements as well to the first method for ease in operation.

3.4. WORKING OF EYE GAZE MOUSE

3.4.1. Video Processing

As we are working on a video in this project, the first step is to collect frames from the video. Further, each frame is individually processed. We are live streaming the video. We use the OpenCV library to perform the task. To capture a video, we need to create a VideoCapture object. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera. After that, we can capture frame-by-frame. At the end, release the capture.

```
import cv2
# capture the video
cap = cv2.VideoCapture(“file path”)


while(True):
    # Capture frame-by-frame
    ret, frame = cap.read()
     # Display the frame
    cv2.imshow('frame',frame)
    # press q to exit
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()
```

UNDERSTANDING OF THE CODE:

cap.read returns a bool (True/False). If frame is read correctly, it will be True. So you can check end of the video by checking this return value.

cv2.imshow used to display an image in a window.

cv2.waitkey(1) It is used to introduce a delay of n milliseconds while rendering images to windows. When used as cv2.waitkey(0) it returns the key pressed by the user on the active window.

To save the result we use a VideoWriter object. We should specify the output file name (eg: output.avi). Then we should specify the FourCC code. Then the number of frames

per second (fps) and frame size should be passed. FourCC is a 4-byte code used to specify the video codec.

3.4.2. Image Conversion to Grayscale

Each frame of the video capture is converted to cv2.COLOR_BGR2GRAY for processing (fig 3.6.).

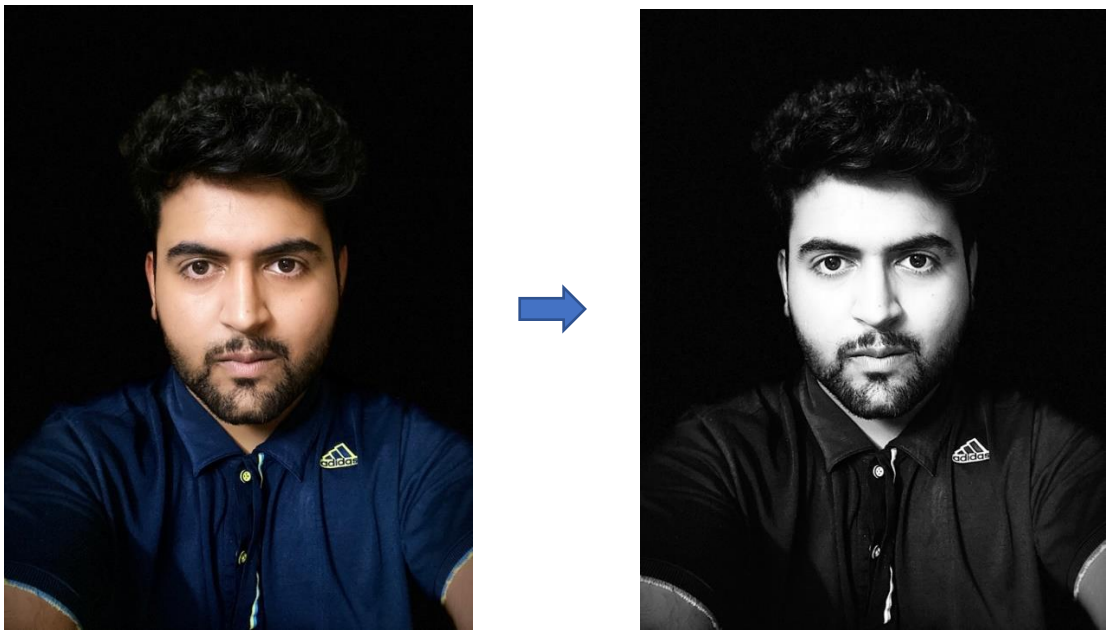

Fig 3.6. Before and after image conversion to Grayscale

A grayscale or greyscale image is one in which the value of each pixel is a single sample representing only an amount of light, that is, it carries only intensity information. Grayscale images, a kind of black-and-white or grey monochrome, are composed exclusively of shades of grey. The contrast ranges from black at the weakest intensity to white at the strongest.

1. Signal to noise. For many applications of image processing, colour information doesn't help us identify important edges or other features. There are exceptions. If there is an edge (a step-change in pixel value) in hue that is hard to detect in a grayscale image, or if we need to identify objects of known hue (orange fruit in front of green leaves), then colour information could be useful. If we don't need colour, then we can consider it noise.
2. Complexity of the code. If you want to find edges based on luminance AND chrominance, you've got more work ahead of you. That additional work (and

additional debugging, additional pain in supporting the software, etc.) is hard to justify if the additional colour information isn't helpful for applications of interest.

3. Multichannel processing rather than starting with full-colour imaging and missing all the important insights that can be learned from single-channel processing.
4. Difficulty of visualization. In grayscale images, the watershed algorithm is easy to conceptualize because we can think of the two spatial dimensions and one brightness dimension as a 3D image with hills, valleys, catchment basins, ridges, etc. In RGB, HSI, Lab, and other colour spaces this sort of visualization is much harder since there are additional dimensions that the standard human brain can't visualize easily
5. Colour is complex. Humans perceive colour and identify colour with deceptive ease but processing coloured images can be long and tidy process.
6. Speed. With modern computers, and with parallel programming, it's possible to perform simple pixel-by-pixel processing of a megapixel image in milliseconds. Facial recognition, OCR, content-aware resizing, mean shift segmentation, and other tasks can take much longer than that. Whatever processing time is required to manipulate the image or squeeze some useful data from it, most customers/users want it to go faster. If we make the hand-wavy assumption that processing a three-channel colour image takes three times as long as processing a grayscale image--or maybe four times as long, since we may create a separate luminance channel--then that's not a big deal if we're processing video images on the fly and each frame can be processed in less than 1/30th or 1/25th of a second. But if we're analysing thousands of images from a database, it's great if we can save ourselves processing time by resizing images, analysing only portions of images, and/or eliminating colour channels we don't need. Cutting processing time by a factor of three to four can mean the difference between running an 8-hour overnight test that ends before you get back to work and having your computer's processors pegged for 24 hours straight.

3.4.3. Face Detection using HAAR CASCADE CLASSIFIER

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features proposed by Paul Viola and

Michael Jones [29] in 2001. It is a machine learning-based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

The algorithm has four stages:

1. Haar Feature Selection
2. Creating Integral Images
3. Adaboost Training
4. Cascading Classifiers

It is well known for being able to detect faces and body parts in an image but can be trained to identify almost any object.

Let's take face detection as an example. Initially, the algorithm needs a lot of positive images of faces and negative images without faces to train the classifier. Then we need to extract features from it.

First step is to collect the Haar Features (fig 3.7, 3.8, 3.9). A Haar feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums.

Fig 3.7. Edge features

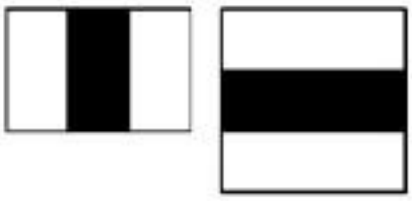

Fig 3.8. Linear features

Fig 3.9. Four rectangle features

But among all these features we calculated, most of them are irrelevant. So how do we select the best features out of 160000+ features? This is accomplished using a concept

called Adaboost which both selects the best features and trains the classifiers that use them. This algorithm constructs a “strong” classifier as a linear combination of weighted simple “weak” classifiers. The process is as follows.
During the detection phase, a window of the target size is moved over the input image, and for each subsection of the image and Haar features are calculated. You can see this in action in the video below. This difference is then compared to a learned threshold that separates non-objects from objects. Because each Haar feature is only a "weak classifier" (its detection quality is slightly better than random guessing) many Haar features are necessary to describe an object with enough accuracy and are therefore organized into *cascade classifiers* to form a strong classifier.

The cascade classifier consists of a collection of stages, where each stage is an ensemble of weak learners. The weak learners are simple classifiers called *decision stumps*. Each stage is trained using a technique called boosting. *Boosting* provides the ability to train a highly accurate classifier by taking a weighted average of the decisions made by the weak learners.

Each stage of the classifier labels the region defined by the current location of the sliding window as either positive or negative. *Positive* indicates that an object was found and *negative* indicates no objects were found. If the label is negative, the classification of this region is complete, and the detector slides the window to the next location. If the label is positive, the classifier passes the region to the next stage. The detector reports an object found at the current window location when the final stage classifies the region as positive.

The stages are designed to reject negative samples as fast as possible. The assumption is that most windows do not contain the object of interest. Conversely, true positives are rare and worth taking the time to verify.

- A *true positive* occurs when a positive sample is correctly classified.
- A *false positive* occurs when a negative sample is mistakenly classified as positive.
- A *false negative* occurs when a positive sample is mistakenly classified as negative.

To work well, each stage in the cascade must have a low false-negative rate. If a stage incorrectly labels an object as negative, the classification stops, and you cannot correct the mistake. However, each stage can have a high false-positive rate. Even if the

detector incorrectly labels a nonobject as positive, you can correct the mistake in subsequent stages. Adding more stages reduces the overall false-positive rate, but it also reduces the overall true positive rate.

Cascade classifier training requires a set of positive samples and a set of negative images. You must provide a set of positive images with regions of interest specified to be used as positive samples. You can use the Image Labeller to label objects of interest with bounding boxes. The Image Labeller outputs a table to use for positive samples. You also must provide a set of negative images from which the function generates negative samples automatically. To achieve acceptable detector accuracy, set the number of stages, feature type, and other functional parameters.

3.4.4. Facial landmarks localization

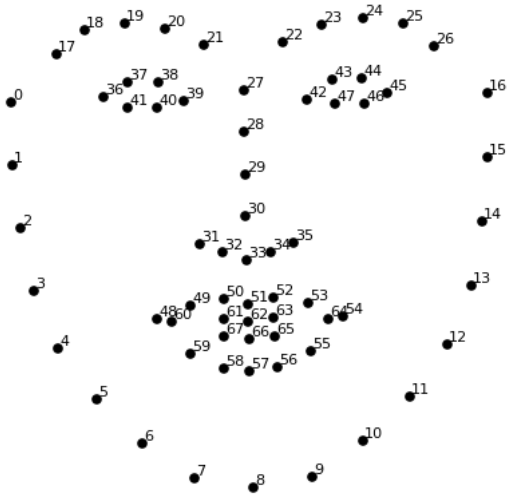

Fig 3.10. Facial markers

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 (fig 3.10.) of interest along the shape.

Facial landmarks are used to localize and represent salient regions of the face, such as:

- Eyes
- Eyebrows
- Nose
- Mouth

- Jawline

3.4.5. Application Flow Detection

On the basis of the model and application flow detection can be any one of the following

1) Eye-tracking to give direction for mouse

2) Eyeball tracking left and right as an extra

3) Blink tracking as an input

4) Left Blink and Right Blink for left-click and right-click

3.4.4.1. Eye-tracking to give direction for mouse

Every time mouse mode or scroll mode is activated two variables are defined using:

anchor_pointx=int((landmarks.part(36).x+landmarks.part(39).x)/2)

anchor_pointy=int((landmarks.part(37).y+landmarks.part(41).y)/2)

These acts as XY coordinates for the anchor point from where a vector to the eye is drawn and using this direction and magnitude for the mouse is calculated. Euclidean Distance from the anchor point to the eye is calculated and scaled down to integer speed. Values for mouse. This process is repeated for each frame giving mouse dynamic speed and smooth movement (fig 3.11.).

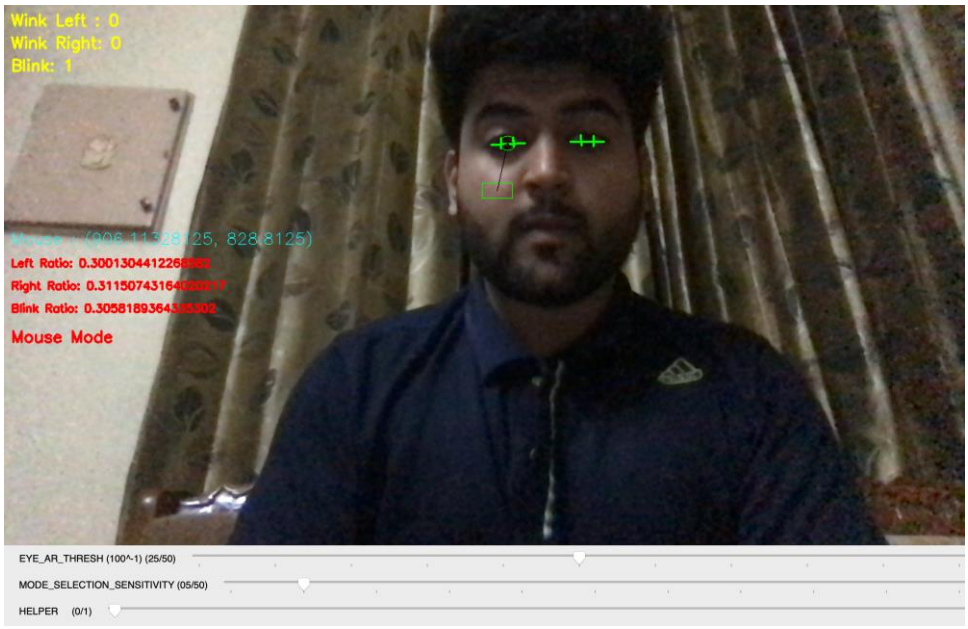

Fig 3.11. Eye-tracking to give mouse direction

3.4.4.2. Eyeball tracking left and right as an extra input

By converting the image into grayscale format, we will see that the pupil is always darker than the rest of the eye. No matter where the eye is looking at and no matter what colour the sclera of the person is.

From the threshold, we find the contours. And we simply remove all the noise selecting the element with the biggest area (which is supposed to be the pupil) and skip all the rest.

The idea is to split the eye into two parts (fig 3.12.) and to find out in which of the two parts there is more sclera visible.

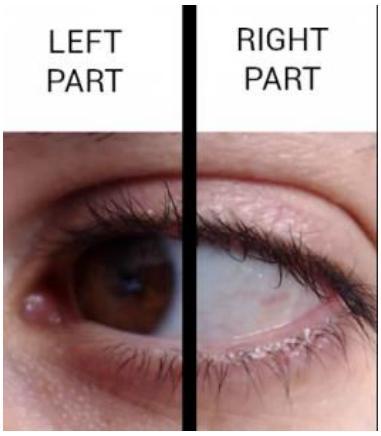

Fig 3.12. Two parts of the eye

If the sclera is more visible on the right part, so the eye is looking at the left (our left) like in this case and vice-versa (fig 3.13.-3.14.)

Fig 3.13. Eye looking right

Fig 3.14. Eye looking left

Whenever we are looking left or right one half of the eye has a lot more white portion than the other half, we calculate the white portion in both halves and use this to calculate Eye Gaze Ratio, Threshold foe this ratio is set to 1.7 less than 1.7 is left and more than 1.7 is right.

We use this as extra input to switch between scroll mode and mouse mode.

Looking to left for a certain no of frame switches to mouse mode and looking right switches to scroll mode. Threshold for frames can be adjusted in the menu. A helper frame is also available which shows different colours to assist the user in verifying he is looking in the correct direction. On successful mode selection, a sound is also generated according to mode to acknowledge mode selection.

3.4.4.3. Blink Detection

We are computing a metric called the *eye aspect ratio* (EAR), introduced by Soukupová and Čech [30] in 2016.

Unlike traditional image processing methods for computing blinks which typically involve some combination of:

1. Eye localization.
2. Thresholding to find the whites of the eyes.
3. Determining if the "white" region of the eyes disappears for a period (indicating a blink).

The eye aspect ratio is instead a *much more elegant solution* that involves a *very simple calculation* based on the ratio of distances between facial landmarks of the eyes (fig 3.15.).

This method for eye blink detection is fast and efficient

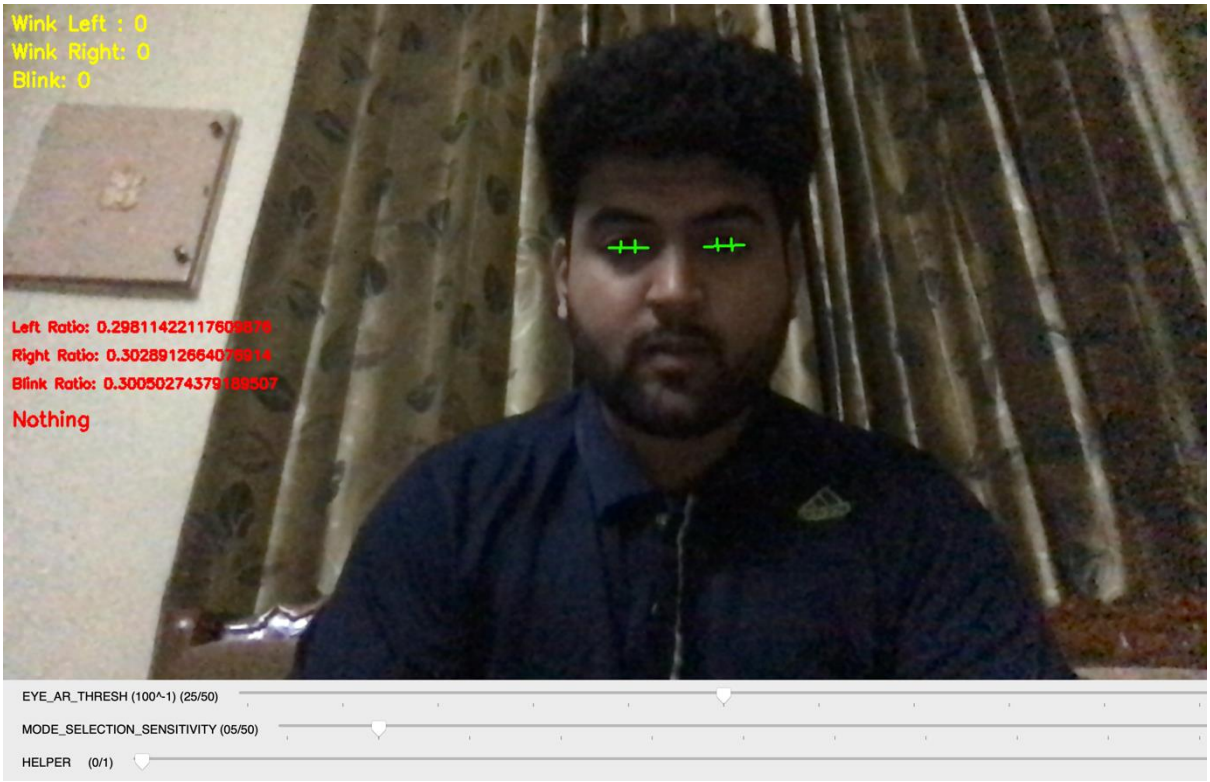

Fig 3.15. Eye Ratios

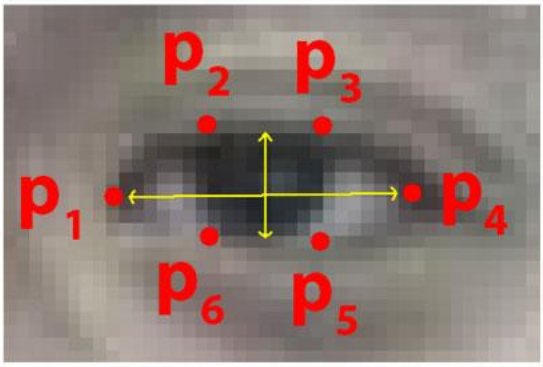

Fig 3.16. points to find out the white region

Each eye is represented by 6 *(x, y)*-coordinates (fig 3.16.), starting at the left corner of the eye (as if you were looking at the person), and then working clockwise around the remainder of the region:

There is a relation between the *width* and the *height* of these coordinates.

Based on the work by Soukupová and Čech in their 2016 paper, *Real-Time Eye Blink Detection using Facial Landmarks*, we can then derive an equation that reflects this relation called the *eye aspect ratio* (EAR) (fig 3.17.):

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

Fig 3.17. EAR formula

Where *p1, …, p6* are 2D facial landmark locations.

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.

the eye aspect ratio is approximately constant while the eye is open but will rapidly fall to zero when a blink is taking place.

3.4.4.4. Left wink and Right wink for left and right-click:

EAR is calculated for both eye and following algorithm is for blink left blink and right blink distinguishing:

```
if left_eye_ratio<EYE_AR_THRESH and right_eye_ratio<EYE_AR_THRESH:
        COUNTER_BLINK += 1
    # otherwise, the eye aspect ratio is not below the blink
    # threshold
    else:
        # if the eyes were closed for an enough time
        # then increment the total number of blinks
        if COUNTER_BLINK >= EYE_AR_CONSEC_FRAMES:
            TOTAL_BLINK += 1
```

```
                sound.play()
                mode_detect=not mode_detect


            # reset the eye frame counter
            COUNTER_BLINK = 0
        if left_eye_ratio < EYE_AR_THRESH and right_eye_ratio > EYE_AR_THRESH:
                COUNTER_LEFT += 1
        else:
            if COUNTER_LEFT >=2:
                TOTAL_LEFT += 1
                print("Left eye winked")
                COUNTER_LEFT = 0
        if right_eye_ratio < EYE_AR_THRESH and left_eye_ratio > EYE_AR_THRESH:
            COUNTER_RIGHT += 1
        else:
            if COUNTER_RIGHT >= 2:
                TOTAL_RIGHT += 1
                print("Right eye winked")
                COUNTER_RIGHT = 0
```

3.4.6. Controls:

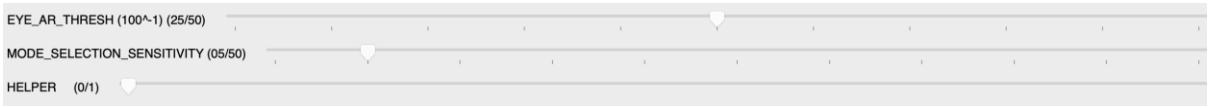

Fig 3.18. Controls

There are 3 controls in this application (fig 3.18.):

1. EYE_AR_THRESH can be adjusted according to the lighting and eye size with given options.
2. Second options give controls to the number of frames required while mode selection
3. Third option is used to activate or deactivate helper frame is eyeball tracking.

CHAPTER 4

RESULTS

In this project, we researched various ways of camera mouse techniques. We identified the problem domain and an alternative solution to it by setting up additional hardware. It can be observed that our approach yields accurate centre estimations not only for images containing dominant pupil but can adjust the threshold according to the size of eye and lighting. We proposed to use face/head motion to control the movement of the mouse cursor. This demonstrates the robustness and proves that our approach can successfully deal with several severe problems that arise in realistic scenarios. Then, the gradient orientations of the pupil and the iris are affected by "noise" and hence their contribution to the sum of squared dot products. The estimated centres are depicted by red squares.

The eye movement tracking using HAAR features have been shown below:

1. Moving up (Fig 4.1.)
2. Looking left (Fig 4.2.)
3. Centre position (Fig 4.3.)
4. Moving right (Fig 4.4.)
5. Moving down (Fig 4.5.)
6. Moving left (Fig 4.6.)

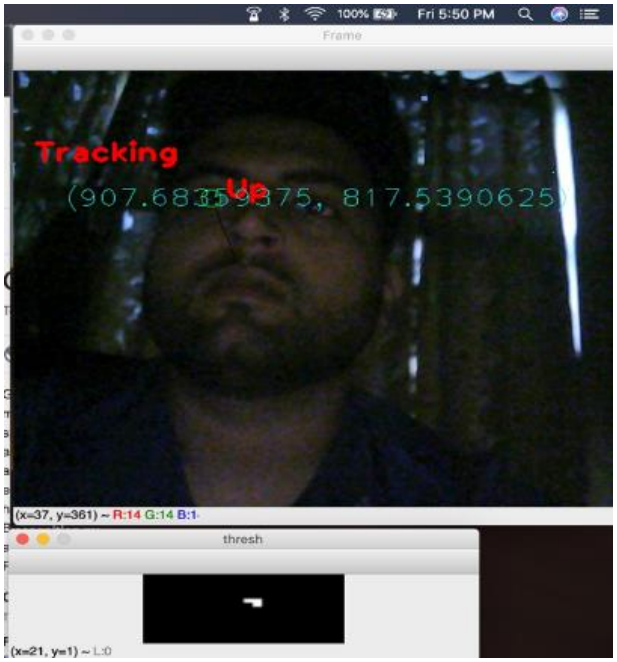

Fig 4.1. Moving up tracking

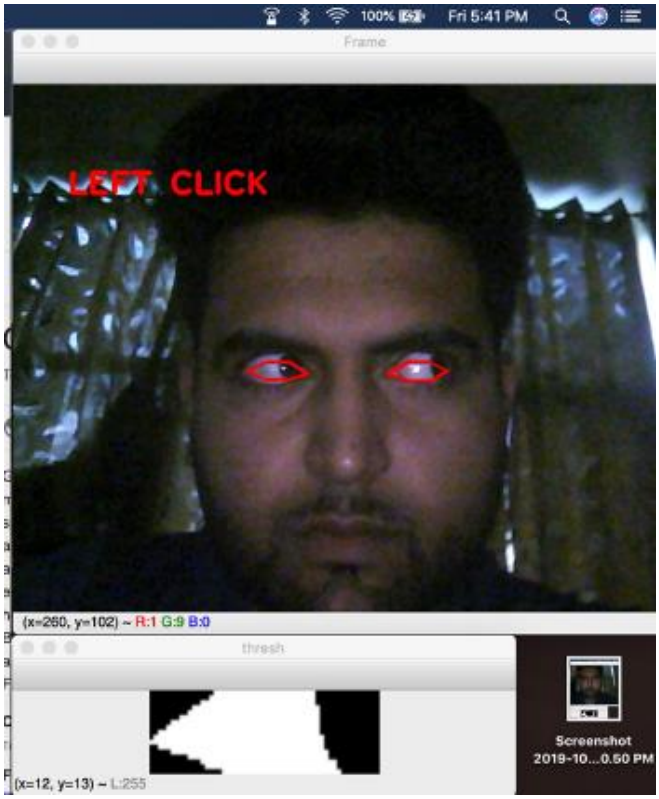

Fig 4.2. Looking left

Fig 4.3. Looking centre

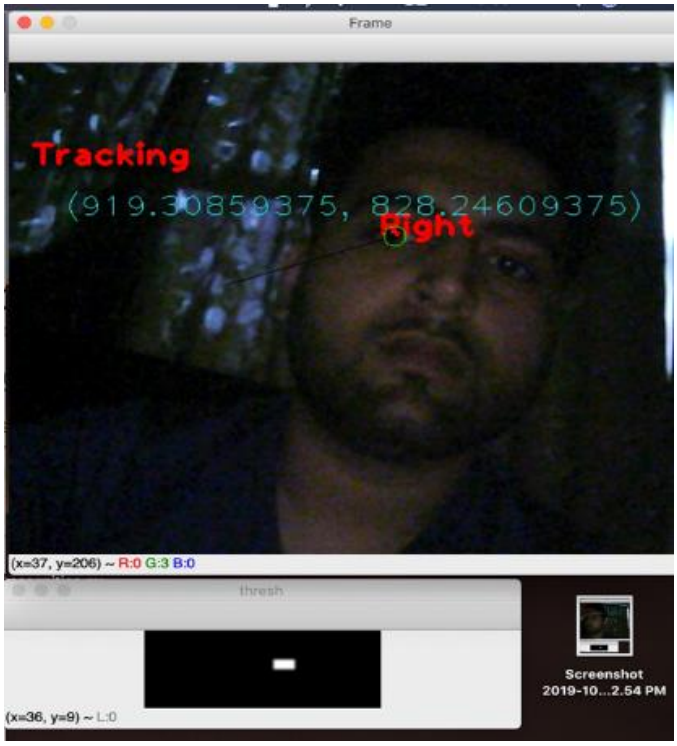

Fig 4.4. Moving right tracking

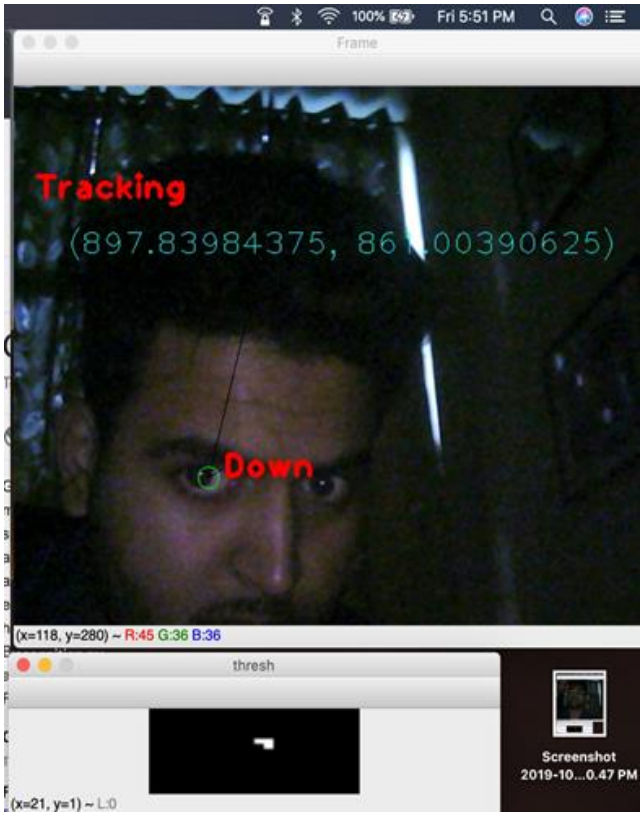

Fig 4.5. Moving down tracking

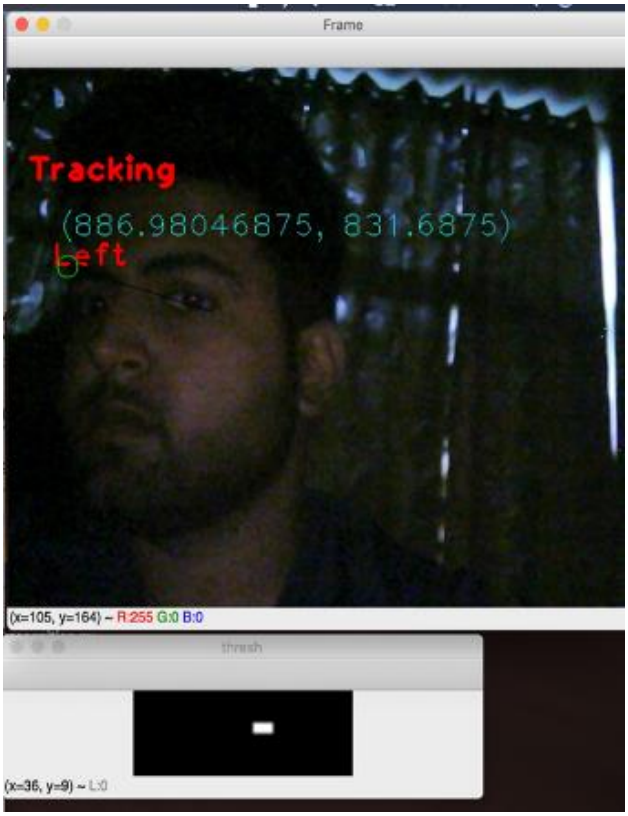

Fig 4.6. Moving left tracking

Note that the estimated centres might be difficult to identify due to low contrast.

The quantitative results of the proposed method defined using the standards of the normalised error using the equation

$$e \leq 1/ d \max (el, er)$$

where el, er is the Euclidean distances between the estimated and the correct left and right eye centres, and d is the distance between the correct eye centres. Our approach yields and accuracy of 90% for eye tracking, 80% for eye blink, 75% for left and right wink and 80% for eyeball movement in left and right directions which will further increase if images with closed eyes and low contrast are left out.

Pose: The image of face can vary due to relative camera-face pose, like; frontal, 45-degree, profile, and upside-down. Also, some features of face (like eyes, mouth ...) may be partially or wholly occluded.

Presence or absence of structural components: Facial features such as beards, moustaches, and glasses may or may not be present and there is a great deal of variability among these components including shape, colour, and size.

Facial expression: The appearances of faces are directly affected by a person's facial expression.

Occlusion: Faces may be partially occluded by other objects. In an image with a group of people, some faces may partially occlude other faces.

Image orientation: Face images directly vary for different rotations about the camera's optical axis.

Imaging conditions: When the image is formed, factors such as lighting (spectra, source distribution and intensity) and camera characteristics (sensor response, lenses) affect the appearance of a face

4.1. Demonstration

When the program is launched, it is neither in scroll nor in mouse mode.(Fig 4.7.)

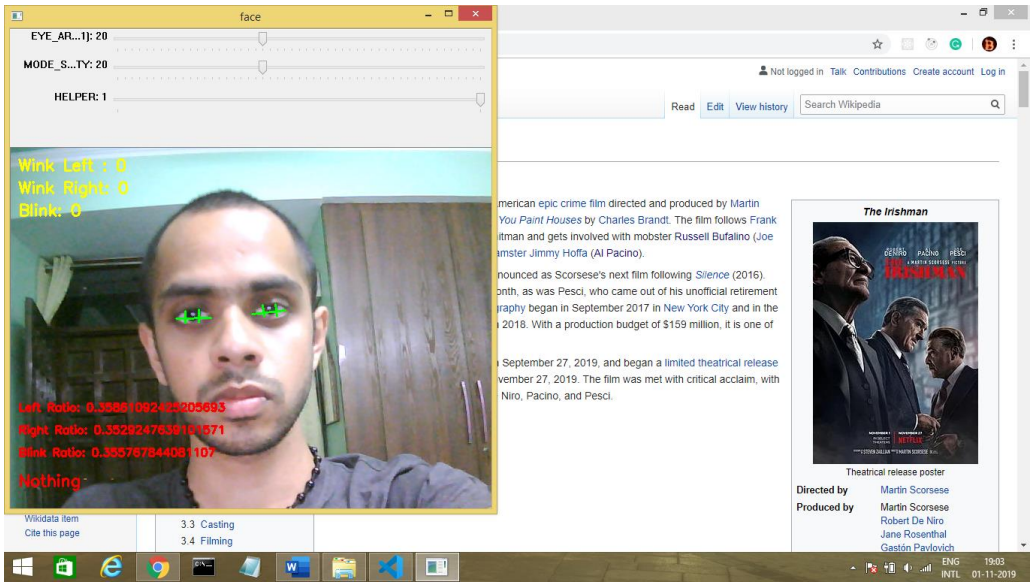

Fig 4.7. Opening screen

To shift the program to mouse mode, close your eyes for 2-3 seconds (Fig 4.8.) and then look right until the pop up window turns red and closes (Fig 4.9.).

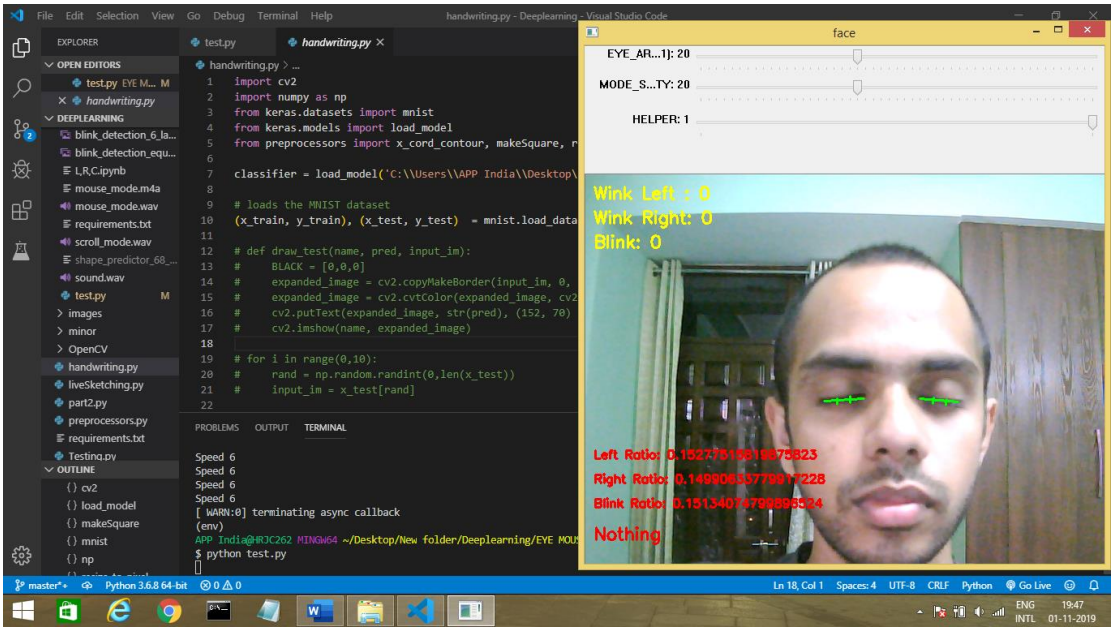

Fig 4.8. Closing eyes to begin tracking

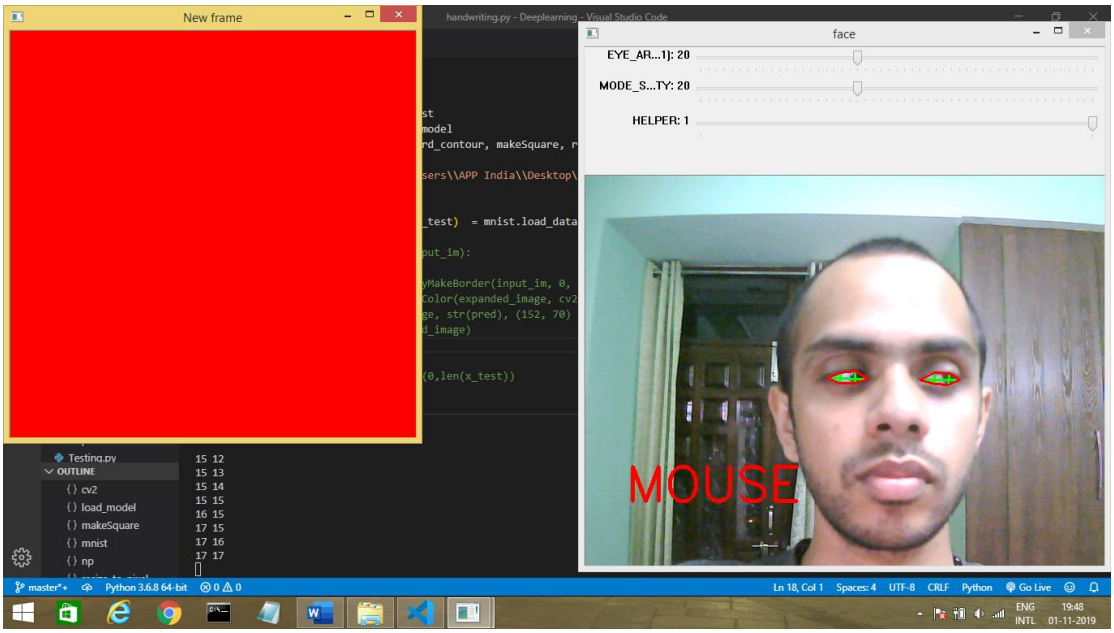

Fig 4.9. selecting mouse mode

To move the mouse, move your eye out of the anchor area box (Fig 4.10.). The mouse will move towards the respective direction your eye moves. The speed of the mouse is controlled by how far your eye from the anchor area box is. If the eye is far then the mouse will move fast, if it is close then the mouse will move slowly. If the eye is within the anchor area box the mouse will not move.

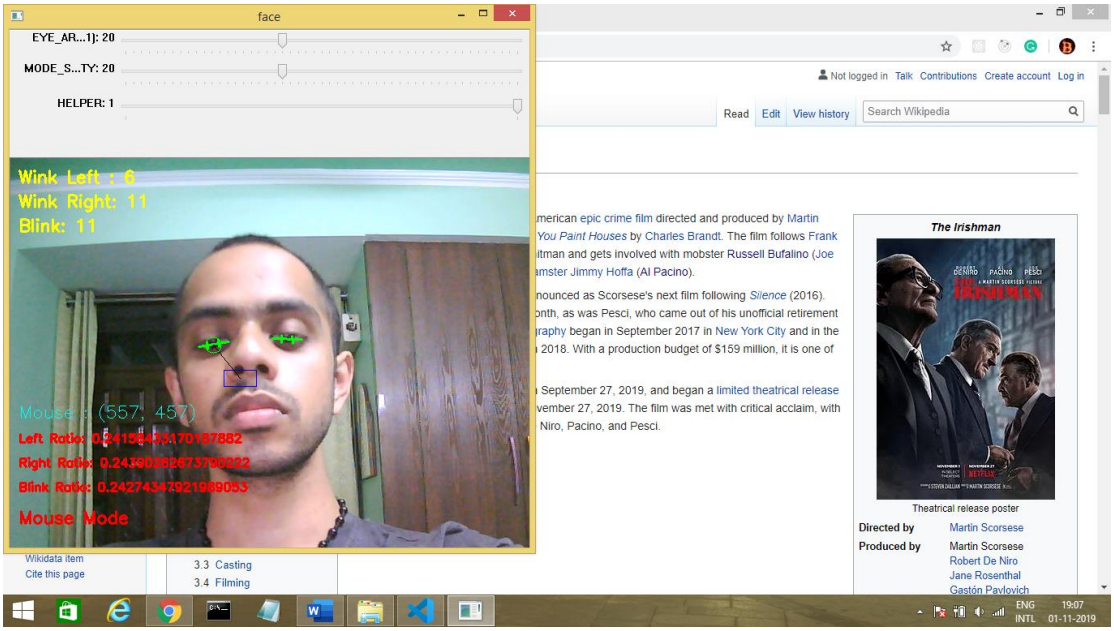

Fig 4.10. Moving the mouse

For click operation, you just must wink your right eye for right click and wink your left eye for left click.(Fig 4.11- 4.12.)

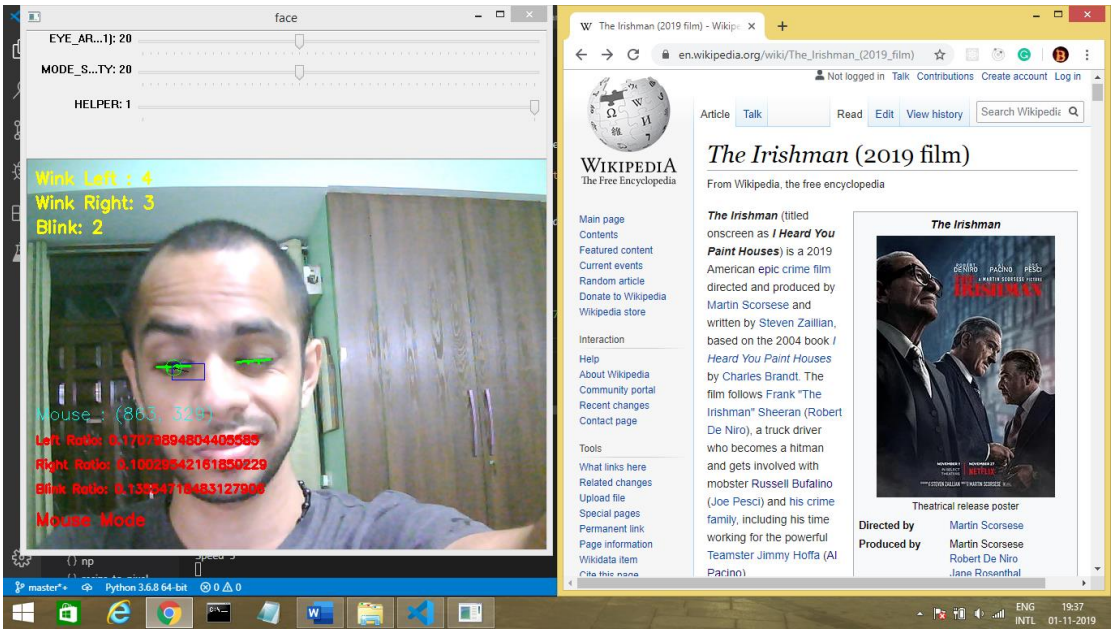

Fig 4.11. Left click

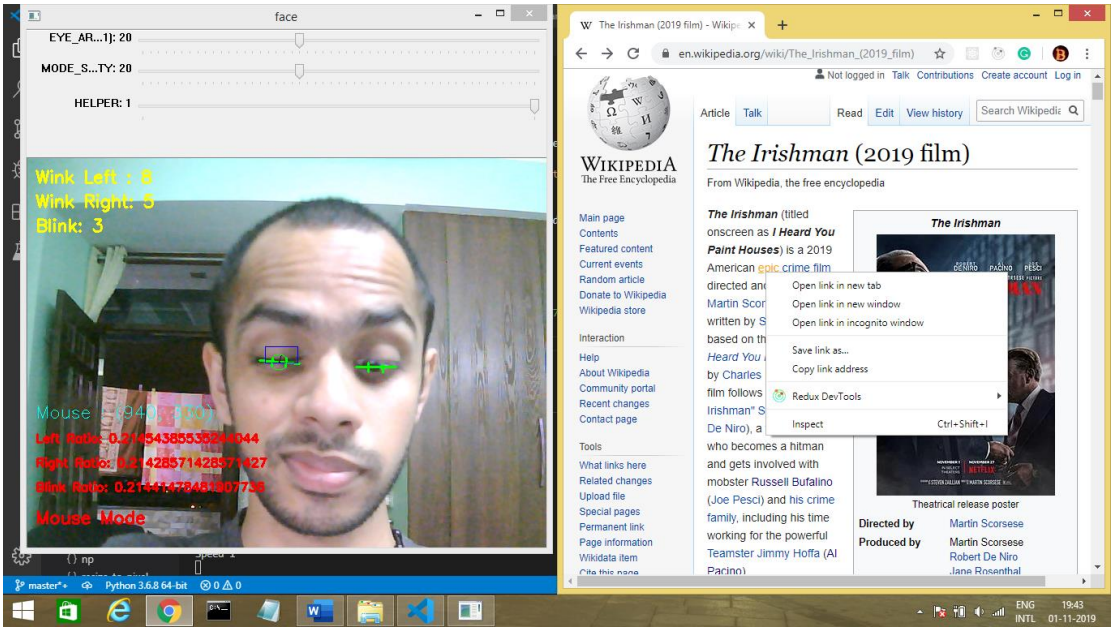

Fig 4.12. Right Click

To shift to scroll mode you must close your eyes for 2-3 seconds and open and look to your left until the pop up window becomes blue and closes.(Fig 4.13.)

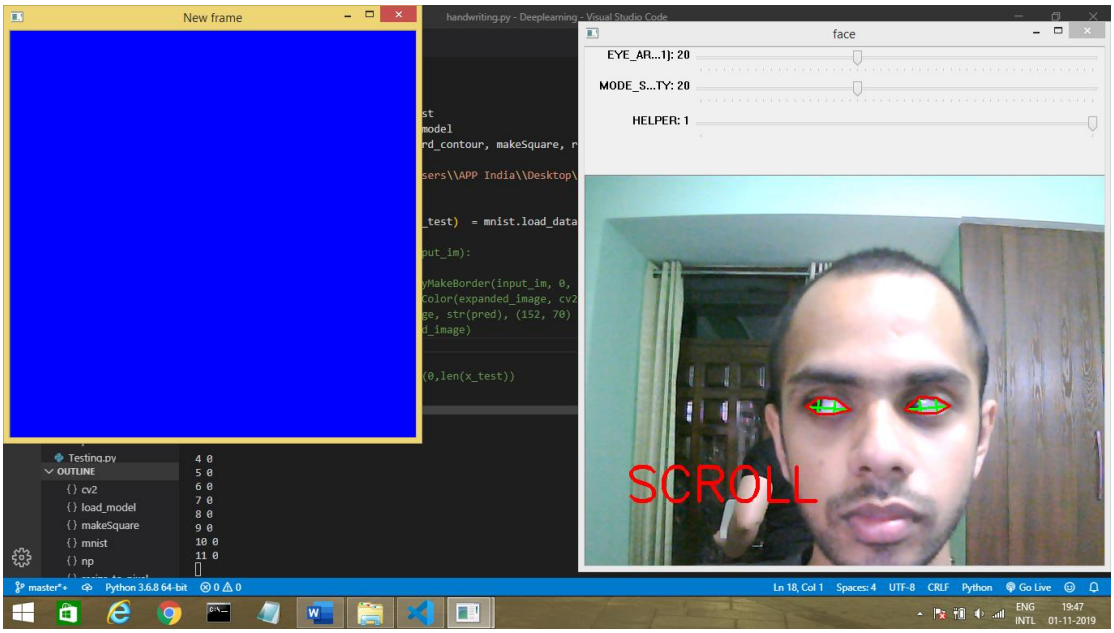

Fig 4.13. Scroll mode selected

To scroll up or down you must look up or down from your current position. The speed of scrolling is constant. (Fig 4.14.)

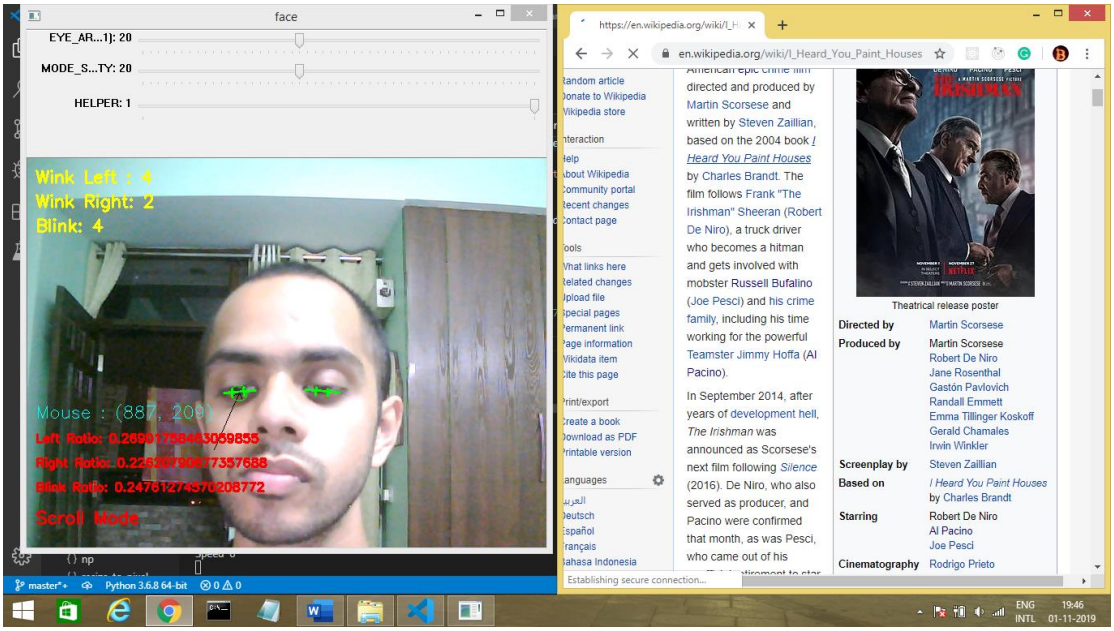

Fig 4.14. Scrolling

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

With this project we were able to develop an Eye Mouse for the motor impaired which can perform the basic functions of a mouse such as click, scroll and movement over the screen based on the user's eye movements. This project will act as a helping hand for those who are paralysed neck down or don't have hands and have to use the mouse while using a computer.

The eye gaze mouse enables these motor impaired patients to control the mouse of a computer by using their gaze. The program uses the head movements of the user to map the movement direction of the mouse and uses left and right winks for the respective clicks. The speed of the mouse also varies as we move away from the centre anchor point. The farther we go the faster the mouse moves. A special scroll mode is also available where the user can scroll easily through a page without the need for clicking the down button on the scrollbar.

With this project patients all around the world can communicate better using the eye gaze mouse. This project also allows them to use web browsers and go through websites, which in todays world is one of the most important application to communicate with the world.

Apart from these, this project will also allow them to use various applications which are not yet upgraded to use voice commands or are yet equipped to help these patients. The users can also use this project to use older applications which have not been updated. The users can also use word processors or chat applications with this project by using a virtual keyboard.

This project involves building camera mouse which works as a mouse replacement system. The following are the applications where this project can be extended:

1. The proposed project is suitable for pc/desktop. These features can be implemented on mobile phones to facilitate hands free access to it.

2. Camera mouse is a way to increase employability of disabled people. This will also help in increasing efficiency.
3. In addition to eye and head movement, we can inculcate speech module.
4. To improve the number of operations performed, we can add facial gesture recognition like smirk, yawn etc.
5. Eye tracking in combination with conventional research methods or other biosensors can even be helpful for diagnosing diseases such as Attention Deficit Hyperactivity Disorder (ADHD), Autism Spectrum Disorder (ASD), Obsessive Compulsive Disorder (OCD), Schizophrenia, Parkinson's and Alzheimer's disease. For instance, it can be used to detect drowsiness or support various other fields for medical, quality assurance or monitoring use.
6. Eye tracking for usability and user experience is an emerging field using these methodologies. One classic example is website testing. Here, attention to real estate, communication, and call to action (CTA) can be measured.
7. Eye tracking has also been introduced to the human-computer interaction and gaming industry which now enables for instance game designers to get a better understanding of the game experience so that it is somewhat possible to control the experience and create features that push the boundaries of reality even more. In the time to come, it will most likely even be possible to personalize the game's development in regard to pupil dilation of the player and the gamer will be able to control the game with eye movements.

REFERENCES

[1] Z. Orman, A. Battal, and E. Kemer. "A study on face, eye detection and gaze estimation." *International journal of computer science & engineering survey (IJCSES)*, vol. 2, no. 3, pp. 29-46, Aug 2011.

[2] V. Raudonis, R. Simutis, and G. Narvydas, "Discrete eye tracking for medical applications," in Proceeding of 2nd International Symposium on Applied Sciences in Biomedical and Communication Technologies, pp. 1–6, 2009.

[3] M. Mehrubeoglu, L. M. Pham, H. T. Le, R. Muddu, and D. Ryu, "Real-time eye tracking using a smart camera," in Proceeding of Applied Imagery Pattern Recognition Workshop, pp. 1–7, 2011.

[4] B. Fu and R. Yang, "Display control based on eye gaze estimation," in Proceeding of 4th International Congress on Image and Signal Processing, vol. 1, pp. 399–403, 2011.

[5] Y. Kuo, J. Lee, and S. Kao, "Eye tracking in visible environment," in Proceeding of 5th International Conference on Intelligent Information Hiding and Multimedia Signal Processing, pp. 114–117, 2009.

[6] A. Kudale, V. Laddha, R. Thakare, and L. Deshpande. "Mouse Navigation Control Using Head Motion." *International Journal of Advanced Research in Computer Science*, vol. 3, no. 3, May 2012.

[7] J. Tu, T. Huang, and H. Tao. "Face as mouse through visual face tracking." *In The 2nd Canadian Conference on Computer and Robot Vision (CRV'05)*, pp. 339-346, May 2005.

[8] M. Betke, J. Gips, and P. Fleming. "The camera mouse: visual tracking of body features to provide computer access for people with severe disabilities." *IEEE Transactions on neural systems and Rehabilitation Engineering*, vol. 10, no. 1, pp. 1-10, Nov 2002.

[9] Y. Fu, and T. S. Huang. "hMouse: Head tracking driven virtual computer mouse." *IEEE Workshop on Applications of Computer Vision (WACV'07)*, pp. 30-30. IEEE, Feb 2007.

[10] T. Siriteerakul, Y. Sato, and V. Boonjing. "Estimating change in head pose from low resolution video using LBP-based tracking." *International Symposium on Intelligent Signal Processing and Communications Systems (ISPACS)*, pp. 1-6, Dec 2011.

[11] Y. Song, Y. Luo, and J. Lin. "Detection of movements of head and mouth to provide computer access for disabled." In *2011 International Conference on Technologies and Applications of Artificial Intelligence*, pp. 223-226, Nov 2011.

[12] L. J. Ball, D. R. Beukelman, and G. L. Pattee. "Communication effectiveness of individuals with amyotrophic lateral sclerosis." *Journal of Communication Disorders*, vol. 37, no. 3 pp. 197-215, May 2004.

[13] Z. Zhao, Y. Wang, and S. Fu. "Head movement recognition based on Lucas-Kanade algorithm." *International Conference on Computer Science and Service System*, pp. 2303-2306. Aug 2012.

[14] Y. Xu, J. Zeng, and Y. Sun. "Head pose recovery using 3D cross model." *4th International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 2, pp. 63-66, Aug 2012.

[15] N. H. Cuong, and H. T. Hoang. "Eye-gaze detection with a single webcam based on geometry features extraction." *11th International Conference on Control Automation Robotics & Vision*, pp. 2507-2512., Dec 2010.

[16] L. E. MacLellan, "Evaluating Camera Mouse as a computer access system for augmentative and alternative communication in cerebral palsy: a case study." PhD diss., COLLEGE OF HEALTH AND REHABILITATION SCIENCES Thesis BS, Boston University, 2018.

[17] A. Królak, and P. Strumiłło. "Eye-blink detection system for human–computer interaction." *Universal Access in the Information Society,* vol. 11, no. 4, pp. 409-419, Nov 2012.

[18] A. Erdem, E. Erdem, Y. Yardimci, V. Atalay, and A. E. Cetin. "Computer vision based mouse." *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, pp. IV-4178, May 2002.

[19] P. Robertson, R. Laddaga, and M. V. Kleek. "Virtual mouse vision based interface." In Proceeding of 9th international conference on Intelligent user interfaces, pp. 177-183, 2004.

[20] R. G. Lupu, F. Ungureanu, and V. Siriteanu. "Eye tracking mouse for human computer interaction." *E-Health and Bioengineering Conference (EHB)*, pp. 1-4., Nov 2013.

[21] C. Rotariu, H. Costin, V. Cehan and O. Morancea, "A communication system with severe neuro-locomotor handicapped persons" In Proceeding of International Conference Biomedical Electronics and Biomedical Informatics, pp. 145-149, 2008.

[22] J. S. Agustin, E. Mollenbach, M. Barret, M. Tall and D. W. Hansen, "Evaluation of a low-cost open-source gaze tracker", In Proceeding of Eye-Tracking Research & Applications Symposium ‘10, pp. 77-80, 2010.

[23] R. G. Lupu, F. Ungureanu and R. G. Bozomitu, "Mobile embedded system for human computer communication in assitive technology", In Proceeding of IEEE International Conference Computational Photography ‘12, pp. 209-212, 2012.

[24] R. G. Bozomitu, C. Barabaşa, V. Cehan, and R. G. Lupu. "The hardware component of the technology used to communicate with people with major neuro-locomotor disability using ocular electromyogram." *IEEE 17th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, pp. 193-196. IEEE, Oct 2011.

[25] J. Hori, K. Sakano, and Y. Saitoh. "Development of communication supporting device controlled by eye movements and voluntary eye blink." In *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, vol. 2, pp. 4302-4305. IEEE, Sep 2004.

[26] M. Betke, J. Gips, and P. Fleming. "The camera mouse: visual tracking of body features to provide computer access for people with severe disabilities." *IEEE Transactions on neural systems and Rehabilitation Engineering* vol. 10, no. 1 pp. 1-10, Nov 2002.

[27] R. G. Lupu, R. G. Bozomitu, V. Cehan, and D. A. Cehan. "A new computer-based technology for communicating with people with major neuro-locomotor disability using ocular electromyogram." In Proceeding of 34th International Spring Seminar on Electronics Technology (ISSE), pp. 442-446, 2011.

[28] R. G. Lupu, R. G. Bozomitu, F. Ungureanu, and V. Cehan. "Eye tracking based communication system for patient with major neoro-locomotor disabilites." In *15th International Conference on System Theory, Control and Computing*, pp. 1-5., Oct 2011.

[29] P. Viola, and M. Jones. "Rapid object detection using a boosted cascade of simple features." *Computer Vision and Pattern Recognition (1)* vol. 1, no. 511-518 p. 3, Dec 2001.

[30] T. Soukupova, and J. Cech. "Eye blink detection using facial landmarks." In *21st Computer Vision Winter Workshop, Rimske Toplice, Slovenia*. 2016.

Appendix-A

Code

```
import cv2
import numpy as np
import dlib
import math
from scipy.spatial import distance as dist
import pyglet
from pynput.mouse import Button, Controller
```

Libraries used

```
mouse = Controller()

cap = cv2.VideoCapture(0)




eye_pos_i=1;
eye_pos=["Mouse Mode","Nothing","Scroll Mode"]
mode_detect=False
text="Nothing"

EYE_AR_THRESH = 0.25
EYE_AR_CONSEC_FRAMES = 3
MODE_SELECTION_SENSITIVITY=5

COUNTER_LEFT = 0
TOTAL_LEFT = 0

COUNTER_RIGHT = 0
TOTAL_RIGHT = 0

COUNTER_BLINK = 0
TOTAL_BLINK = 0

EYEBALL_LEFT_COUNTER=0
EYEBALL_RIGHT_COUNTER=0

anchor_pointx=60
anchor_pointy=60
```

Base Values

```
#-------FRAME------------
cv2.namedWindow('face')
cv2.createTrackbar('EYE_AR_THRESH (100^-1)','face',20,50,nothing)
cv2.createTrackbar('MODE_SELECTION_SENSITIVITY','face',20,50,nothing)
cv2.createTrackbar('HELPER','face',1,1,nothing)

#------SOUNDS------------
sound = pyglet.media.load("sound.wav", streaming=False)
mouse_mode_sound = pyglet.media.load("mouse_mode.wav", streaming=False)
scroll_mode_sound = pyglet.media.load("scroll_mode.wav", streaming=False)



detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("./shape_predictor_68_face_landmarks.dat")
```

Setting up window, detector and predictor

```
def get_blinking_ratio(eye_points, facial_landmarks):
    p1 = (facial_landmarks.part(eye_points[0]).x, facial_landmarks.part(eye_points[0]).y)
    p2 = (facial_landmarks.part(eye_points[1]).x, facial_landmarks.part(eye_points[1]).y)
    p3 = (facial_landmarks.part(eye_points[2]).x, facial_landmarks.part(eye_points[2]).y)
    p4 = (facial_landmarks.part(eye_points[3]).x, facial_landmarks.part(eye_points[3]).y)
    p5 = (facial_landmarks.part(eye_points[4]).x, facial_landmarks.part(eye_points[4]).y)
    p6 = (facial_landmarks.part(eye_points[5]).x, facial_landmarks.part(eye_points[5]).y)

    line_14 = cv2.line(frame, p1, p4, (0, 255, 0), 2)
    line_26 = cv2.line(frame, p2, p6, (0, 255, 0), 2)
    line_35 = cv2.line(frame, p3, p5, (0, 255, 0), 2)

    line_14_length = dist.euclidean(p1,p4)
    line_26_length = dist.euclidean(p2,p6)
    line_35_length = dist.euclidean(p3,p5)

    eye_aspect_ratio = (line_26_length+line_35_length)/(2*line_14_length)
    return eye_aspect_ratio
```

Calculating blinking ratio

```python
def get_gaze_ratio(eye_points, facial_landmarks):
    left_eye_region = np.array([(facial_landmarks.part(eye_points[0]).x, facial_landmarks.part(eye_points[0]).y),
                                (facial_landmarks.part(eye_points[1]).x, facial_landmarks.part(eye_points[1]).y),
                                (facial_landmarks.part(eye_points[2]).x, facial_landmarks.part(eye_points[2]).y),
                                (facial_landmarks.part(eye_points[3]).x, facial_landmarks.part(eye_points[3]).y),
                                (facial_landmarks.part(eye_points[4]).x, facial_landmarks.part(eye_points[4]).y),
                                (facial_landmarks.part(eye_points[5]).x, facial_landmarks.part(eye_points[5]).y)],
                                np.int32)
    cv2.polylines(frame, [left_eye_region], True, (0, 0, 255), 2)

    height, width, _ = frame.shape
    mask = np.zeros((height, width), np.uint8)
    cv2.polylines(mask, [left_eye_region], True, 255, 2)
    cv2.fillPoly(mask, [left_eye_region], 255)
    eye = cv2.bitwise_and(gray, gray, mask=mask)

    min_x = np.min(left_eye_region[:, 0])
    max_x = np.max(left_eye_region[:, 0])
    min_y = np.min(left_eye_region[:, 1])
    max_y = np.max(left_eye_region[:, 1])

    gray_eye = eye[min_y: max_y, min_x: max_x]
    _, threshold_eye = cv2.threshold(gray_eye, 70, 255, cv2.THRESH_BINARY)
    height, width = threshold_eye.shape
    left_side_threshold = threshold_eye[0: height, 0: int(width / 2)]
    left_side_white = cv2.countNonZero(left_side_threshold)

    right_side_threshold = threshold_eye[0: height, int(width / 2): width]
    right_side_white = cv2.countNonZero(right_side_threshold)
```

Eye gaze ratio calculation

```python
while True:
    _, frame = cap.read()

    new_frame = np.zeros((500, 500, 3), np.uint8)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = detector(gray)

    EYE_AR_THRESH=cv2.getTrackbarPos('EYE_AR_THRESH (100^-1)','face')/100
    for face in faces:
        #x, y = face.left(), face.top()
        #x1, y1 = face.right(), face.bottom()
        #cv2.rectangle(frame, (x, y), (x1, y1), (0, 255, 0), 2)

        landmarks = predictor(gray, face)

        # CALCULATE RATIO
        left_eye_ratio = get_blinking_ratio([36, 37, 38, 39, 40, 41], landmarks)
        right_eye_ratio = get_blinking_ratio([42, 43, 44, 45, 46, 47], landmarks)
        blinking_ratio = (left_eye_ratio + right_eye_ratio) / 2
```

Clicking image and calculating ratios using landmarks

```
                    #DETECT BLINK,LEFT WINK,RIGHT EINK
#--------------------------------------------------------------------------------
if left_eye_ratio<EYE_AR_THRESH and right_eye_ratio<EYE_AR_THRESH:
    COUNTER_BLINK += 1
# otherwise, the eye aspect ratio is not below the blink
# threshold
else:
    # if the eyes were closed for a sufficient number of
    # then increment the total number of blinks
    if COUNTER_BLINK >= EYE_AR_CONSEC_FRAMES:
        TOTAL_BLINK += 1
        #sound.play()
        mode_detect=not mode_detect

    # reset the eye frame counter
    COUNTER_BLINK = 0
if left_eye_ratio < EYE_AR_THRESH and right_eye_ratio>EYE_AR_THRESH:
    COUNTER_LEFT += 1
else:
     if COUNTER_LEFT >=2:
        TOTAL_LEFT += 1
        print("Left eye winked")
        COUNTER_LEFT = 0
        mouse.click(Button.right,1)
if right_eye_ratio < EYE_AR_THRESH and left_eye_ratio>EYE_AR_THRESH:
     COUNTER_RIGHT += 1
else:
     if COUNTER_RIGHT >= 2:
        TOTAL_RIGHT += 1
        print("Right eye winked")
        COUNTER_RIGHT = 0
        mouse.click(Button.left,1)
```

Detecting blink, left wink and right wink

```
#--------------------------------------------------------------------------------
    x1=int((landmarks.part(36).x+landmarks.part(39).x)/2)
    y1=int((landmarks.part(37).y+landmarks.part(41).y)/2)
    leftup=(int(anchor_pointx)-20,int(anchor_pointy)-10)
    rightbottom=(int(anchor_pointx)+20,int(anchor_pointy)+10)
    cv2.rectangle(frame,leftup,rightbottom,(255,0,0))


    vector_length=dist.euclidean((x1,y1),(anchor_pointx,anchor_pointy))

    x_mag=y_mag=math.floor(vector_length/6)

    print('Speed',x_mag)
    speedx=0;
    speedy=0;

    if(x1<leftup[0]):
        speedx=-x_mag
        #cv2.putText(frame, "Left", (x1-14, y1), font, 0.7, (0, 0, 255), 2)
    elif(x1>leftup[0] and x1 <rightbottom[0]):
        speedx=0;
    else:
        speedx=x_mag;
        #cv2.putText(frame, "Right", (x1-14, y1), font, 0.7, (0, 0, 255), 3)


    if(y1<leftup[1]):
        speedy=-y_mag
        #cv2.putText(frame, "Up", (x1+14, y1), font, 0.7, (0, 0, 255), 3)
    elif(y1>leftup[1] and y1 <rightbottom[1]):
        speedy=0;
    else:
        speedy=y_mag;
```

Mouse mode

```
x1=int((landmarks.part(36).x+landmarks.part(39).x)/2)
y1=int((landmarks.part(37).y+landmarks.part(41).y)/2)
leftup=(int(anchor_pointx)-20,int(anchor_pointy)-10)
rightbottom=(int(anchor_pointx)+20,int(anchor_pointy)+10)

x_mag=y_mag=6

print('Speed',x_mag)
speedx=0;
speedy=0;

if(x1<leftup[0]):
    speedx=-x_mag
    #cv2.putText(frame, "Left", (x1-14, y1), font, 0.7, (0, 0, 255), 2)
elif(x1>leftup[0] and x1 <rightbottom[0]):
    speedx=0;
else:
    speedx=x_mag;
    #cv2.putText(frame, "Right", (x1-14, y1), font, 0.7, (0, 0, 255), 3)


if(y1<leftup[1]):
    speedy=-y_mag
    #cv2.putText(frame, "Up", (x1+14, y1), font, 0.7, (0, 0, 255), 3)
elif(y1>leftup[1] and y1 <rightbottom[1]):
    speedy=0;
else:
    speedy=y_mag;
    #cv2.putText(frame, "Down", (x1+14, y1), font, 0.7, (0, 0, 255), 3)

mouse.scroll(speedx,speedy)
```

Scroll mode

```
else:
    gaze_ratio_left_eye = get_gaze_ratio([36, 37, 38, 39, 40, 41], landmarks)
    gaze_ratio_right_eye = get_gaze_ratio([42, 43, 44, 45, 46, 47], landmarks)
    gaze_ratio = (gaze_ratio_right_eye + gaze_ratio_left_eye) / 2
    anchor_pointx=int((landmarks.part(36).x+landmarks.part(39).x)/2)
    anchor_pointy=int((landmarks.part(37).y+landmarks.part(41).y)/2)

    MODE_SELECTION_SENSITIVITY=cv2.getTrackbarPos('MODE_SELECTION_SENSITIVITY','face')
    print(EYEBALL_LEFT_COUNTER,EYEBALL_RIGHT_COUNTER)

    if gaze_ratio > 1:
        cv2.putText(frame, "MOUSE", (50, 400), font, 2, (0, 0, 255), 3)
        EYEBALL_RIGHT_COUNTER+=1
        new_frame[:] = (0, 0, 255)
        if EYEBALL_RIGHT_COUNTER>=MODE_SELECTION_SENSITIVITY:
            EYEBALL_LEFT_COUNTER=0
            EYEBALL_RIGHT_COUNTER=0
            eye_pos_i=0
            #mouse_mode_sound.play()
            print("hi")
            mode_detect=not mode_detect

    else:
        new_frame[:] = (255, 0, 0)
        EYEBALL_LEFT_COUNTER+=1
        cv2.putText(frame, "SCROLL", (50, 400), font, 2, (0, 0, 255), 3)
        if EYEBALL_LEFT_COUNTER>=MODE_SELECTION_SENSITIVITY:
            EYEBALL_LEFT_COUNTER=0
            EYEBALL_RIGHT_COUNTER=0
            eye_pos_i=2
            #scroll_mode_sound.play()
            mode_detect=not mode_detect
```

Gaze detection