OCULAR INPUT FOR MONITOR MANIPULATION FOR HAND MOBILITY

A report submitted in partial fulfillment of the requirements for the Degree of

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

BY

B. Sai Sankeerth (2011CS010049)

Under the esteemed guidance of

Mrs. K. MANASA
ASSISTANT PROFESSOR

Department of Computer Science and Engineering

School of Engineering

MALLA REDDY UNIVERSITY

Maisammaguda, Dulapally, Hyderabad, Telangana - 500100

2024

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that this is the bonafide record of the project entitled **"Ocular Input For Monitor Manipulation"**, submitted by **B. Sai Sankeerth (2011CS010049),** towards the partial fulfillment for the award of Bachelor's Degree in Computer Science and Engineering, Department of CSE, Malla Reddy University, Hyderabad, during the year 2023-2024.. The results embodied in the work are not submitted to any other University or Institute for award of any degree or diploma.

Internal Guide
Mrs. K. Manasa
Assistant Professor

Head of the Department
Dr.Meeravali Shaik
M.Tech., Ph.D.

DECLARATION

We hereby declare that the project report entitled **"Ocular Input For Monitor Manipulation For Hand Mobility"** has been carried out by us and this work has been submitted to the Department of Computer Science and Engineering, Malla Reddy University, Hyderabad in partial fulfillment of the requirements for the award of degree of Bachelor of Technology. We further declare that this project work has not been submitted in full or part for the award of any other degree in any other educational institutions.

Place: Hyderabad
Date: 27-04-24

B.Sai Sankeerth **2011CS010049** **(B.Sankeerth)**

ACKNOWLEDGEMENT

We extend our sincere gratitude to all those who have contributed to the completion of this project report. Firstly, We would like to extend our gratitude to Dr. V. S. K Reddy, Vice-Chancellor, for his visionary leadership and unwavering commitment to academic excellence.

We would also like to express my deepest appreciation to our project guide Mrs.K.Manasa, Assistant Professor, whose invaluable guidance, insightful feedback, and unwavering support have been instrumental throughout the course of this project for successful outcomes.

We are also grateful to Dr.Meeravali Shaik, Head of the Department of CSE, for providing us with the necessary resources and facilities to carry out this project. We would like to thank Dr. Kasa Ravindra, Dean, School of Engineering, for his encouragement and support throughout my academic pursuit.

My heartfelt thanks also go to Dr. Harikrishna Kamatham, Associate Dean School of Engineering for his guidance and encouragement.

We are deeply indebted to all of them for their support, encouragement, and guidance, without which this project would not have been possible.

B.Sai Sankeerth 2011CS010049

ABSTRACT

Controlling the monitor by a physically challenged person is really tough one. To find a solution for such challenged individuals, we have proposed this monitor controlling mechanism using Ocular input. Facial structure and it's actions is an alternative way of accessing a computer with the help of Machine Learning. For someone who find touchscreens, mouse, monitor inaccessible, facial structure is an alternative method to allow a user to operate their computer, using the movement of their eyes, mouth and nose. In order to improve the reliability, mobility, and usability of facial tracking technique in user-computer dialogue, a novel facial feature control system is proposed in this system using Webcam and without using any extra hardware making it cost effective and efficient to use. The proposed system focuses on providing a simple and convenient interactive mode by only using user's face. The usage flow of the proposed system is designed to perfectly follow human natural habits. The proposed system describes the implementation of both facial features and manipulation of monitor according to Lineaments position which can be used to control the monitor using Python.

This project proposes a novel approach to enhance computer interaction for individuals with limited hand mobility by integrating ocular input with traditional hand-based manipulation of computer monitors. The aim is to develop a system that allows users to control monitor functions and interact with on-screen content through a combination of eye movements and gestures, thereby providing an alternative method for those with impaired hand dexterity. The system utilizes eye-tracking technology alongside hand motion sensors to detect and interpret user commands. Through a user-centered design approach, the system will be tailored to accommodate varying levels of hand mobility and visual capabilities. The implementation of this technology has the potential to significantly improve accessibility to digital interfaces for individuals with disabilities, fostering greater independence and inclusivity in computer usage. The project's success will be evaluated through usability testing and user feedback, with the ultimate goal of providing a seamless and intuitive interface for individuals with diverse motor abilities.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF OUTPUTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

In an increasingly digital world, the ability to interact with computer monitors and navigate digital interfaces is fundamental for participation in various aspects of modern life, from education and employment to social engagement and entertainment. However, for individuals with limited hand mobility due to conditions such as muscular dystrophy, spinal cord injuries, or arthritis, traditional methods of computer interaction pose significant challenges.

Recognizing the importance of addressing these challenges, the project "Ocular Input for Monitor Manipulation for Hand Mobility" emerges as a pioneering endeavor to redefine how individuals with impaired hand dexterity engage with technology. By seamlessly integrating ocular input with conventional hand-based manipulation of computer monitors, this project aims to bridge the accessibility gap and empower users to interact with digital interfaces more effectively.

The significance of this project lies not only in its technological innovation but also in its potential to transform the lives of individuals with disabilities. By providing an alternative method of interaction that leverages eye-tracking technology and motion sensors, the system offers newfound independence and agency to users who may have previously struggled to navigate digital environments.

Moreover, the project adopts a user-centered design approach, ensuring that the developed system is not only functional but also intuitive and adaptable to the diverse needs and preferences of its users. Through iterative prototyping and rigorous usability testing, the interface will be refined to prioritize ease of use and accessibility, ultimately enhancing the overall user experience.

As technology continues to evolve, it is essential to consider the principles of inclusivity and accessibility in the design and development of digital interfaces. By pioneering the integration of ocular input with monitor manipulation, this project represents a significant step forward in advancing accessibility in computer interaction, paving the way for a more inclusive digital

future.

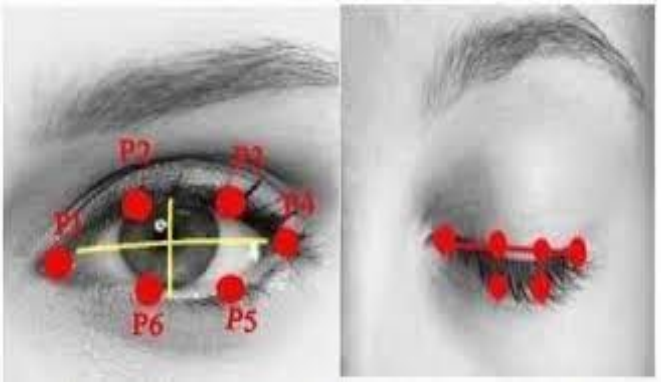

Fig.1.1.1 The 6 facial points linked to the eye

At present situation paralyzed peoples need a guidance to do any work. One person should be they're with that person to taken care of him. By using the eye ball tracking mechanism, we can fix the centroid on the eye based on the centroid we need to track that paralyzed person's eye this eye ball track mechanism involves many applications like home automation by using python GUI robotic Control and virtual keyboard application.

Our existing laptop and computer user interface cannot be used by all people, in other words the differently abled , they often face difficulty while using these devices. They always need to seek out help from others which makes them dependent on someone. To connect them with the modern world and so as to make them feel independent, the following project will give a boost in their day to day life. Also this project is useful for all types of people who would use their systems efficiently and with greater speed. This project is not only for the sole purpose of the differently abled but also for those who spend hours using their pc and the mouse , it is observed that excessive use of the mouse can put their wrist in danger of getting some serious inflammation which could lead to some major issues of internal tissue damage .Through this project it can be said that how an electronic device can be used to simplify things without any excessive usage of hands , by simply using the vision one can control the mouse without any hindrance.

Nowadays, computer systems have become a vast part in our day to day survival since they are used in various areas such as at workplace etc.Personal Computers mostly use input method as mouse. But in case of disability or physically handicapped they cannot able to operate computers. In such cases, it might be preferable to use input methods such as eye movements. To enable such input method , a system is designed to control cursor on a computer system without the use of mouse . In the new system, the cursor movement is controlled by the eyeball movement using OpenCV. It has camera which detects the Eyeball movements and based on these eyeball movements the cursor is controlled using OPENCV.

In today's world computers are widely used , but still there is an hindrance for the physically disabled people.They cannot use the computer independently ,they need help from someone.To ensure that even people with physical disability can use the computers,and that too independently , the proposed project will ensure that they can operate the computer with much ease . Also this project is useful for all types of people who would use their systems efficiently and with greater speed. This project is not only useful for the physically disabled people ,but anyone can use it , because it has been observed that people after using computer for long hours are facing discomfort in their wrists .Through this project it can be said that an electronic device ie mouse is replaced with simple the eye movements of the user .The paper reduces the use of hardware and promotes to use only software and webcam .The system performs by using a webcam that will capture live images of the user .The Haar cascade algorithm is applied to each frame to get all the faces in the image. In the next step a facial scanner is applied to each detected face to detect facial features such as eyes, eyebrows, nose, mouth, etc.A particular area of interest, the eyes in this situation, are considered and some image processing techniques are used to work better in eye tracking. The cursor can be controlled by certain functions in the Python library.A delayed blink of an eye performs a click-through action on the device. The users face and eyes are captured on a photo using Facial Landmark Detection. By using eye-to-eye eye recognition features are obtained and by using those links we are able to track eye movements.Then by mapping those links with a computer cursor link,the cursor control can be accessed.

1.2 PROBLEM STATEMENT

Individuals with limited hand mobility face barriers in effectively interacting with digital interfaces due to reliance on traditional manual input devices like keyboards and mice. Existing assistive technologies have limitations in accuracy and compatibility, hindering accessibility for users with varying needs. There is a pressing need for a comprehensive and intuitive solution that integrates innovative technologies like ocular input and motion sensors to enhance accessibility and inclusivity in computer interaction for individuals with impaired hand dexterity.

1.3 SCOPE

The scope of the project includes the design, development, and implementation of a system that integrates ocular input with traditional hand-based manipulation of computer monitors. The system will be tailored to accommodate individuals with limited hand mobility, providing alternative methods for interacting with on-screen content. Key components of the project include the integration of eye-tracking technology, motion sensors, and user interface design to create a seamless and intuitive interface. Usability testing and refinement will ensure the system meets the diverse needs and preferences of users with varying levels of hand mobility and visual capabilities.

1.4 OBJECTIVE

This project's primary objective is to empower individuals with disabilities, particularly those with limited hand mobility, to effectively interact with computers by controlling the cursor through eye movements.Utilizing OpenCV-based algorithms, the project aims to achieve precise and stable pupil position detection within the eye, facilitating accurate cursor control.Through eye-based cursor control, the project strives to enhance the independence, inclusion, and equitable participation of individuals with impairments in the digital world.Ultimately, the project seeks to provide a user-friendly and accessible means of computer interaction, specifically designed to cater to individuals with disabilities who cannot use traditional input devices.

1.5 IMPORTANT TERMS AND CONCEPTS

Python OpenCV

OpenCV, short for Open Source Computer Vision Library, is an open-source computer vision and machine learning software library. It's designed to provide a common infrastructure for computer vision applications and to accelerate the development of various vision-related tasks

1.5.1 How OpenCV works

OpenCV (Open Source Computer Vision Library) works by providing a set of functions and algorithms for various computer vision tasks. Here's a high-level overview of how OpenCV works:

- Image Input: OpenCV allows users to read images and videos from files, cameras, or other sources. It supports various image formats such as JPEG, PNG, and BMP, as well as video formats like AVI and MP4.
- Image Processing: Once an image is loaded, OpenCV provides a wide range of functions for image processing. These include operations such as resizing, cropping, rotating, and transforming images. OpenCV also offers numerous filters for tasks like blurring, sharpening, and edge detection.
- Feature Detection and Description: OpenCV includes algorithms for detecting and describing features in images, such as corners, keypoints, and descriptors. These features are essential for tasks like object detection, tracking, and matching.
- Object Detection and Recognition: OpenCV provides algorithms for detecting and recognizing objects in images and videos. This includes techniques like Haar cascades for face detection, HOG (Histogram of Oriented Gradients) for pedestrian detection, and deep learning-based methods for object recognition.
- Video Analysis: OpenCV allows users to process and analyze videos frame by frame. This includes tasks such as motion detection, optical flow estimation, and background subtraction.
- Machine Learning Integration: OpenCV provides tools for machine learning tasks, including classification, clustering, regression, and dimensionality reduction. It also includes interfaces for popular machine learning libraries like TensorFlow and PyTorch, allowing users to leverage pre-trained deep learning models for various vision tasks.

Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.
Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something

about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

CHAPTER 2

LITERATURE SURVEY

2.1 FEASIBILITY STUDY

A project's importance impacts how successfully it can be completed. A feasibility study is carried out to see if the chosen solution is practical and meets the project criteria. The feasibility study considers data such resources, software cost estimates, business advantages, and software maintenance expenses.

2.1.1 Organizational Feasibility

The effectiveness of the proposed information system in meeting the organization's purpose and strategic information system strategy is one of the most critical aspects in organizational feasibility. It could include information on the founders, their education, and the abilities they need to establish and maintain a company.

2.1.2 Economical Feasibility

Economic feasibility is a type of cost-benefit analysis used to determine whether or not a project can be carried out. This phrase refers to the examination and study of a project's ability to assist decision-making by objectively and logically identifying its strengths, flaws, opportunities, and dangers, as well as the resources needed to complete it and an estimate of its chances of success.

2.1.3 Technical Feasibility

Technical feasibility evaluates the expert system's technical complexity and typically includes determining whether the expert system can be employed with cutting-edge procedures and equipment. For expert systems, determining the shell in which the system will be built is critical since it influences technical feasibility.

2.1.4 Behavioural Feasibility

Change has been facilitated by computers in the past, but humans are intrinsically resistant to change. An estimate of the possibility of user personnel reacting unfavorably to the installation of an automated system should be made. Employee job status, relocations, retraining, and turnover have all been known to be influenced by IT installations.

2.2 LITERATURE SURVEY

[1] Proposed by G.Norris and E.Wilson focuses on eye movement with Electroencephalogram(EEG) which is set up consisting of an instrumentation amplifier and an inverting op amp and the system is set up by wearing it on your head and attaching the EEG specifically to the required points on the head . The EYE Mouse detects the change in EOG from looking up,right ,down, left since there is a variation of potential and this is captured accordingly w.r.t the eye movement and it is recorded.This in turn is able to click .

[2] The system proposed by Bullying, J. A. Ward contains the eye mouse,and how the EEG system is used with the cursor movement by the simulation of the brain signal and then mapped accordingly to the cursor.

[3] The work proposed by V. Khare, Et.al focuses on tracking the eye movement by capturing the real time video using a microprocessor .The experiment include the use of a webcam which tracks the live video which in turn is broken into frames and a certain threshold is predefined for some certain movement of eyes ,with the help of comparison of the predefined threshold, thecursor will move accordingly with the movement of the eyes.

[4] In a paper by Mohamed Nasor Et.al focuses on how the Iris detection system is functioned using MATLAB. With the help of a webcam , the face is detected first and then iris is detected and extracted using the library of MATLAB which leads in tracking
of the eyes and then iris shift is calculated ,the shift is then mapped with the help of Graphical user interface and the eye is detected and it is mapped with the cursor and the mouse cursor moves accordingly .

[5] S. Mathew et.al in their proposed paper provides an idea to control home appliances for disabled people . The method here uses an eye tracking method for eye movement for individuals, further following a simple circuitry. In this system , HOG is used to find the Histogram of the image and with the addition of SVM, detection of face takes place and the iris portion is cropped and then there are some points in eye which are targeted and the movements of those points maps the cursor movements ,this is a non training based algorithm and it uses Image processing for all the functions .

[6] In a paper proposed by S. R. Fahim, et alit focus on the uses HOG system and motion vector with python programming and Haar Cascade Algorithm which is a training based algorithm, and it is used mainly with programming in machine learning , eye dataset is given in this, by having multiple dataset of eyes, then the eye data is collected and the following system works accordingly to the eye movement and clicking is done with the help of the eye blinking.

[7] This section describes about our study of the existing systems which we have used as a reference for proposing our system. The system used consists of the face captured by the usage of Matlab vfm tool. The eye detection is done by dividing face into three
equal regions and localizing the upper one-third part. The eye movements are tracked using the corners of the iris as reference andcalculating the iris shift which helps in tracking of eye movement.

[8] This paper contains an experimental procedure of webcambased eye-tracker especially for low powered devices. It consists of five processes which includes background suppression, haarcascade feature based face detection algorithm, geometrical determination of eye position, tracking of eye ball center using mean of gradient vector and detection of the user's gaze. A novel eye movement analysis model is proposed.

[9] This paper based on fiveeye feature points. In this paper Convolutional Neural Network is trained for eye feature point detection. Dataset was constructed containing almost 0.5 million frames from 38 subjects. This system [4] shows use of a commercial eye tracker as a peripheral device implementing it as pointer which only needs an eye movement of the user. The tests were conducted to check the discomfort and willingness of the subjects.

[10] The system implemented in this is used to perform gaze direction estimation from human eye movement using a consumer level depth sensor. The pre- processing of the images is

done to remove the unnecessary information such as background objects. The proposed method obtains the position of the user's head in depth images.

[11] The system uses combination of head pose and eye location information to obtain enhanced gaze estimation. The transformation matrix which is obtained from the head pose is used to normalize the eye regions.

[12] The system in is a computer interface that provides the functionality of an input device like mouse, based on eye actions such as eye blink, eye gaze and gaze control. The implemented system comprises of face detection using APIs, Forehead detection using Canny's Edge detection, eye detection using Cross shaped model, morphological erosion which subdues speckle and jitter noise, smoothening of protrusions present around eye region, morphological dilation, eyedetection, Gaussian filtering , iris detection and finally equalization.

[13] The system used in consists of electro-oculography approach for analyzing the eye movements. It also uses Hough transform technique and canny edge detection. The illumination of the eyes is done using IR LED which is mounted on either sides of the camera lenses which is invisible to the naked eye and hence will not cause any strain to the user's eyes.

[]14] Eye Typing Using Cursor Movement and Blink Detection: A Comparative Study" by Akila, S., and Jaisankar, N. (2018).This paper compares different techniques for eye typing using cursor movement and blink detection. It evaluates the effectiveness and efficiency of these techniques in enabling users to input text and perform actions using only their eye movements.

[15] "Eye Gesture Based Virtual Keyboard and Mouse for Hands-Free Typing and Control" by Lin, T. Y., Lee, K. C., and Huang, K. C. (2016).The authors propose a virtual keyboard and mouse system that enables hands-free typing and control using eye gestures. The system tracks eye movements to move the cursor and select keys on the virtual keyboard, allowing users to input text and perform actions without physical input devices.

[16] "Eye Tracking Mouse Pointer Control System" by Rakshit, S., and Kalyani, N. (2018).This paper presents an eye tracking mouse pointer control system that allows users to

control the mouse pointer using eye movements. The system tracks the user's gaze to move the cursor on the screen and perform mouse actions such as clicking and dragging.

[17] "Design and Implementation of a Virtual Mouse System Using Eye Tracking" by Asif, M. S., and Rahman, M. A. (2019).The authors describe the design and implementation of a virtual mouse system that utilizes eye tracking technology. The system enables users to control the mouse cursor and perform mouse actions through eye movements, providing an alternative input method for individuals with physical disabilities.

[18] "EyeMouse: Real-Time Eye Tracking-Based Mouse Cursor Control System" by Huang, K. C., Cheng, C. H., and Chiang, M. C. (2017).This paper introduces EyeMouse, a real-time eye tracking-based mouse cursor control system. The system tracks the user's eye movements to move the mouse cursor on the screen and perform mouse actions, offering an intuitive and hands-free interaction method for computer users.

[19] "Using Eye Gaze Data to Enhance Cursor-Based Interaction" by Majaranta, P., and Bulling, A. (2014).The authors investigate the use of eye gaze data to augment traditional cursor-based interaction methods. They explore how eye gaze information can be integrated into existing interaction techniques to improve efficiency and usability.

[20] "EyeMouse: Efficient Eye-Controlled Mouse Cursor for Real-Time Applications" by Zhang, X., Guo, H., and Li, W. (2019).This paper presents EyeMouse, an efficient eye-controlled mouse cursor system designed for real-time applications. The system employs deep learning techniques to accurately track eye movements and control the mouse cursor with low latency.

[21] "GazeHover: Enhancing Hover-Based Interaction with Gaze" by Wu, T., Akcay, O., and Gellersen, H. (2018).The authors propose GazeHover, a system that enhances hover-based interaction by integrating gaze information. By combining cursor movement with eye gaze, the system enables more precise and context-aware interaction in graphical user interfaces.

CHAPTER 3

SOFTWARE REQUIREMENT ANALYSIS

3.1 SOFTWARE REQUIREMENTS

a) Python

Python is a powerful programming language and a higher – level language. Python has straightforward declarations in English that allow us to know the code without much expertise of it. With its use of a considerable whitespace, Python's structure allows readability of the code. It is also followed by indents that do not prevent the code from being analyzed or debugged. The object-oriented approach is followed by a clear overview of the logical and operational project part by the users. Projects do not have to be small.

Python Packages:

●**BeautifulSoup:** BeautifulSoup is a HTML and XML parsing Python package. It generates a parse tree for parsed pages, which is helpful in web scraping, in process of extracting HTML data. Leonard Richardson, who continues to contribute to the project and is approved by Tidelift, founded BeautifulSoup. Python 2.7 and Python 3 are both supported.
The Python library BeautifulSoup is designed for fast upturn, screen scraping projects. It is effective with three factors:

1. To traverse, browse, and change the parse tree, Beautiful Soup includes some easy ways as well as Pythonic languages.: a document deconstructs and obtain toolkit. To type an application does not take a lot of code

2. Beautiful Soup converts incoming and outgoing documents at an instant to Unicode and UTF-8. You must not be thinking of coding except if you stipulate an encoding in the document, nor can be detected by Beautiful Soup. The actual encoding then only must be specified.

3. BeautifulSoup is built on top of popular Python parsers like lxml and html5lib, allowing you to experiment with different parsing algorithms or trade speed for versatility.

- **Pprint:** The pprint module allows arbitrarily chosen Python data structures to be 'pretty-printed' in a form that can be used as the interpreter's input. The interpretation cannot be loaded if the structures configured include objects that do not include essential Python types. This could be the case if objects like files, sockets, classes and other objects were included in the that cannot be represented as Python literals.

  The formatted display holds items on a line if possible and tries to break them in multiple items if not within the width permitted. If you are to adjust the spacing constraint, build PrettyPrinter objects clearly and unambiguously

- **NumPy:** NumPy or Number Python is a library for certain array functions. It contains a n-dimensional array object useful for the whole array to apply algebra and different other mathematical formulas. The object NumPy can also be called the array NumPy. It stores information so that mathematical calculations are effectively carried out. In tabular format, NumPy Array store information, i.e., rows and columns. To use NumPy check if NumPy is or is not installed. If the NumPy package is not installed using the steps below. Go to the command prompt to setup NumPy. Enter and type the command below: "pip install numpy"

- **Pandas:** Pandas is a data manipulation and analysis programming language written in Python. It includes data structures and procedures for numerical tables and, in particular, time series manipulation. This programme is freely accessible under the three-clause BSD licence. The name derives from the term "panel data" which includes observations over multiple periods for the same people, an econometric term for data sets. The phrase "Python data analysis" itself is the reproduction of its name. Wes McKinney began building the AQR Capital pandas as a researcher between 2007 and 2010.

- **Sklearn:** Scikit-learn is a free machine learning package for Python. Python supports numerical and scientific libraries such as NumPy and SciPy, and it includes many techniques such as vector support machines, random forests, and k-neighbors. Scikit- Learn, usually known as sklearn, is Python's most popular machine learning library.

While other packages are found that work better on certain tasks, the versatility of Scikit-Learn makes it the best starting point for most ML problems. It's also an excellent beginner's library because it provides a high-level interface for a variety of activities (e.g., pre-processing data, cross-validation, etc.). This enables you to practice and comprehend the overall machine learning workflow. The project was launched as a Google Summer Code project in 2007 by David Cornopean, and has contributed numerous volunteers ever since. A team of volunteers is currently in charge of the project. Previously Scikit-learn was called Scikits.learn.

- **PyQt5:** PyQt5 is a set of Python bindings for the Qt application framework version 5 from The Qt Company. Qt is a suite of platform-independent C++ libraries and development tools for graphical user interfaces, networking, threads, regular expressions, SQL databases, SVG, OpenGL, XML, user and application settings, positioning and location services, and short-range communications (NFC and Bluetooth), web browsing, 3D animation, charts, 3D data visualization, and interacting with apps. PyQt5 implements almost 1000 of these classes as a set of Python modules. PyQt5 requires Python v3.5 or newer and runs on Windows, Linux,UNIX, Android, macOS, and iOS. PyQt5 also contains several utility programs

b) Interface – VS Code

Visual Studio Code is a free and open-source code editor developed by Microsoft. It is highly popular among developers for its lightweight yet powerful features, extensibility, and cross-platform support. Here are some key features and functionalities of Visual Studio Code:

- Cross-Platform: Visual Studio Code runs on Windows, macOS, and Linux, allowing developers to work seamlessly across different operating systems.
- Intuitive User Interface: It features a clean and intuitive user interface with customizable themes, syntax highlighting, and code folding, providing a comfortable coding environment.
- Integrated Terminal: Visual Studio Code includes an integrated terminal, allowing developers to execute commands, run scripts, and interact with the command-line interface without leaving the editor.
- Language Support: It provides built-in support for a wide range of programming languages, including JavaScript, TypeScript, Python, Java, C#, and many more. Language-specific features such as IntelliSense (code completion), syntax checking, and debugging are available

out-of-the-box or through extensions.

- Extensions Ecosystem: Visual Studio Code offers a rich ecosystem of extensions that extend its functionality and support additional programming languages, frameworks, and tools. Users can install extensions from the Visual Studio Code Marketplace to customize their development environment according to their needs.
- Version Control Integration: It seamlessly integrates with version control systems such as Git, enabling developers to manage source code repositories, view changes, and perform version control operations directly from the editor.
- Debugging Tools: Visual Studio Code provides powerful debugging capabilities with support for breakpoints, variable inspection, call stack navigation, and interactive debugging sessions for various programming languages and platforms.
- Task Automation: Developers can define and run tasks, such as build scripts or automated tests, using the built-in task runner or custom task configurations defined in the workspace settings.
- Live Share: Visual Studio Code includes Live Share functionality, allowing multiple developers to collaborate in real-time on the same codebase, share terminals, and debug together, regardless of their location.
- Accessibility: Accessibility features such as screen reader support, high contrast themes, and keyboard shortcuts make Visual Studio Code accessible to users with disabilities, ensuring an inclusive development experience.

Overall, Visual Studio Code is a versatile and feature-rich code editor that caters to the needs of developers across different programming languages and platforms. Its extensibility, ease of use, and powerful capabilities make it a popular choice for both individual developers and teams working on diverse software projects.

CHAPTER 4

SOFTWARE DESIGN

Unified Modelling Language is a tool that helps a designer present his ideas about the project to his client and his developer. Modeling plays a crucial role in designing software. A poorly designed model can lead to poorly developed software. A UML system contains five different views that can be used to describe systems from various angles. Each view has a set of diagrams and components that represent the real-time objects.A UML system contains five different views that can be used to describe systems from various angles. The real-time objects are represented by a set of diagrams and components in each view.

a. User Model View

b. Structural Model View

c. Behavioral Model View

d. Implementation Model View

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.
The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

4.1 USE CASE DIAGRAM

In UML, Use-case diagrams model a system's behavior and aid in capturing the system's needs. A use case diagram is a graphical representation of the interactions between a system and its actors, which are the external entities that interact with the system. It is a high-level view of the system that shows the system's functionality and the actors that interact with it.

A use case diagram consists of use cases, actors, and the relationships between them. A use case represents a specific functionality or behavior of the system, while an actor represents a user or external system that interacts with the system. The relationships between use cases and actors are represented by lines, which can be solid or dashed, depending on the type of relationship.

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

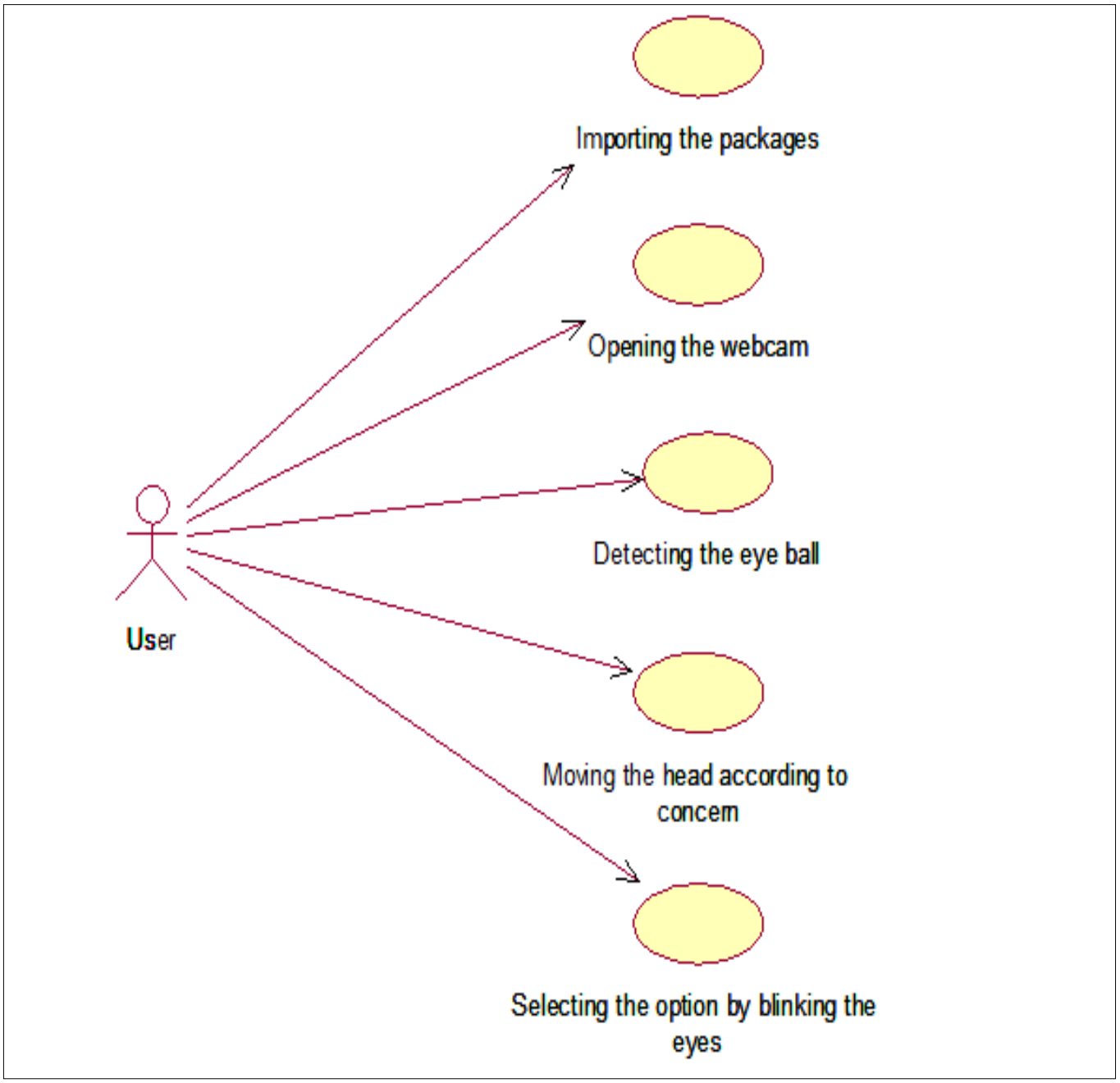

fig 4.1.1 Use Case Diagram

4.2 CLASS DIAGRAM

A static view of an application is depicted in the class diagram. It depicts the several types of things found in the system and the relationships between them. A class is made up of its objects, but it can also be inherited from other classes. A class diagram is used to represent, explain, and document numerous parts of a system and to create executable software code.

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

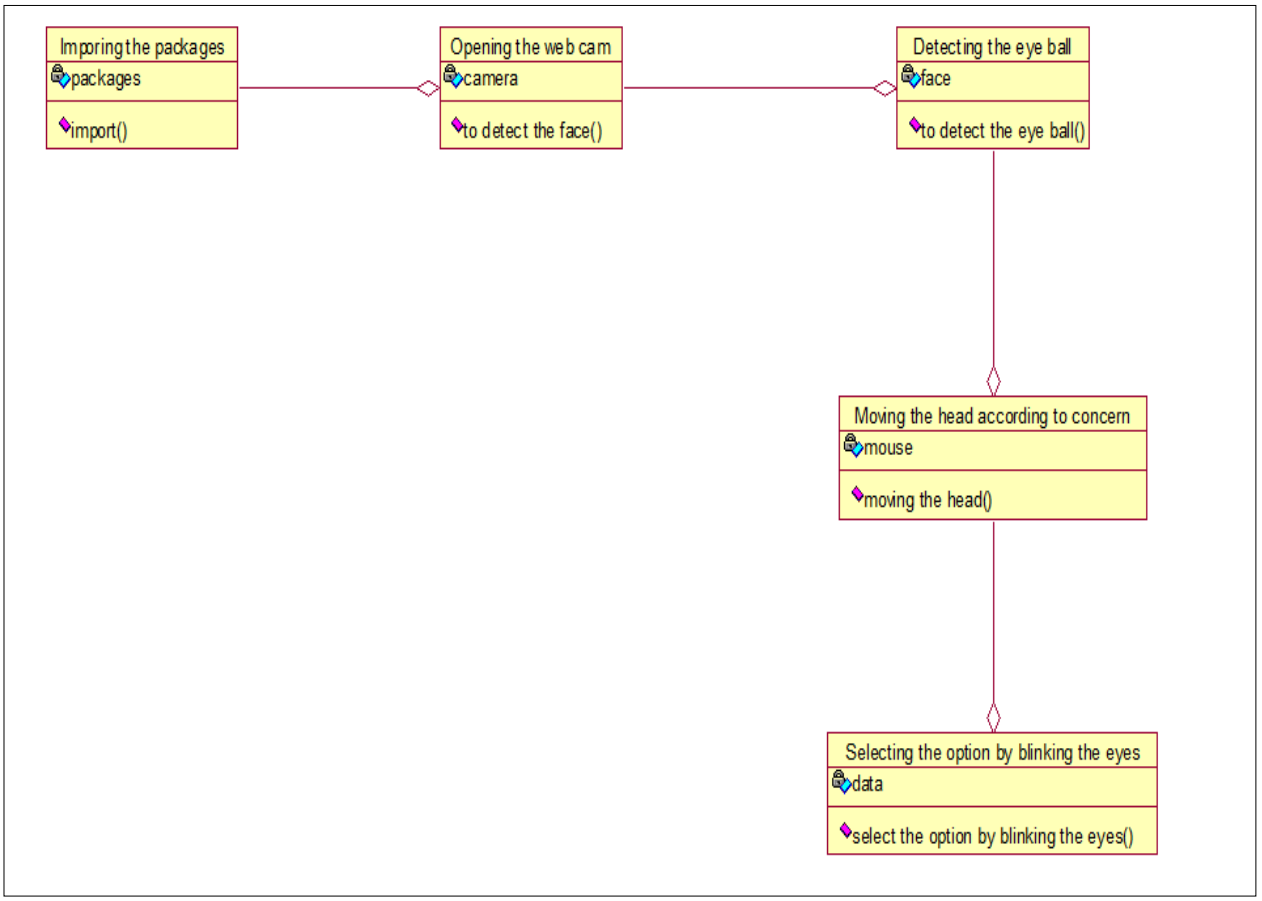

fig 4.2.1 Class Diagram

4.3 SEQUENCE DIAGRAM

The Sequence Diagram depicts the interaction of several elements in an application across time. It represents the sequence of communications that objects send to one another to perform the required functionality. It consists of the lifelines which are usually parallel vertical lines. It consists of horizontal arrows which indicates the direction of the messages that are exchanged in a proper order which makes the user easy to understand. The lifeline for a given object represents a role.

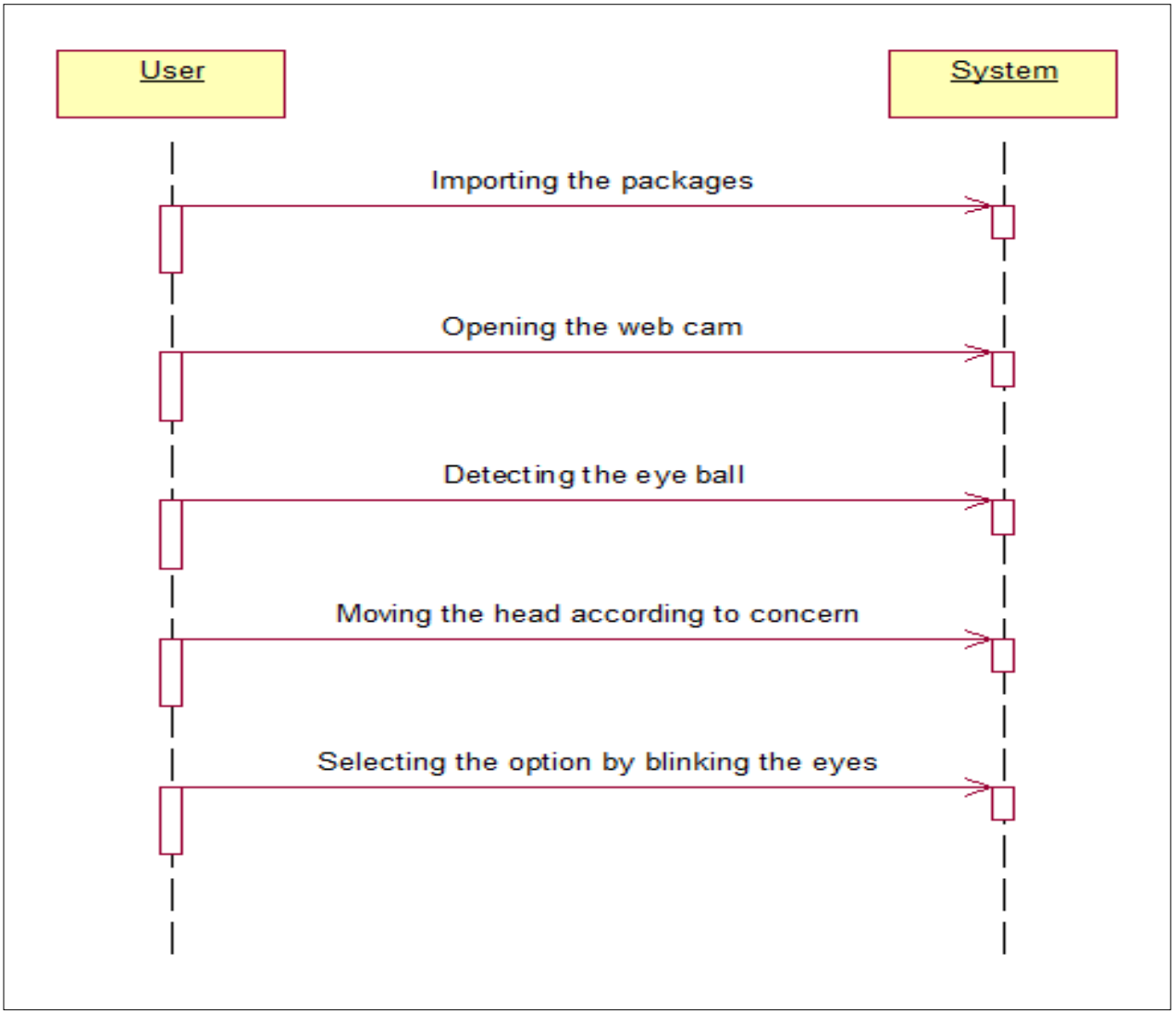

fig 4.3.1 Sequence Diagram

4.4 ACTIVITY DIAGRAM

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions. Another essential diagram in UML for describing the system's dynamic characteristics is the activity diagram. An activity diagram is a flowchart that illustrates the flow of information from one action to the next. The action can be described as a system operation.

The purpose of an activity diagram can be described as – •

Draw the activity flow of a system.

- Describe the sequence from one activity to another.
- Describe the system's parallel, branching, and concurrent flow.

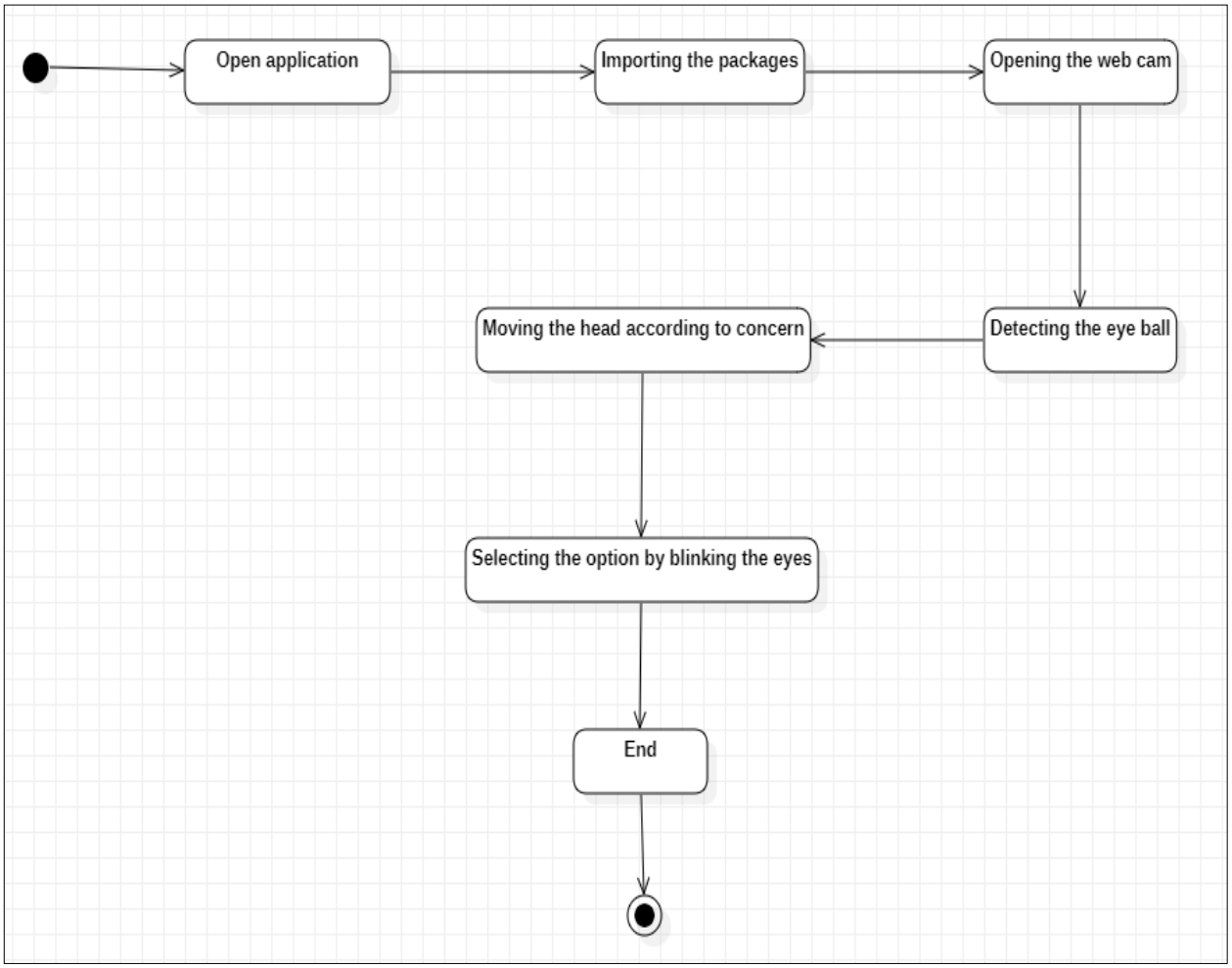

fig 4.4.1 Activity Diagram

4.5 FLOW CHART

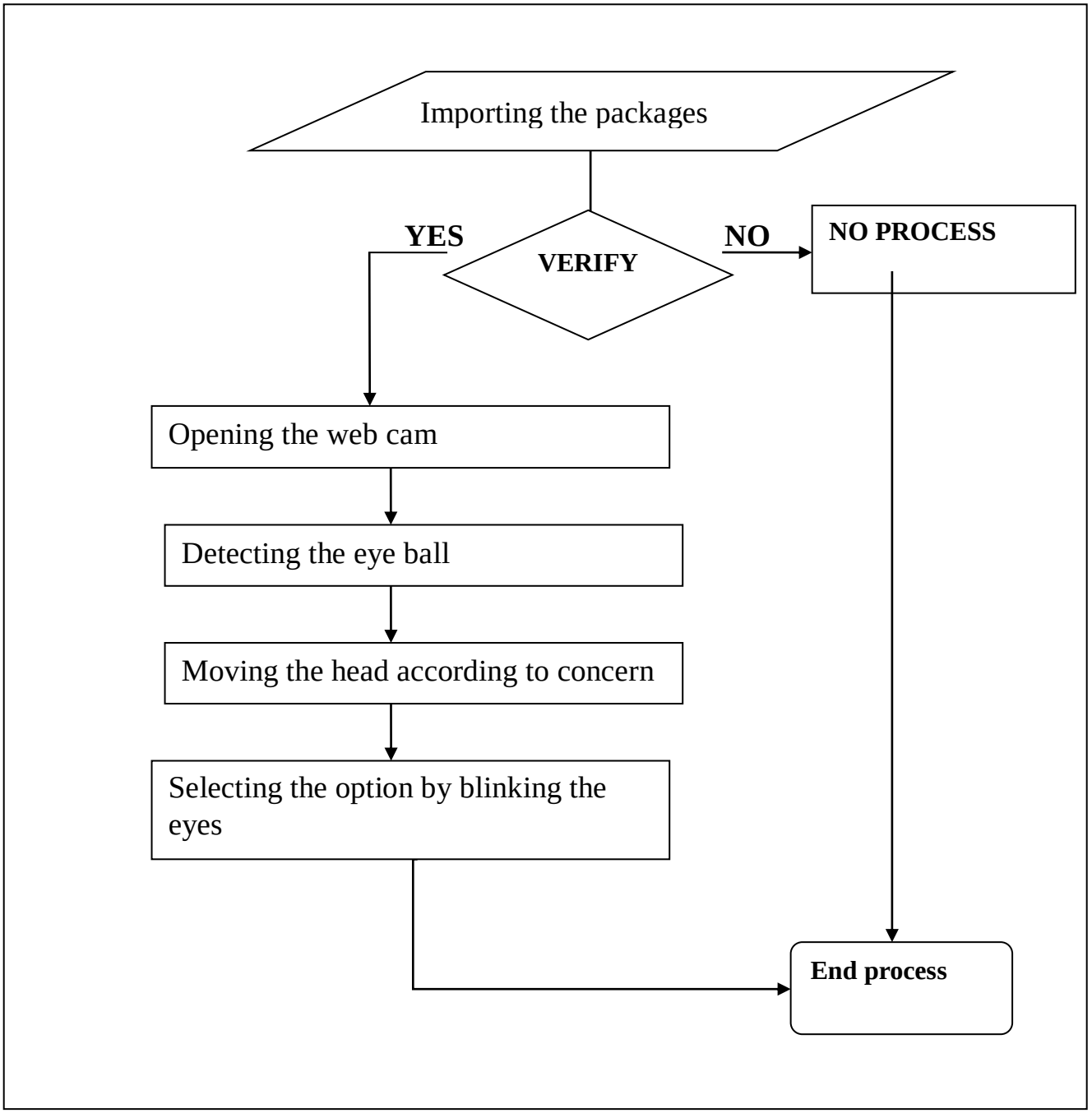

Fig 4.5.1 Flow diagram of the project

CHAPTER 5

PROPOSED SYSTEM

The proposed system is designed to address the specific needs of individuals with limited hand mobility through an innovative eye-based cursor movement approach. Leveraging OpenCV-based algorithms, the system aims to achieve precise and stable pupil position detection within the user's eyes, ensuring accurate cursor control. The initial step involves accessing the computer's webcam to capture real-time video or images, serving as the input source for eye tracking. The system employs computer vision techniques for eye detection, possibly incorporating face detection to narrow down the region of interest. To enhance accuracy, the project includes a module for tracking head movement, adjusting the cursor position based on the user's head orientation. The core functionality lies in interpreting eye movements, particularly blinking, as input commands for actions like selecting or clicking on-screen elements. Ultimately, this user-friendly solution is poised to foster independence and inclusivity in the digital realm, promoting equal opportunities for individuals with disabilities.

5.1 Advantages of proposed system:

1. Enables users with limited hand mobility to interact seamlessly with computers through eye-based cursor control.

2. Provides an intuitive and natural interaction method, fostering ease of use for individuals with disabilities.

3. Utilizes OpenCV algorithms for accurate and stable pupil position detection, ensuring reliable cursor control.

4. Incorporates head tracking to refine cursor positioning, enhancing overall system accuracy and user experience.

Tailored to unique needs, promoting inclusivity and empowering diverse user groups for equitable participation in digital activities.

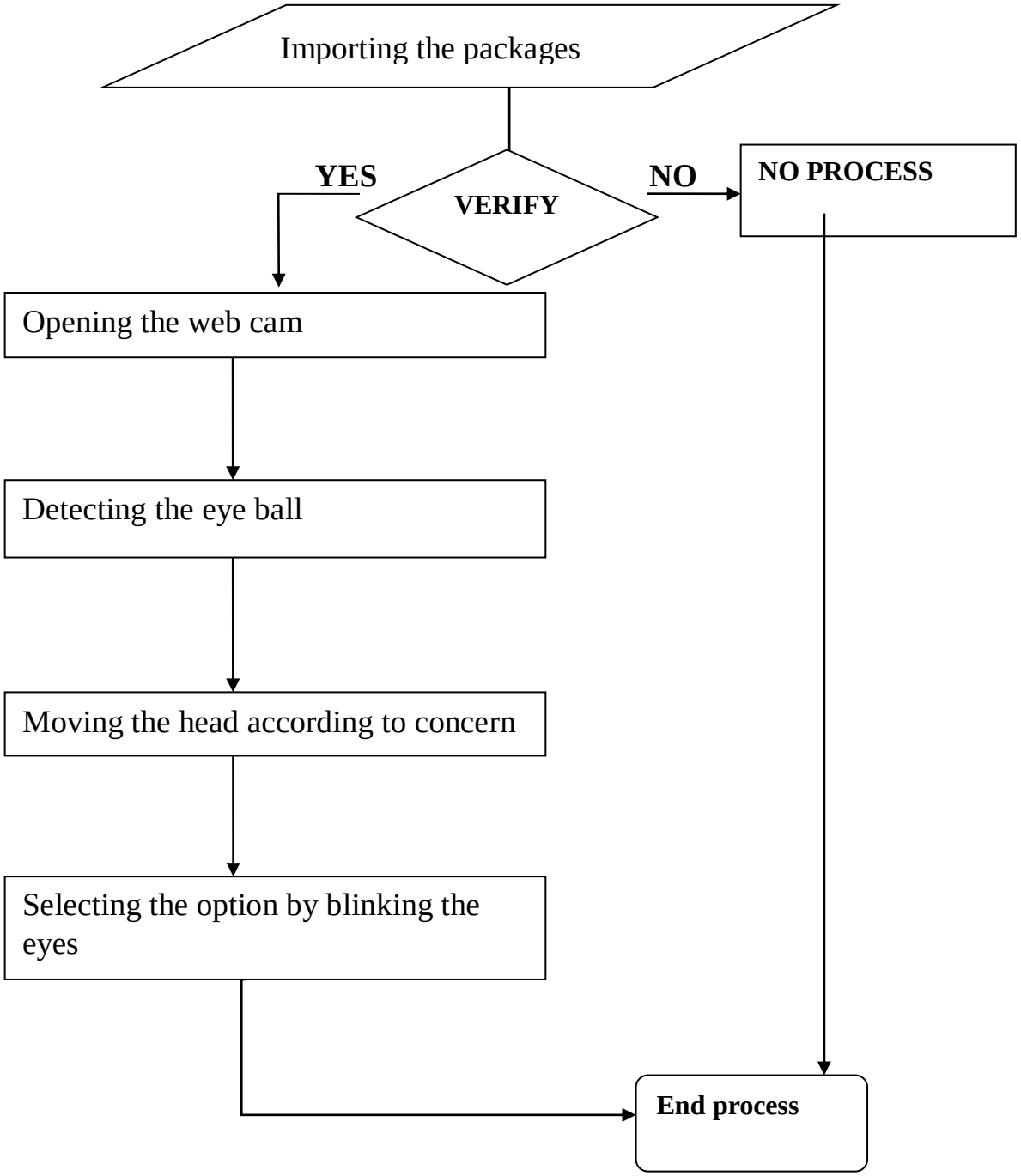

Fig 5.1.1 Data Flow Diagram

5.2 Open CV

- OpenCV: OpenCV (Open-Source Computer VisionLibrary) stands as a pivotal component in our system, facilitating real- time image processing and facial feature analysis crucial for sleep detection and wake-up alerts. The utilization of OpenCV is multifaceted within our system architecture, serving fundamental roles in several key processes.

- Facial Edge Detection:

  OpenCV's extensive collection of image processing functions enables efficient facial edge detection. This initial step involves capturing frames from the webcam input and implementing OpenCV's edge detection algorithms to precisely identify facial boundaries. This process lays the foundation for subsequent facial landmark identification.

- Facial Landmark Identification:

  The library's functionality is leveraged to employ the Facial Landmark Detector within the D-lib Library.OpenCV seamlessly integrates with D-lib to accurately identify critical facial landmarks, including the eyes and mouth regions. This meticulous identification of landmarks enables precise measurement and analysis of eye closure and mouth opening duration's, pivotal in detecting signs of drowsiness.

- Real-Time Analysis :

  The user has to sits in front of the display screen of private computer or pc, a specialized video camera established above the screen to study the consumer’s eyes. The laptop constantly analysis the video photo of the attention and determines wherein the consumer is calling at the display screen. not anything is attached to the consumer’s head or body. To “pick out” any key, the user seems at the key for a exact period of time and to “press” any key, the consumer just blink the eye. On this device,calibration procedure is not required. For this system enter is simplest eye. No outside hardware is connected or required.

- System Optimization:

  Camera gets the input from the eye. After receiving these streaming movies from the cameras, it'll split into frames.After receiving frames, it will check for lights conditions because cameras require enough lighting fixtures from external sources in any other case blunders

message will show at the screen. The captured frames which can be already in RGB mode are transformed into Black 'n' White. Five. Pics (frames) from the enter supply focusing the eye are analyzed for Iris (middle of eye)

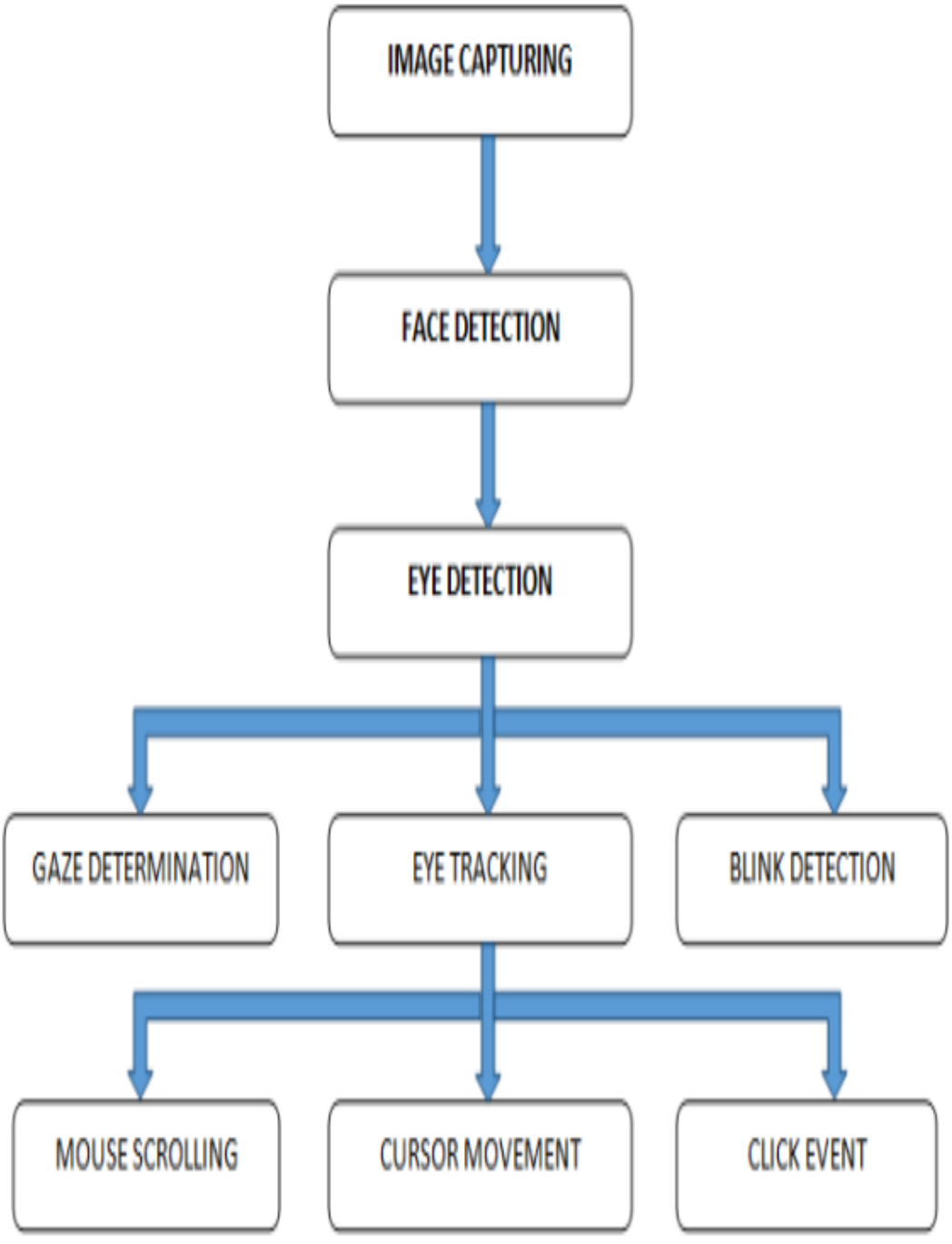

Fig 5.2.1 System Architecture

5.3 D-lib

D-LIB: Dlib, a versatile library, contributes significantly to the system by providing robust tools for facial landmark detection, crucial for accurate identification of key facial features. Within our project, Dlib serves as a fundamental element in the identification and localization of facial landmarks, specifically targeting the eyes and mouth regions.

i. Facial Landmark Detection: Dlib's Facial Landmark Detector incorporates a pre- trained model capable of precisely localizing facial landmarks within images or video frames. This detector accurately identifies crucial facial features,such as eye corners, eye centers, and mouth edges, enabling granular analysis of facial expressions and states.

ii. Integration with OpenCV: One of the key strengths of Dlib lies in its seamless integration with OpenCV, a core component of our system. Dlib's functionality for facial landmark detection is effortlessly interfaced with OpenCV's image processing capabilities. This integration allows for the efficient utilization of Dlib's facial landmark detector on frames captured through OpenCV'swebcam input.

iii. Precise Feature Identification: By leveraging Dlib's Facial Landmark Detector within our workflow, we achieve high precision in identifying specific facial landmarks crucial for sleep detection.The accurate identification of eye and mouth regions enables precise measurement of eye closure and mouth opening durations, pivotal indicators for moving cursor using eye moments.

iv. Detection of Eye: Vertical integral projection and horizontal projection are used to determine the exact position of the pupil. These projections subdivide entire image into homogeneous subsets. The proposed method employsan arbitrary threshold. Gaussian filter can be used to remove noise. The minimum gradient point is used to calculate the strong pixel value. The lower threshold protects the contrast region fromsplitting edges.The circular Hough transform is used to determine inner and outer boundaries.

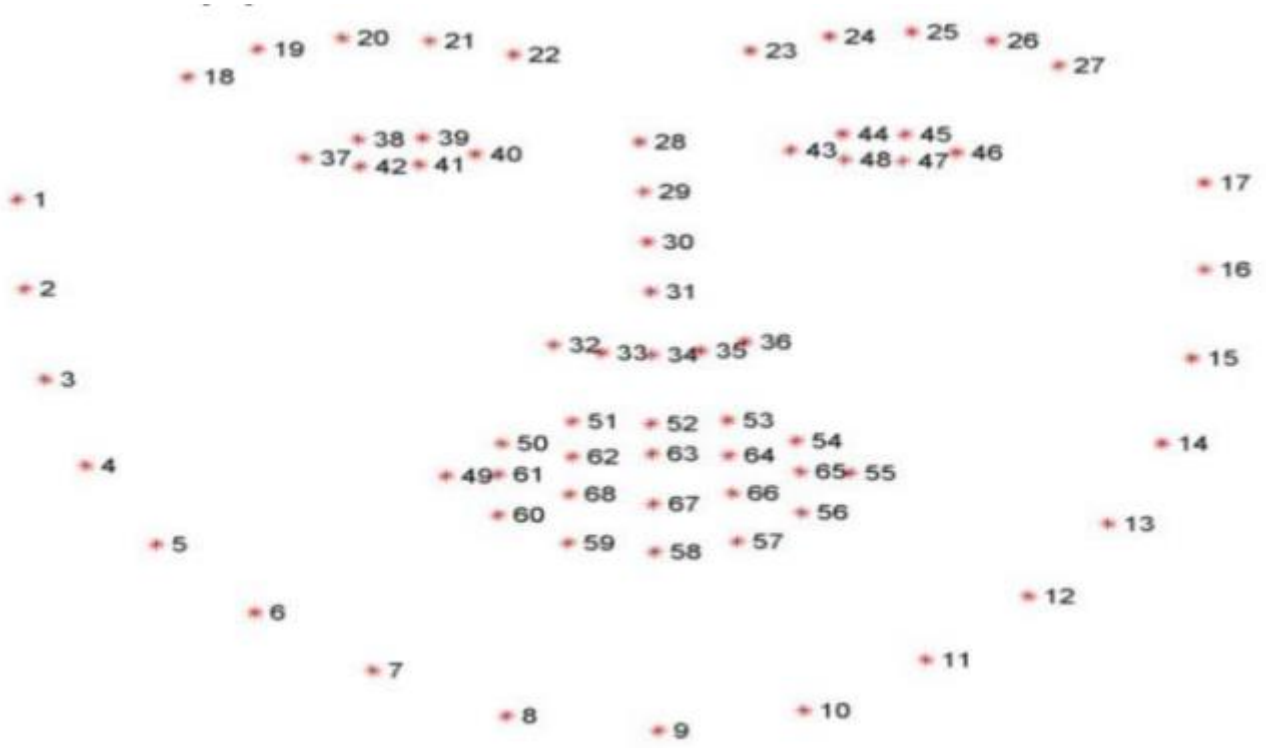

Fig 5.3.1 dlib 68 landmark

5.4 Integration

The integration of face recognition with the Computer vision-based eye detection and cursor moment in system introduces an advanced layer of functionality. This integration involves multiple sequential steps to facilitate individual identification and personalized alert mechanisms.

A. Thresholds and Parameters:Thresholds and consecutive frame lengths are defined for triggering mouse actions based on mouth, eye, and wink movement.

B. Initialization:Counters and Boolean variables are initialized to keep track of consecutive frames and indicate if specific actions areperformed.

C. Dlib Face Detector and Landmark Predictor:Dlib's face detector and facial landmark predictor are initialized using pre-trained models During real-time monitoring, extract facial features from webcam feed. Compare these features with the encoded features in the database using similarity metrics (e.g., cosine similarity, Euclidean distance) to identify known individuals.

D. Facial Landmarks Indices:Indices for facial landmarks corresponding to the left eye, right eye, nose, and mouth are obtained using face_utils.

E. Video Capture:OpenCV is used to capture video from the default camera.The frame is flipped, resized, and converted to grayscale.

F. Main Loop:The main loop continuously processes frames from the video feed. Faces are detected using the Dlib face detector, and facial landmarks are extracted.Eye aspect ratio (EAR), mouth aspect ratio (MAR),and other facial features are calculated.Blinking and wink detection logic is implemented tosimulate left and right mouse clicks.Mouth movement is tracked to toggle input mode and simulate mouse dragging.Cursor movement and scrolling are simulated based on the direction of facial movement.

G. Display:The processed frame is displayed, and additional information such as input mode and scroll mode status is overlaid.

H. User Interaction:The script responds to user actions, allowing for mouse clicks,scrolling, and cursor movement based on facial expressions.

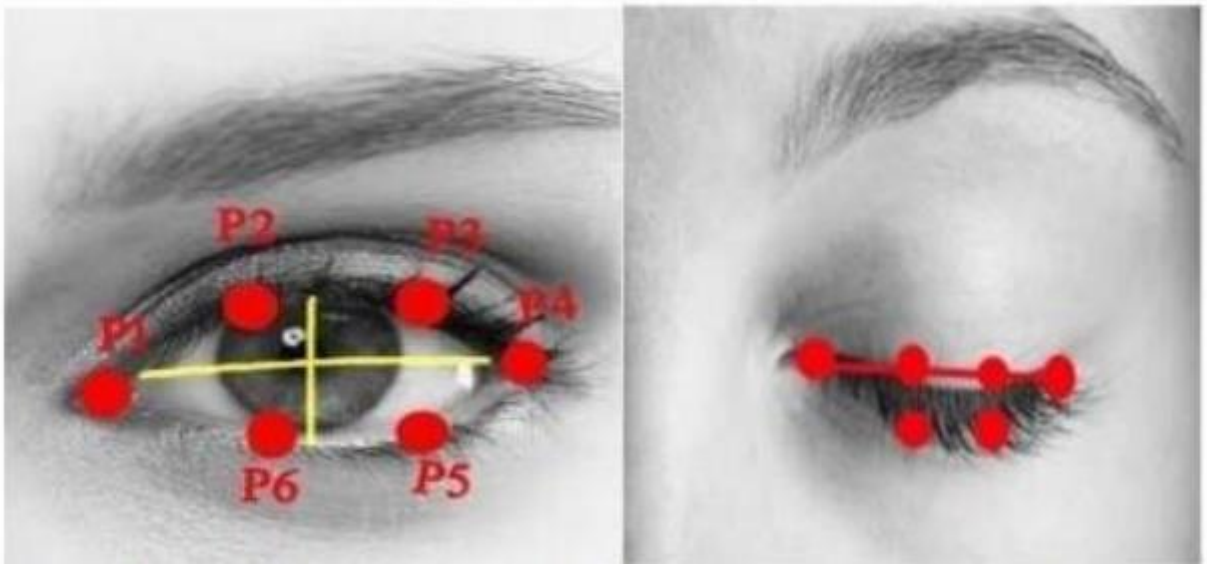

Fig 5.4.1 Landmarks of the eye when the eye is fully open (left) and landmarks of the eye when the eye is closed (right)

CHAPTER 6

IMPLEMENTATION

In this project we are instructing mouse cursor to change its location based on eye ball movement, in this application using OPENCV we will connect to webcam and then extract each frame from the webcam and pass to OPENCV to detect eye balls location. Once eye ball location detected then we can extract x and y coordinates of eye balls from OPENCV and then using python pyautogui API we can instruct mouse to change its current location to given eyeballs X and Y Coordinates. Below is the example to move mouse in python.

pyautogui.moveTo(int(data_x),int(data_y))

In above line moveTo function move cursor to given data_x and data_y location

To implement above concept we are using following modules

Video Recording: Using this module we will connect application to webcam using OPENCV built-in function called VideoCapture.

Frame Extraction: Using this module we will grab frames from webcam and then extract each picture frame by frame and send that frame to GazeTracking.

GazeTracking: Using this module we can detect eyeballs and the extract x and y coordinates of both left and right pupil.

MoveCursor: Using this module we will instruct mouse to change its current location to given new x and y location.

To stop video recording from webcam press ‘Esc’ key.

OpenCV is an artificial intelligence API available in python to perform various operation on images/videos such as image recognition, face detection, eye detection/eye ball tracking and convert images to gray or coloured images etc. This API written in C++ languages and then make C++ functions available to call from python using native language programming. Steps involved in face detection using OpenCV.

Face Detection/Eye Detection Using OpenCV.This seems complex at first but it is very easy. Let me walk you through the entire process and you will feel the same.

Step 1: Considering our prerequisites, we will require an image, to begin with. Later we need

to create a cascade classifier which will eventually give us the features of the face.

Step 2: This step involves making use of OpenCV which will read the image and the features file. So at this point, there are NumPy arrays at the primary data points.

All we need to do is to search for the row and column values of the face NumPyN dimensional array. This is the array with the face rectangle coordinates.
Step 3: This final step involves displaying the image with the rectangular face box.

CODE:

```
import cv2

import numpy as np

import pyautogui

# Initialize OpenCV face and eye classifiers

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')

# Initialize the webcam

cap = cv2.VideoCapture(0)

while True:

    ret, frame = cap.read()

    # Convert the frame to grayscale

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect faces in the frame

    faces = face_cascade.detectMultiScale(gray, 1.3, 5)

    for (x, y, w, h) in faces:

        # Draw a rectangle around the face
```

```
    cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
    # Extract the region of interest (ROI) for eyes
    roi_gray = gray[y:y + h, x:x + w]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex, ey, ew, eh) in eyes:
        # Draw a rectangle around each eye
        cv2.rectangle(frame, (x + ex, y + ey), (x + ex + ew, y + ey + eh), (0, 255, 0), 2)
        # Simulate cursor movement based on eye position
        eye_center_x = x + ex + ew // 2
        eye_center_y = y + ey + eh // 2
        # Adjust these scaling factors based on your screen resolution
        screen_width, screen_height = pyautogui.size()
        scale_x = screen_width / w
        scale_y = screen_height / h
        # Move the cursor
        pyautogui.moveTo(eye_center_x * scale_x, eye_center_y * scale_y)
        # Check for a blink (you may need to adjust the threshold)
        if eh < 10:
            pyautogui.click()

# Display the resulting frame
cv2.imshow('Eye-Based Cursor Control', frame)
# Break the loop when 'q' is pressed
```

```
    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

# Release the webcam and close all windows

cap.release()

cv2.destroyAllWindows()
```

Eye detection.py:

```
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
import numpy as np
import argparse
import imutils
import dlib
import cv2
import sched
import time
import sys
import Quartz
import math
from datetime import datetime, date
class Mouse():
        down = [Quartz.kCGEventLeftMouseDown, Quartz.kCGEventRightMouseDown, Quartz.kCGEventOtherMouseDown]
            up = [Quartz.kCGEventLeftMouseUp, Quartz.kCGEventRightMouseUp, Quartz.kCGEventOtherMouseUp]
    [LEFT, RIGHT, OTHER] = [0, 1, 2]
    def position(self):
        point = Quartz.CGEventGetLocation( Quartz.CGEventCreate(None) )
        return point.x, point.y


    def __mouse_event(self, type, x, y):
                mouse_event = Quartz.CGEventCreateMouseEvent(None, type, (x, y), Quartz.kCGMouseButtonLeft)
```

```
        Quartz.CGEventPost(Quartz.kCGHIDEventTap, mouse_event)

    def move(self, x, y):
        self.__mouse_event(Quartz.kCGEventMouseMoved, x, y)
        Quartz.CGWarpMouseCursorPosition((x, y))

    def press(self, x, y, button=0):
        event = Quartz.CGEventCreateMouseEvent(None, Mouse.down[button], (x, y), button)
        Quartz.CGEventPost(Quartz.kCGHIDEventTap, event)

    def release(self, x, y, button=0):
        event = Quartz.CGEventCreateMouseEvent(None, Mouse.up[button], (x, y), button)
        Quartz.CGEventPost(Quartz.kCGHIDEventTap, event)

    def doubleClick(self, x, y, clickCount, button=0):
        print("Double click event")
            theEvent = Quartz.CGEventCreateMouseEvent(None, Mouse.down[button], (x, y),
button)
             Quartz.CGEventSetIntegerValueField(theEvent, Quartz.kCGMouseEventClickState,
clickCount)
        Quartz.CGEventPost(Quartz.kCGHIDEventTap, theEvent)
        Quartz.CGEventSetType(theEvent, Quartz.kCGEventLeftMouseUp)
        Quartz.CGEventPost(Quartz.kCGHIDEventTap, theEvent)
        Quartz.CGEventPost(Quartz.kCGHIDEventTap, theEvent)
        Quartz.CGEventSetType(theEvent, Quartz.kCGEventLeftMouseDown)
        Quartz.CGEventPost(Quartz.kCGHIDEventTap, theEvent)
        Quartz.CGEventSetType(theEvent, Quartz.kCGEventLeftMouseUp)
        Quartz.CGEventPost(Quartz.kCGHIDEventTap, theEvent)
        print("Double click event ended")
    def click(self, button=0):
        x, y = self.position()
        self.press(x, y, button)
        self.release(x, y, button)

    def click_pos(self, x, y, button=0):
        self.move(x, y)
```

```
        self.click(button)

    def torelative(self, x, y):
        curr_pos = Quartz.CGEventGetLocation( Quartz.CGEventCreate(None) )
        x += curr_pos.x;
        y += curr_pos.y;
        return [x, y]

    def move_rel(self, x, y):
        [x, y] = self.torelative(x, y)
                                moveEvent = Quartz.CGEventCreateMouseEvent(None,
Quartz.kCGEventMouseMoved, Quartz.CGPointMake(x, y), 0)
        Quartz.CGEventPost(Quartz.kCGHIDEventTap, moveEvent)

    def mouseEvent(self, type, posx, posy):
                theEvent = Quartz.CGEventCreateMouseEvent(None, type, (posx,posy),
Quartz.kCGMouseButtonLeft)
        Quartz.CGEventPost(Quartz.kCGHIDEventTap, theEvent)

    def mousedrag(self, posx, posy):
        self.mouseEvent(Quartz.kCGEventLeftMouseDragged, posx,posy)


def eye_aspect_ratio(eye):
    # compute the euclidean distances between the two sets of
    # vertical eye landmarks (x, y)-coordinates
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])

    # compute the euclidean distance between the horizontal
    # eye landmark (x, y)-coordinates
    C = dist.euclidean(eye[0], eye[3])

    # compute the eye aspect ratio
    ear = (A + B) / (2.0 * C)
```

```
    # return the eye aspect ratio
    return ear


# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape-predictor", required=True,
                help="path to facial landmark predictor")
args = vars(ap.parse_args())
EYE_AR_THRESH = 0.3

# initialize the frame counters and the total number of blinks
COUNTER = 0
TOTAL = 0

s = sched.scheduler(time.time, time.sleep)

def print_time():
    if ear > .3:
        COUNTER = 0

def print_some_times():
    print (time.time())
    s.enter(1, 1, print_time, ())
    s.run()
    print (time.time())


# initialize dlib's face detector (HOG-based) and then create
# the facial landmark predictor
print("Loading facial landmark predictor...")
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])

# grab the indexes of the facial landmarks for the left and
# right eye, respectively
```

```
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]


print("Starting live video stream...")

vs = VideoStream(src=0).start()

fileStream = False
time.sleep(1.0)
currentCount = 0

mouse = Mouse()
face_cascade = cv2.CascadeClassifier('res/haarcascade_frontalface_default.xml')

while True:
    if fileStream and not vs.more():
        break
    frame = vs.read()
    frame = imutils.resize(frame, width=450)

    height, width, c = frame.shape

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cv2.circle(frame, ((int)(width/2),(int)(height/2)), 4, (0,0,255), 2)
    cv2.circle(frame, ((int)(width/2),(int)(height/2)), 20, (128,0,128), 2)
    face = face_cascade.detectMultiScale(gray, 1.15)

    min_dis = 100000
    x=0
    y=0
    w=0
    h=0
    for (mx, my, mw, mh) in face:
        d = dist.euclidean((mx+w/2, my+h/2), (width/2, height/2))
        if(d < min_dis):
```

```
            min_dis = d
            x = mx
            y = my
            w = mw
            h = mh


    if(x!=0 and y!=0 and w!=0 and h!=0):
        cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
        #to avoid mirror image
        x = width - (x + w)
        slope = math.atan((float)((y+h/2 - height/2)/(x+w/2 - width/2)))

        r = (int)(width/2 + 100 * math.cos(slope))
        t = (int)(height/2 + 100 * math.sin(slope))

        cv2.line(frame, ((int)(width/2), (int)(height/2)), ((int)(x+w/2), (int)(y+h/2)), (255,0,0), 2)
        cv2.circle(frame, ((int)(x+w/2), (int)(y+h/2)), 3, (255,0,0), 2)
        d = dist.euclidean((x+w/2, y+h/2), (width/2, height/2))
        c, e = mouse.position()

        if(d>20):
            speed = 5
            if(x+w/2 > width/2):
                mouse.move(c + speed * math.cos(slope), e + speed * math.sin(slope))
            else :
                mouse.move(c - speed * math.cos(slope), e - speed * math.sin(slope))


    # detect faces in the grayscale frame
    rects = detector(gray, 0)
    prevcount = 0
    # loop over the face detections
    for rect in rects:
        # 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)



leftEye = shape[lStart:lEnd]
rightEye = shape[rStart:rEnd]
leftEAR = eye_aspect_ratio(leftEye)
rightEAR = eye_aspect_ratio(rightEye)

# average the eye aspect ratio together for both eyes\
ear = (leftEAR + rightEAR) / 2.0

leftEyeHull = cv2.convexHull(leftEye)
rightEyeHull = cv2.convexHull(rightEye)
cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
cv2.drawContours(frame, [rightEyeHull],  -1, (0, 255, 0), 1)

if leftEAR < EYE_AR_THRESH - 0.12 and rightEAR > EYE_AR_THRESH - 0.12:
    print ("Left Eye Blinked")
    m,n = mouse.position()
    mouse.click_pos(m, n, 0)
    time.sleep(1)

elif rightEAR < EYE_AR_THRESH - 0.12 and leftEAR > EYE_AR_THRESH - 0.12:
    print ("Right Eye Blinked")
    m,n = mouse.position()
    mouse.click_pos(m, n, 1)
    time.sleep(1)

if (leftEAR < EYE_AR_THRESH - 0.12 and rightEAR < EYE_AR_THRESH - 0.12):
    print("Both Eyes Blinked")
    m,n = mouse.position()
    mouse.doubleClick(m, n, 2, 0)
    COUNTER += 1
    TOTAL += 1
```

```
            prevcount = COUNTER
            time.sleep(1)

        # the computed eye aspect ratio for the frame

        cv2.putText(frame, "Blinks: {}".format(TOTAL), (10, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        cv2.putText(frame, "Left: {:.2f}".format(leftEAR), (10, 50),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        cv2.putText(frame, "Right: {:.2f}".format(rightEAR), (10, 70),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

        break


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

    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break

cv2.destroyAllWindows()
vs.stop()
```

app.py:

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

```
# Thresholds and consecutive frame length for triggering the mouse action.
MOUTH_AR_THRESH = 0.6
MOUTH_AR_CONSECUTIVE_FRAMES = 15
EYE_AR_THRESH = 0.19
EYE_AR_CONSECUTIVE_FRAMES = 15
WINK_AR_DIFF_THRESH = 0.04
WINK_AR_CLOSE_THRESH = 0.19
WINK_CONSECUTIVE_FRAMES = 10
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)
shape_predictor = "model/shape_predictor_68_face_landmarks.dat"
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(shape_predictor)
 (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"]
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:
    _, 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:
                pyag.click(button='left')

                WINK_COUNTER = 0

    elif leftEAR > rightEAR:
```

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

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

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

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

                    # nose point to draw a bounding box around it

            else:
                EYE_COUNTER = 0
                WINK_COUNTER = 0

        if mar > MOUTH_AR_THRESH:
            MOUTH_COUNTER += 1

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

        else:
            MOUTH_COUNTER = 0
```

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

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

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


    cv2.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()
```

Utils.py:

```
import numpy as np


# Returns EAR given eye landmarks
def eye_aspect_ratio(eye):
    # Compute the euclidean distances between the two sets of
    # vertical eye landmarks (x, y)-coordinates
    A = np.linalg.norm(eye[1] - eye[5])
    B = np.linalg.norm(eye[2] - eye[4])

    # Compute the euclidean distance between the horizontal
    # eye landmark (x, y)-coordinates
    C = np.linalg.norm(eye[0] - eye[3])

    # Compute the eye aspect ratio
    ear = (A + B) / (2.0 * C)

    # Return the eye aspect ratio
    return ear


# Returns MAR given eye landmarks
def mouth_aspect_ratio(mouth):
```

```
    # Compute the euclidean distances between the three sets
    # of vertical mouth landmarks (x, y)-coordinates
    A = np.linalg.norm(mouth[13] - mouth[19])
    B = np.linalg.norm(mouth[14] - mouth[18])
    C = np.linalg.norm(mouth[15] - mouth[17])

    # Compute the euclidean distance between the horizontal
    # mouth landmarks (x, y)-coordinates
    D = np.linalg.norm(mouth[12] - mouth[16])

    # Compute the mouth aspect ratio
    mar = (A + B + C) / (2 * D)

    # Return the mouth aspect ratio
    return mar


# Return direction given the nose and anchor points.
def direction(nose_point, anchor_point, w, h, multiple=1):
    nx, ny = nose_point
    x, y = anchor_point

    if nx > x + multiple * w:
        return 'right'
    elif nx < x - multiple * w:
        return 'left'

    if ny > y + multiple * h:
        return 'down'
    elif ny < y - multiple * h:
        return 'up'

    return 'none'
```

Mouse.py:

```
import sys
import time
import Quartz

class Mouse():
        down = [Quartz.kCGEventLeftMouseDown, Quartz.kCGEventRightMouseDown,
Quartz.kCGEventOtherMouseDown]
            up = [Quartz.kCGEventLeftMouseUp, Quartz.kCGEventRightMouseUp,
Quartz.kCGEventOtherMouseUp]
    [LEFT, RIGHT, OTHER] = [0, 1, 2]

    def position(self):
        point = Quartz.CGEventGetLocation( Quartz.CGEventCreate(None) )
        return point.x, point.y

    def __mouse_event(self, type, x, y):
                mouse_event = Quartz.CGEventCreateMouseEvent(None, type, (x, y),
Quartz.kCGMouseButtonLeft)
        Quartz.CGEventPost(Quartz.kCGHIDEventTap, mouse_event)

    def move(self, x, y):
        self.__mouse_event(Quartz.kCGEventMouseMoved, x, y)
        Quartz.CGWarpMouseCursorPosition((x, y))

    def press(self, x, y, button=0):
        event = Quartz.CGEventCreateMouseEvent(None, Mouse.down[button], (x, y), button)
        Quartz.CGEventPost(Quartz.kCGHIDEventTap, event)

    def release(self, x, y, button=0):
        event = Quartz.CGEventCreateMouseEvent(None, Mouse.up[button], (x, y), button)
        Quartz.CGEventPost(Quartz.kCGHIDEventTap, event)

    def doubleClick(self, x, y, clickCount, button=0):
        print("Double click event")
```

```
        theEvent = Quartz.CGEventCreateMouseEvent(None, Mouse.down[button], (x, y),
button)
        Quartz.CGEventSetIntegerValueField(theEvent, Quartz.kCGMouseEventClickState,
clickCount)
    Quartz.CGEventPost(Quartz.kCGHIDEventTap, theEvent)
    Quartz.CGEventSetType(theEvent, Quartz.kCGEventLeftMouseUp)
    Quartz.CGEventPost(Quartz.kCGHIDEventTap, theEvent)
    Quartz.CGEventPost(Quartz.kCGHIDEventTap, theEvent)
    Quartz.CGEventSetType(theEvent, Quartz.kCGEventLeftMouseDown)
    Quartz.CGEventPost(Quartz.kCGHIDEventTap, theEvent)
    Quartz.CGEventSetType(theEvent, Quartz.kCGEventLeftMouseUp)
    Quartz.CGEventPost(Quartz.kCGHIDEventTap, theEvent)
    print("Double click event ended")


  def click(self, button=0):
    x, y = self.position()
    self.press(x, y, button)
    self.release(x, y, button)

  def click_pos(self, x, y, button=0):
    self.move(x, y)
    self.click(button)

  def torelative(self, x, y):
    curr_pos = Quartz.CGEventGetLocation( Quartz.CGEventCreate(None) )
    x += curr_pos.x;
    y += curr_pos.y;
    return [x, y]

  def move_rel(self, x, y):
    [x, y] = self.torelative(x, y)
                    moveEvent = Quartz.CGEventCreateMouseEvent(None,
Quartz.kCGEventMouseMoved, Quartz.CGPointMake(x, y), 0)
    Quartz.CGEventPost(Quartz.kCGHIDEventTap, moveEvent)
```

```
    def mouseEvent(self, type, posx, posy):
                theEvent = Quartz.CGEventCreateMouseEvent(None, type, (posx,posy),
Quartz.kCGMouseButtonLeft)
        Quartz.CGEventPost(Quartz.kCGHIDEventTap, theEvent)

    def mousedrag(self, posx, posy):
        self.mouseEvent(Quartz.kCGEventLeftMouseDragged, posx,posy)

if __name__ == '__main__':
    mouse = Mouse()
    if sys.platform == "darwin":
        print("Current mouse position: %d:%d" % mouse.position())
        mouse.move_rel(25, 16)
        print("Clickingthe right button...");
        mouse.move(25, 26)
        time.sleep(0.05)
        mouse.move(35, 26)
        time.sleep(0.05)
        mouse.move(40, 26)
        time.sleep(0.05)
        mouse.move(44, 26)
        time.sleep(0.05)
        mouse.move(50, 26)
        time.sleep(0.05)
        mouse.move(55, 26)
        time.sleep(0.05)
        mouse.doubleClick(1264, 416, 2, 0)
        time.sleep(0.05)
        mouse.click_pos(1264, 416, 1)
        mouse.doubleClick(1264, 46, 2, 0)

    elif sys.platform == "win32":
        print("Error: Platform not supported!")
```

CHAPTER 7

TESTING

Testing cases for a project involving cursor movements and actions using eyeball tracking:

Basic Cursor Movement:

- Verify that the cursor moves smoothly and accurately according to the user's eye movements.
- Test different eye movement speeds to ensure the cursor responds appropriately.

Precision Testing:

- Test the accuracy of cursor placement by moving the cursor to specific points on the screen.
- Verify that the cursor stops accurately when the user fixates on a particular point.

Boundary Testing:

- Test the behavior of the cursor when reaching the screen boundaries.
- Verify that the cursor does not move beyond the screen edges.

Click Testing:

- Test the click functionality by fixating on an object and verifying that a click action is triggered.
- Test double-click functionality by fixating on an object twice in quick succession.

Drag and Drop Testing:

- Verify that dragging and dropping objects on the screen works accurately using eye movements.
- Test dragging objects to different locations on the screen and dropping them successfully.

Scrolling Testing:

- Test scrolling functionality by fixating on scroll bars and verifying that scrolling occurs in the intended direction.
- Verify that scrolling speed is appropriate and controllable based on eye movement speed.

Error Handling:

- Test error scenarios such as sudden changes in ambient light or calibration issues.
- Verify that the system provides appropriate feedback and recalibrates if necessary.

Compatibility Testing:

- Test the system with different screen sizes and resolutions to ensure compatibility.
- Verify that the system works effectively on various devices and platforms.

User Interface Testing:

- Test the visibility and usability of the cursor and other interface elements.
- Verify that the interface design accommodates users with different eye-tracking abilities.

Performance Testing:

- Test the system's performance under heavy usage, including scenarios with multiple users or intensive eye-tracking tasks.
- Measure the system's response time and resource utilization.

Accessibility Testing:

- Test the system's accessibility features, such as options for adjusting cursor sensitivity or interface elements for users with disabilities.
- Verify compatibility with screen reader software and other accessibility tools.

Integration Testing:

- Test integration with other software or systems, such as operating systems, applications, or assistive technology.
- Verify that the system works seamlessly with other components without any conflicts or compatibility issues.
- These testing cases cover a range of scenarios to ensure the reliability, accuracy, usability, and compatibility of the cursor movement and actions using eyeball tracking. Adjust and expand upon these cases based on the specific requirements and features of your project.

CHAPTER 8

RESULTS

In this part, the results of implementing the proposed system are presented.

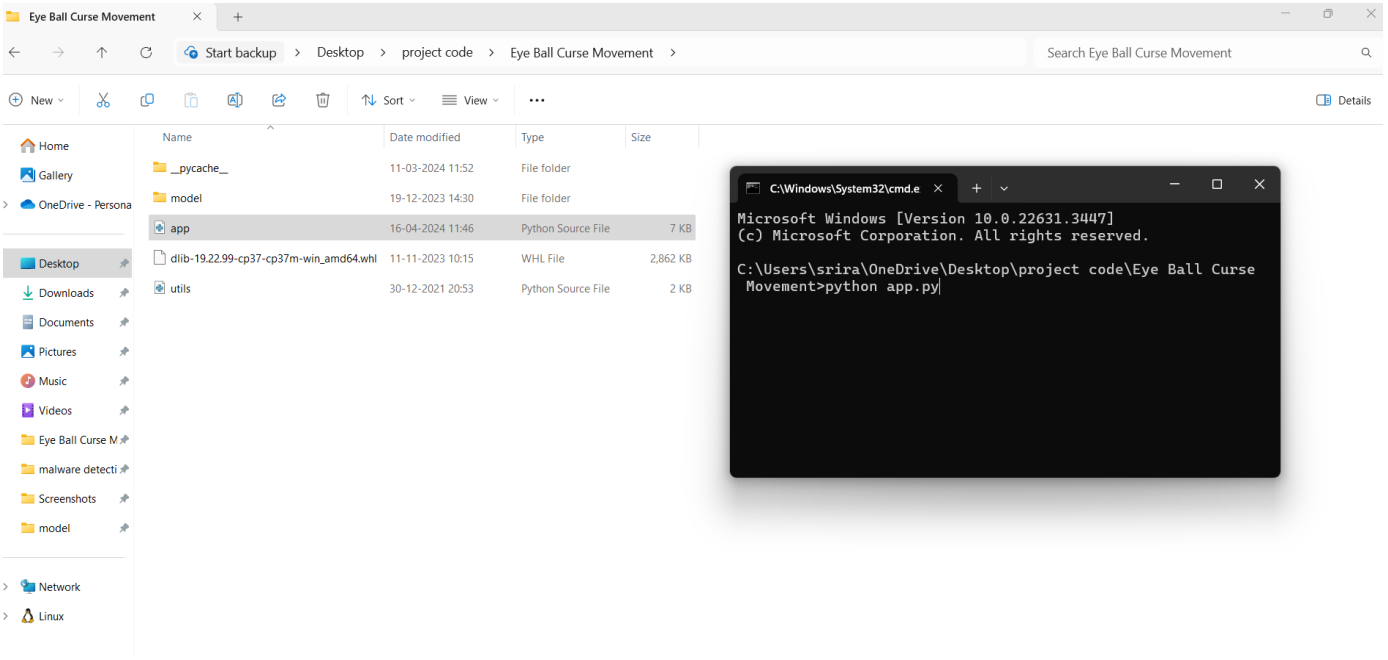

Fig 8.1.1 Output - 1

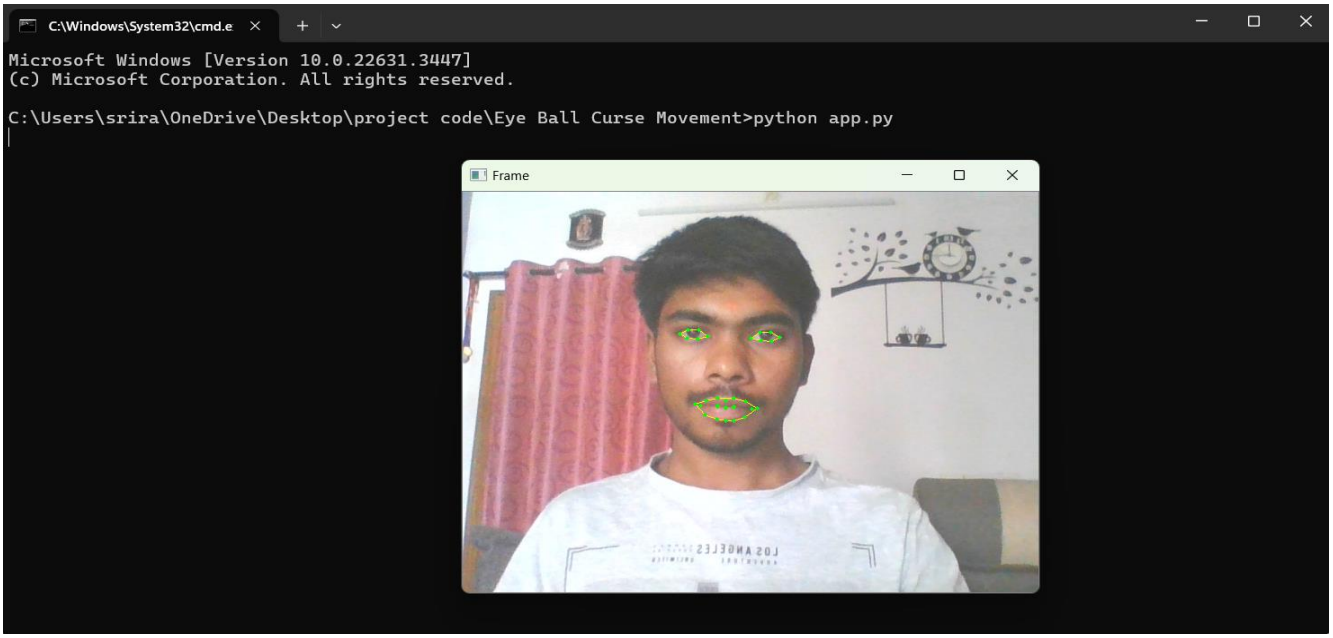

Fig 8.1.2 Output - 2

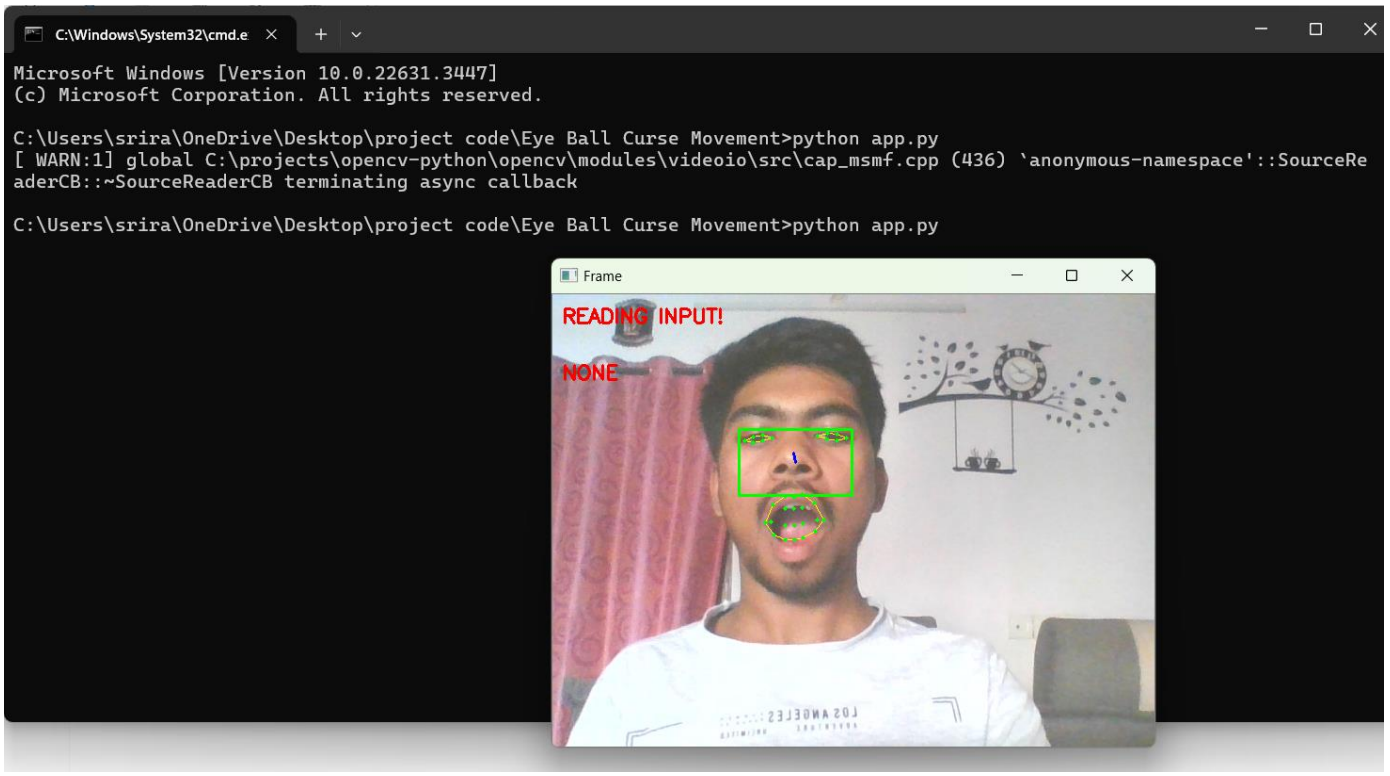

Fig 8.1.3 Output - 3

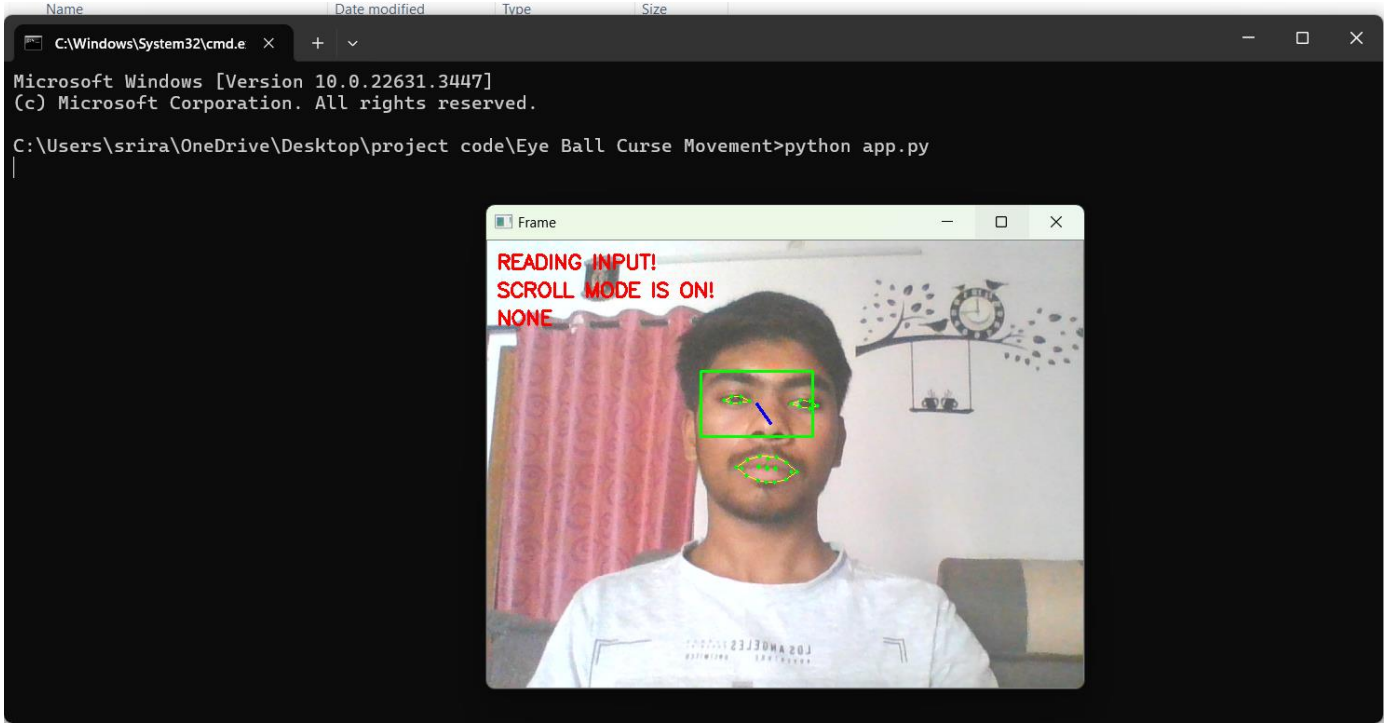

Fig 8.1.4 Output - 4

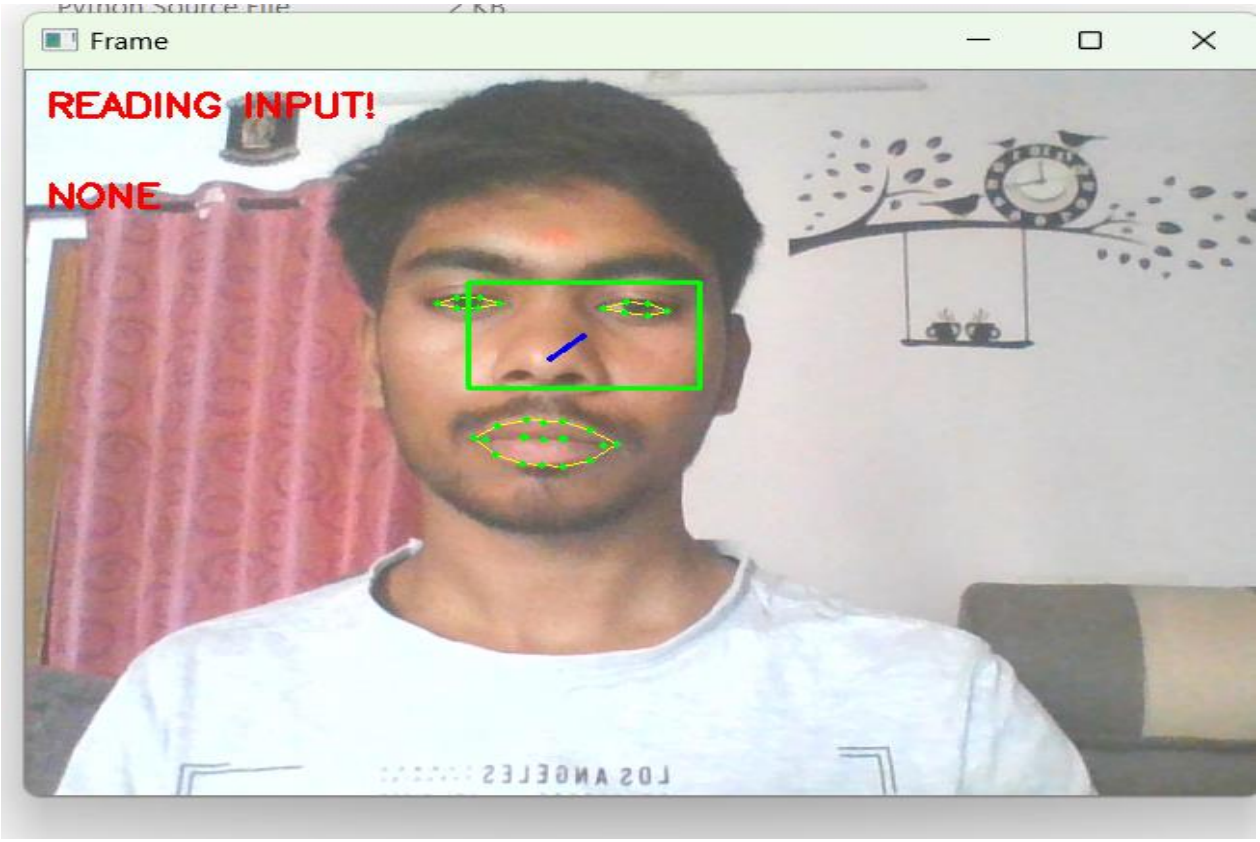

Fig 8.1.5 Output - 5

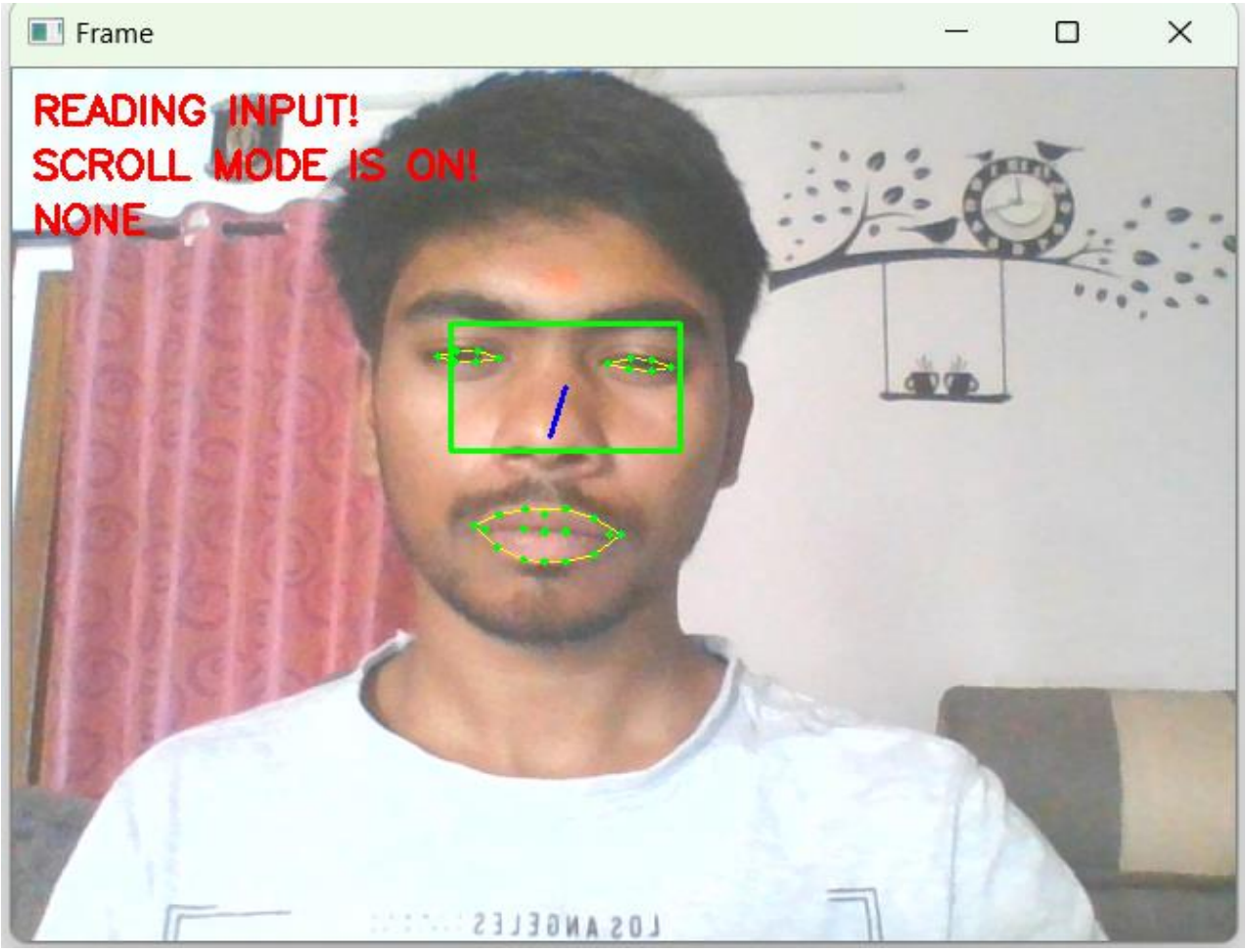

Fig 8.1.6 Output - 6

CHAPTER 9

CONCLUSION

The project has achieved its primary goal by creating a system that allows people with disabilities to interact with computers effectively using their eye movements.This successful implementation means that individuals who may have limited or no use of their hands can still use computers independently. The project's use of advanced algorithms for tasks like has significantly improved how accessible computers are for individuals with disabilities.These algorithms ensure that the system can accurately and reliably track a user's eye movements, which is crucial for effective cursor control. By providing a means for disabled individuals to interact with computers more easily and independently, the project has the potential to greatly improve their overall quality of life. Being able to engage with the digital world effectively opens up opportunities for communication, education, work, and entertainment that might otherwise be limited. In conclusion, this project is a significant step forward in making technology more inclusive and accessible for all individuals, regardless of their physical abilities. It aligns with the broader societal goal of creating an Information Society where everyone can participate fully and equally in the digital age, bridging the gap between technology and disability.

CHAPTER 10

FUTURE WORK

Here are some potential future enhancements for the project focused on cursor movement and actions using eyeball movements:

- Improved Accuracy and Precision: Enhance the accuracy and precision of eyeball tracking algorithms to ensure more reliable cursor movement and interaction. This could involve refining calibration procedures, optimizing tracking algorithms, and incorporating advanced machine learning techniques for eye tracking.

- Expanded Gesture Recognition: Introduce support for a wider range of eye gestures and movements to enable more diverse interactions with the cursor. This could include gestures such as scrolling, zooming, right-clicking, and dragging, allowing users to perform a broader range of actions using only their eye movements.

- Customization and Personalization: Provide options for users to customize and personalize their eye gaze interaction experience. This could involve allowing users to adjust sensitivity settings, customize gesture mappings, and define their own eye movement patterns for specific actions.

- Integration with Assistive Technologies: Integrate the eye gaze interaction system with existing assistive technologies to provide enhanced accessibility features for users with disabilities. This could include compatibility with screen readers, voice control systems, and alternative input devices to accommodate diverse user needs.

- Multi-Modal Interaction: Explore the integration of eye gaze interaction with other input modalities, such as voice commands, hand gestures, or brain-computer interfaces. This multi-modal approach could offer users more flexibility and control over their interactions with digital devices and applications.

CHAPTER 11

REFERENCES

[1] Neil Castellino and Michelle Alva, “An image-based eye controlled assistive system for paralytic patients”, IEEE Conference Publications, 2017.

[2] Shu-Fan Lin, Xuebai Zhang, Shyan-Ming Yuan, "Eye Tracking Based Control System for Natural Human-Computer Interaction", Computational Intelligence and Neuroscience, 2017.

[3] Čech and Soukupová, “Real-Time Eye Blink Detection using Facial Landmarks”, Center for Machine Perception, February 2016.

[4] Suree Pumrin and Chairat kraichan, “Face and eye tracking for controlling computer functions”, IEEE Conference Publications, 2014.

[5] A. Geetha and M. Mangaiyarkarasi, “Cursor Control System Using Facial Expressions for Human-Computer Interaction”, Vol 8 Issue 1 APRIL 2014, ISSN: 0976-1353 International Journal of Emerging Technology in Computer Science & Electronics, pp 30-34.

[6] Florina Ungureanu, Robert Gabriel Lupu and Valentin Siriteanu, “Eye tracking mouse for human-computer interaction”, IEEE Conference Publications, 2013.

[7] Jilin Tu and Thomas Huang, *Face as Mouse through Visual Face Tracking*, 2005.

[8] EniChul Lee Kang Ryoung Park, A robust eye gaze tracking method based on a virtual eyeball model, Springer, pp. 319-337, Apr 2008.

[9] John J. Magee, Margrit Betke, James Gips, Matthew R. Scott and Benjamin N. Waber, "A Human-Computer Interface Using Symmetry Between Eyes to Detect Gaze Direction", *IEEE Trans*, vol. 38, no. 6, pp. 1248-1259, Nov 2008.

[10] Sunita Barve, Dhaval Dholakiya, Shashank Gupta and Dhananjay Dhatrak, "Facial Feature Based Method For Real Time Face Detection and Tracking I-

CURSOR", *International Journal of Engg Research and App.*, vol. 2, pp. 1406-1410, Apr 2012.

[11] Yu-Tzu Lin, Ruei-Yan Lin, Yu-Chih Lin and Greg C Lee, Real-time eye-gaze estimation using a low-resolution webcam, Springer, pp. 543-568, Aug 2012.

[12] Samuel Epstein-Eric MissimerMargritBetke, "Using Kernels for avideo-based mouse-replacement interface", *Springer link*, Nov 2012.

[13] Zakir Hossain, Md Maruf Hossain Shuvo and Prionjit Sarker, "Hardware and software implementation of real time electrooculogram (EOG) acquisition system to control computer cursor with eyeball movement", *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*, pp. 132-137, 2017.

[14] Jun-Seok Lee, Kyung-hwa Yu, Sang-won Leigh, Jin-Yong Chung and Sung-Goo Cho, *Method for controlling device on the basis of eyeball motion and device therefor*, January 2018.

[15] Po-Lei Lee, Jyun-Jie Sie, Yu-Ju Liu, Chi-Hsun Wu, Ming-Huan Lee, Chih-Hung Shu, et al., "An SSVEP-actuated brain computer interface using phase-tagged flickering sequences: a cursor system", *Annals of biomedical engineering*, vol. 38, no. 7, pp. 2383-2397, 2010.