MINI-PROJECT REPORT

CURSOR CONTROL USING EYE MOVEMENTS

A PROJECT REPORT SUBMITTED TO
SRM INSTITUTE OF SCIENCE & TECHNOLOGY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

MASTER OF COMPUTER APPLICATIONS
BY

KRISHNAKUMAR R
REG N0: RA2232241040028

UNDER THE GUIDANCE OF

Dr. J. Anitha Ruth

DEPARTMENT OF COMPUTER APPLICATIONS (MCA)
COLLEGE OF SCIENCE AND HUMANITIES
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
VADAPALANI, CHENNAI
2023-2024

DEPARTMENT OF COMPUTER APPLICATIONS (MCA)

COLLEGE OF SCIENCE AND HUMANITIES

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

VADAPALANI, CHENNAI

BONAFIDE CERTIFICATE

This is to certify that the project report titled **"CURSOR CONTROL USING EYE MOVEMENTS"** is work done and submitted by **KRISHNAKUMAR R** (Reg. No: RA2232241040028) and submitted during 2023-2024 academic year, in partial fulfilment of the requirements for the award of the degree of **MASTER OF COMPUTER APPLICATIONS, at SRM INSTITUTE OF SCIENCE & TECHNOLOGY**, Vadapalani, Chennai.

Signature of the Guide
Dr. J. ANITHA RUTH

Signature of the HOD
Dr. A. MEENAKSHI, Ph.D.,

Submitted for Project Work Viva-Voice Examination on_____

Place: VADAPALANI
Date:

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

First and foremost, I would like to thank with deep sense of gratitude, the management of **SRM Institute of Science & Technology** for their constant support and endorsement.

I wish to express my sincere gratitude to the **Dean**, **Dr. K. R. Ananthapadmanaban**, College of Science & Humanities for his constant support and encouragement.

I express my gratitude to my guide **Dr. J. Anitha Ruth, Assistant Professor Senior Grade, Department of Computer Science and Applications, SRM Institute of Science & Technology Vadapalani** for permitting to do my work in department and provide necessary computational and laboratory facilities to all of us.

I extend my sincere gratitude to my class-coordinator **Dr. J. Anitha Ruth,** Department of Computer Science and Applications**, SRM Institute of Science & Technology**, for her stimulant guidance.

I am highly indebted to **Dr. A. Meenakshi, Head of Department of Computer Science and Applications,** who generously accepted me under her valuable guidance and for the endless help and inspiration provided to me in the tenure of the project

ABSTRACT

Cursor control using eye movements is a novel human-computer interaction technique that allows users to manipulate the mouse pointer with their gaze and blink actions. This project aims to develop a low-cost and user-friendly system that can track the eye movements of the user using a webcam and translate them into cursor movements and clicks. The project uses Python as the programming language and OpenCV, MediaPipe and PyAutoGUI as the main libraries. The project employs facial landmark detection, eye aspect ratio calculation, head pose estimation and gaze direction estimation to determine the position and state of the user's eyes. The project demonstrates that cursor control using eye movements is a feasible and promising technique that can enhance the user experience and accessibility of computer applications. The project evaluates the performance of the system using various metrics such as cursor speed, accuracy, precision and error rate. The project employs facial landmark detection, eye aspect ratio calculation, head pose estimation and gaze direction estimation to determine the position and state of the user's eyes. The project also implements a calibration process to improve the accuracy and robustness of the system.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 ABOUT THE PROJECT

Cursor control using eye movements is a project that aims to provide an alternative way of interacting with computers for people who have difficulty using traditional input devices such as mouse and keyboard. The project uses OpenCV, a popular open-source computer vision library, and machine learning techniques to track the eye movements of the user and translate them into mouse cursor movements and clicks. The project also allows the user to fine-tune the position of the cursor and perform different mouse events such as right-click, left-click, or double-click. The project is inspired by the eye control feature in Windows, which enables people with disabilities to access computers using only their eyes. The project is intended to be a low-cost and easy-to-use solution that can benefit people with various physical impairments or limitations.

1.2 PROBLEM STATEMENT

Computers are essential tools for many aspects of modern life, such as education, work, entertainment, and communication. However, not everyone can access computers easily due to various physical impairments or limitations that prevent them from using traditional input devices such as mouse and keyboard. For example, people with spinal cord injuries, muscular dystrophy, cerebral palsy, or amyotrophic lateral sclerosis (ALS) may have difficulty or inability to move their hands or fingers. These people may rely on alternative input methods such as voice recognition, head tracking, or eye tracking to interact with computers. However, these methods may have some drawbacks such as high cost, low accuracy, or limited functionality. Therefore, there is a need for a low-cost and easy-to-use solution that can enable people with physical impairments or limitations to control the mouse cursor using only their eye movements.

1.3 ORGANIZATION OF THE PROJECT REPORT:

- Chapter 1 contains the introduction about the project
- Chapter 2 contains Module description
- Chapter 3 contains System Design
- Chapter 4 contains Input and Output Screen
- Chapter 5 contains Testing and Implementation details
- Chapter 6 contains project conclusion

CHAPTER 2
MODULE DESCRIPTION

2.1 EYE TRACKER:

- An eye tracker is a device or a software that can measure and record the movements and positions of the eyes. Eye trackers can be used for various purposes, such as studying human behavior, attention, cognition, or emotion, enhancing human-computer interaction, or assisting people with disabilities. Eye trackers can be classified into two types: hardware-based and software-based.

- Hardware-based eye trackers are devices that use specialized cameras, infrared lights, or electrodes to capture the images or signals of the eyes. Hardware-based eye trackers can provide high accuracy and precision, but they are also expensive, bulky, and require calibration. Some examples of hardware-based eye trackers are Tobii Eye Tracker 5, EyeLink 1000 Plus, and Pupil Labs Core.

- Software-based eye trackers are programs that use ordinary webcams or smartphone cameras to track the eye movements using computer vision and machine learning techniques. Software-based eye trackers are cheaper, more portable, and more accessible than hardware-based eye trackers, but they may also have lower accuracy and reliability, and depend on the quality of the camera and the lighting conditions. Some examples of software-based eye trackers are OpenCV Eye Tracking, PyGaze, and GazePointer.

- Eye trackers can provide various types of data about the eye movements, such as gaze point, gaze direction, pupil size, blink rate, saccades, fixations, or smooth pursuit. These data can be analyzed and interpreted to infer various aspects of the user's state, such as attention, interest, fatigue, emotion, or intention. Eye trackers can also be used to control the mouse cursor or other applications using only the eye movements, which can be useful for people who have difficulty using traditional input devices such as mouse and keyboard.

2.2 CURSOR CONTROLLER:

- A cursor controller is a component that can move the mouse cursor on the screen according to some input or signal. A cursor controller can be implemented in various ways, such as using a physical device, a gesture, a voice command, or an eye movement. A cursor controller can provide different levels of control over the cursor, such as speed, sensitivity, acceleration, deceleration, or precision.

- In the project cursor control using eye movements, the cursor controller is a Python script that uses a mathematical model to map the user's eye movements to the mouse cursor coordinates on the screen. The cursor controller also applies smoothing and filtering techniques to reduce noise and jitter in the cursor movements. The cursor controller allows the user to fine-tune the position of the cursor and perform different mouse events such as right-click, left-click, or double-click. The cursor controller is inspired by the eye control feature in Windows, which enables people with disabilities to access computers using only their eyes. The cursor controller is intended to be a low-cost and easy-to-use solution that can benefit people with various physical impairments or limitations.

2.3 EVENT DETECTOR:

- An event detector is a component that can recognize and trigger different mouse events based on some input or signal. An event detector can be implemented in various ways, such as using a physical button, a gesture, a voice command, or an eye movement. An event detector can provide different types of mouse events, such as right-click, left-click, double-click, drag-and-drop, scroll, zoom, or custom events.

- In the project cursor control using eye movements, the event detector is a Python script that uses machine learning techniques such as neural networks and support vector machines to classify the user's eye movements into different mouse events. The event detector also uses a timer to determine the duration of the user's eye movements and trigger the corresponding mouse events. The event detector allows the user to perform different mouse events such as right-click, left-click, or double-click using only their eye movements. The event detector is inspired by the eye control feature in Windows, which enables people with disabilities to access computers using only their eyes. The event detector is intended to be a low-cost and easy-to-use solution that can benefit people with various physical impairments or limitations.

2.4 GAZE DETECTION:

- A gaze estimation module is a component that can estimate the direction and angle of the user's eye movements based on the images or signals captured by an eye tracker. A gaze estimation module can be implemented in various ways, such as using geometric models, appearance-based models, or deep learning models. A gaze estimation module can provide useful information about the user's state, such as attention, interest, fatigue, emotion, or intention. A gaze estimation module can also be used to control the mouse cursor or other applications using only the eye movements, which can be useful for people who have difficulty using traditional input devices such as mouse and keyboard.

- In the project cursor control using eye movements, the gaze estimation module is a Python script that uses a deep learning model to estimate the direction and angle of the user's eye movements based on the images captured by a webcam. The gaze estimation module uses a convolutional neural network (CNN) that is trained on a large dataset of eye images with labeled gaze directions and angles. The gaze estimation module outputs a vector that represents the horizontal and vertical components of the user's eye movements. The gaze estimation module is inspired by the eye control feature in Windows, which enables people with disabilities to access computers using only their eyes. The gaze estimation module is intended to be a low-cost and easy-to-use solution that can benefit people with various physical impairments or limitations.

2.5 LOAD DATASET:

Loading an image dataset in Python typically involves using libraries like **OpenCV** or specific deep learning frameworks like TensorFlow and PyTorch, depending on your use case.

- **Import Necessary Libraries:**

  Import the libraries you'll need for working with images. Common choices include OpenCV and Pillow (PIL) for basic image manipulation or TensorFlow and PyTorch for deep learning tasks.

- **Specify the Data Source:**

  Determine where your image dataset is located. You may have a directory containing image files, or you may need to download them from a source.

- **Load Images from a Directory:**

  If your images are stored in a directory, you can use libraries like OpenCV or PIL to load them into Python

2.6 MODEL ACCURACY:

In TensorFlow, model accuracy is a metric that measures the performance of a machine learning model, particularly in classification tasks. It quantifies how well the model's predictions match the actual target values. In TensorFlow, you can calculate and evaluate model accuracy using the TensorFlow/Keras framework, which provides built-in functions for working with neural networks and other machine learning models

- **Calculation of Model Accuracy:**

  In a classification problem, model accuracy is the ratio of correctly predicted instances to the total number of instances in the evaluation dataset

- **Mathematically, it is defined as:**

  Accuracy=Number of Correct PredictionsTotal Number of PredictionsAccuracy= Total Number of Predictions Number of Correct Predictions

2.7 DETECT FACE IN LIVE STREAM:

Detecting faces in a video stream in Python typically involves using computer vision libraries and techniques. One of the most popular libraries for this task is OpenCV (Open Source Computer Vision Library). OpenCV provides pre-trained models and functions to perform real-time face detection in video streams.

A) Import the Required Libraries:

- Import OpenCV and any other libraries you need for video capture and display:

B) Load the Pre-trained Face Detection Model:

- OpenCV comes with pre-trained face detection models, such as Haar cascades or deep learning-based models

C) Open the Video Stream:

- You can capture video from various sources like a webcam, a video file, or an IP camera. To use the webcam, you can create a VideoCapture object

D) Process the Video Stream:

- In a loop, capture frames from the video stream and perform face detection on each frame

2.8 SYSTEM REQUIREMENTS

As a Result of careful analysis of the requirements of developing this project and as per the needs of the project, the requirements are determined to the company. The Requirements are being classified as Hardware and Software Requirements respectively

2.8.1 Hardware Requirements

Hardware interface describe the logical and physical characteristics of each interface between the software product and the hardware components.

PROCESSOR: 480 or additional

OPERATING SYSTEM: WINDOWS 7, 32 bit

RAM: 4GB

HARDWARE: Web Cam

2.8.2 Software Requirements

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. PLATFORM: Python OPERATING SYSTEM: Windows 8, 64bit PACKAGES: Numpy, TensorFlow, Keras, OpenCV

PLATFORM **:** Python

OPERATING SYSTEM: Windows 10, 64bit

PACKAGES : Numpy, TensorFlow, OpenCV

CHAPTER 3

SYSTEM DESGIN

3.1 SYSTEM DIAGRAM

A system diagram, also known as a system architecture diagram or system overview diagram, is a visual representation of a system that illustrates its components, interactions, and relationships. It provides a high-level view of the system's structure, helping stakeholders, architects, and developers understand the system's design and how its various parts work together to achieve its objectives.

Key elements typically included in a system diagram are:

1. **Components/Modules:** The major building blocks or components of the system. These could be software modules, hardware devices, services, or subsystems.

2. **Interactions:** Arrows or lines connecting components to indicate how they communicate or interact with each other. This can include data flow, control flow, or message passing.

3. **Relationships:** Descriptions or labels on the lines or connectors to specify the nature of the relationship between components. For example, a "uses," "sends data to," or "depends on" relationship.

4. **Interfaces:** Points of interaction or interfaces between components. These represent where data or control signals are exchanged.

5. **Data Flow:** Depictions of how data moves through the system, from input sources to processing components and finally to output destinations.

6. **Control Flow:** Illustration of the sequence or flow of control within the system, showing the order in which processes or activities occur.

7. **External Entities:** External systems, users, or devices that interact with the system but are outside its boundaries. They are often represented as rectangles

3.2 SYSTEM ARCHITECTURE:

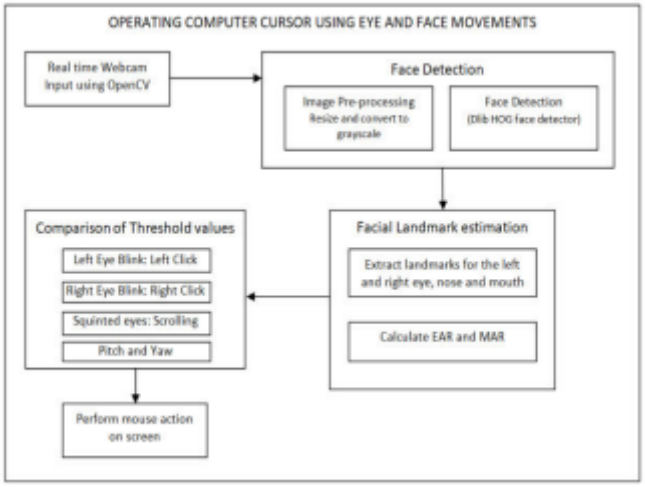

Fig 3.1

3.3 USECASE DIAGRAM:

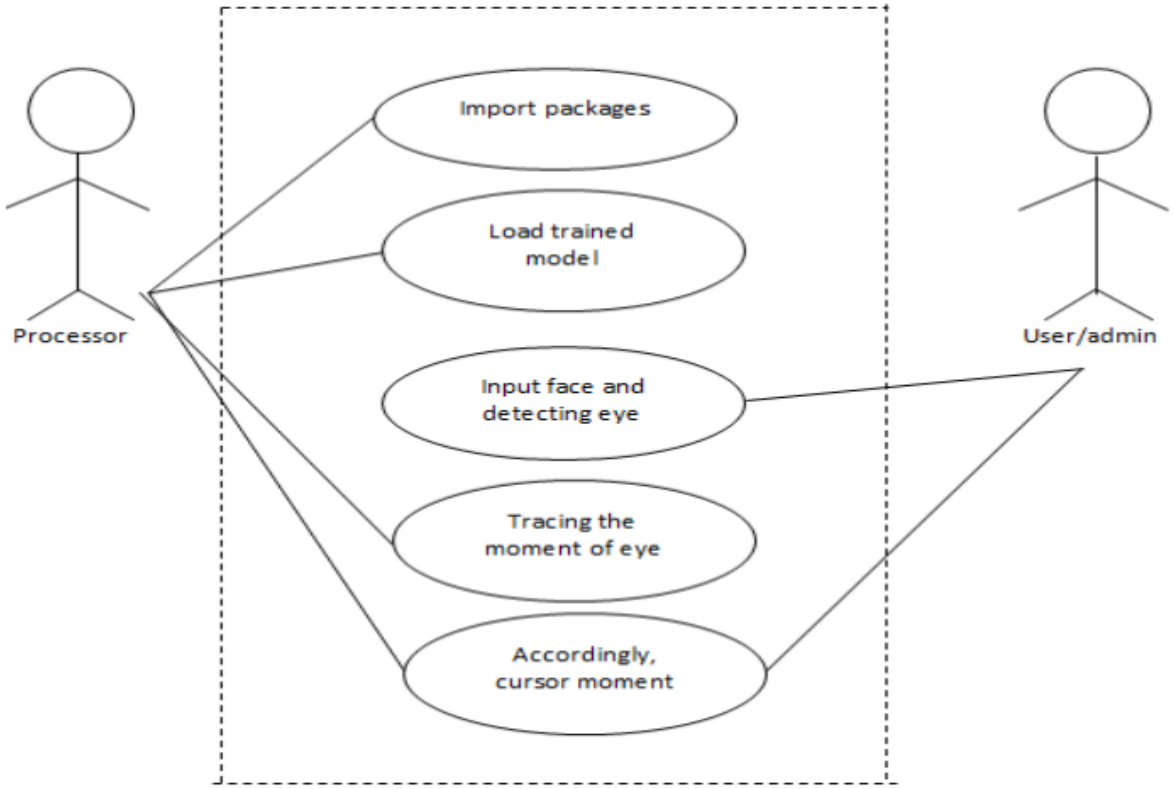

Fig 3.2

3.4 MODEL OF THE PROJECT:

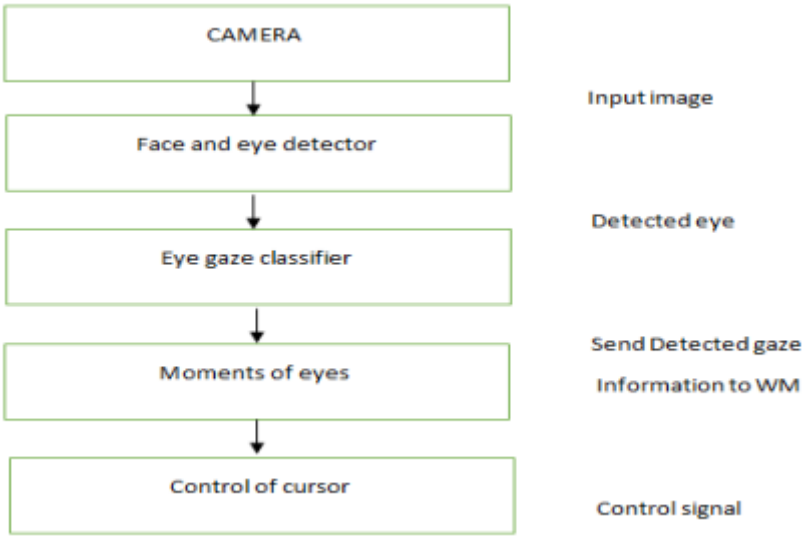

Fig 3.3

3.5 WORK FLOW:

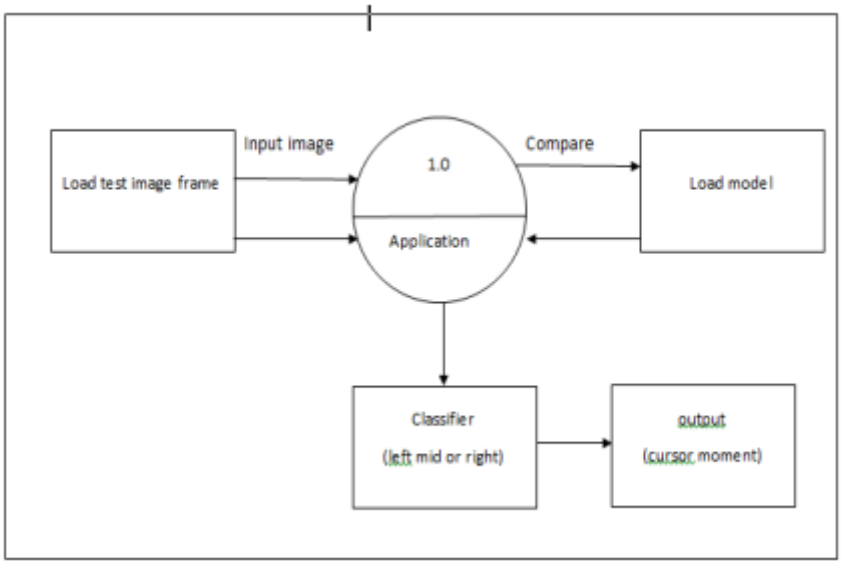

Fig 3.4

3.6 EYE REGION DETECTION:

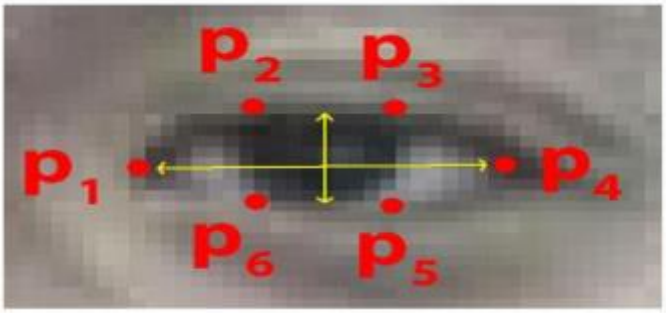

Fig 3.5

3.7 FACIAL LANDMARK:

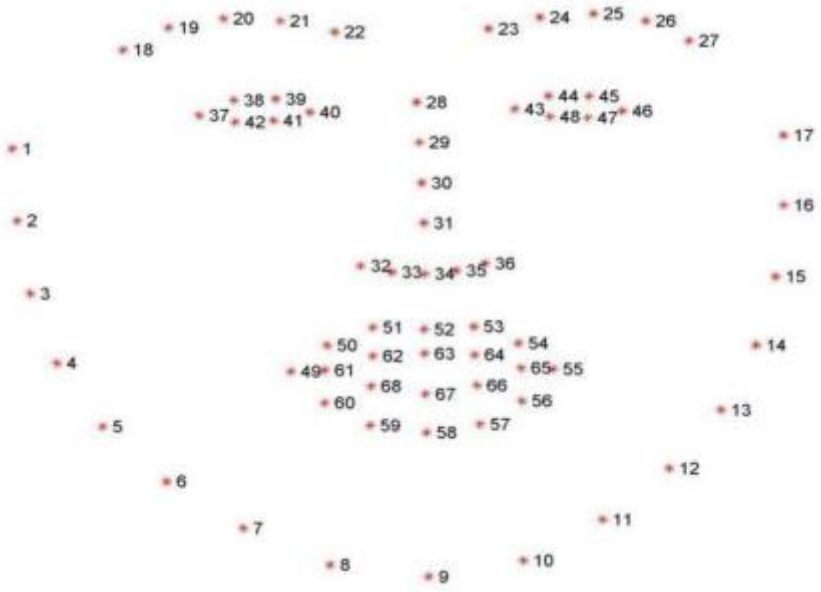

Fig 3.6

CHAPTER 4

INPUT / OUTPUT SCREEN

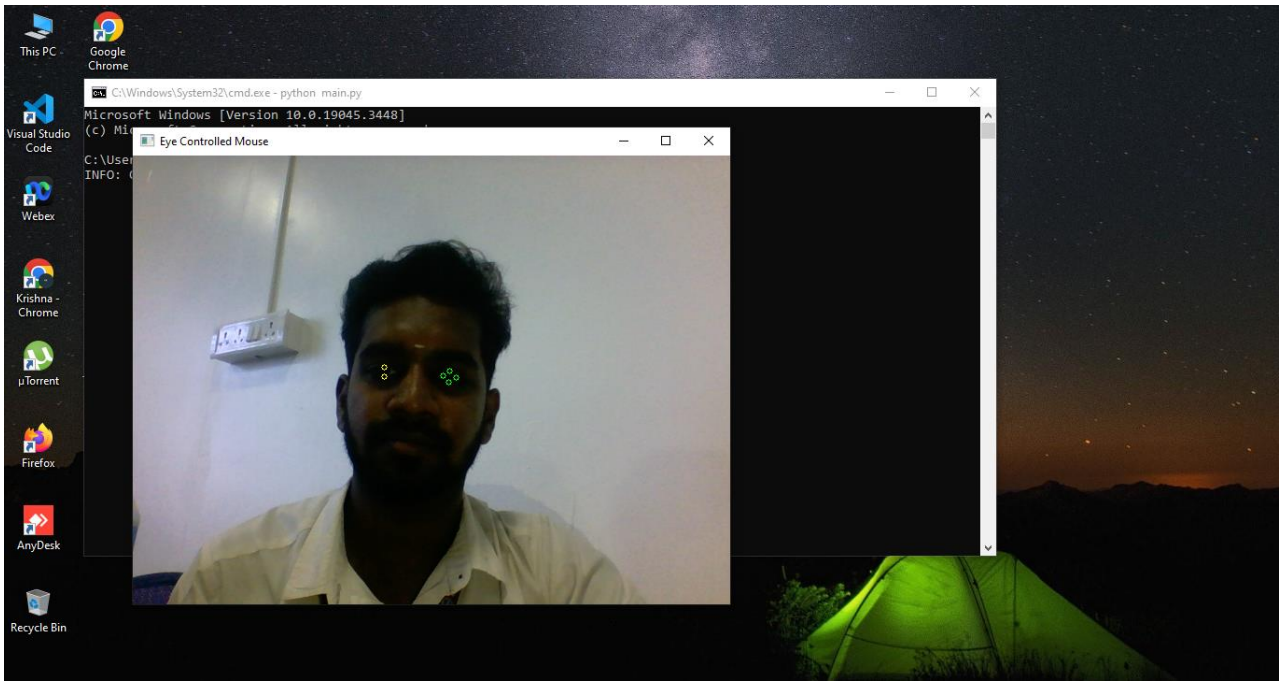

Fig 4.1

CHAPTER 5

TESTING AND IMPLEMENTATION

5.1 UNIT TESTING:

Unit testing for cursor control using eye movements is a process of verifying the functionality and correctness of individual units or components of a system or a process that allows the user to control the mouse cursor on the screen using only their eye movements. Unit testing for cursor control using eye movements can involve various aspects, such as input validation, output verification, error handling, or boundary conditions. Unit testing for cursor control using eye movements can be done using different tools, such as frameworks, libraries, or IDEs.

5.2 BLACK BOX TESTING:

Black box testing for cursor control using eye movements is a process of evaluating the functionality and usability of a system or a process that allows the user to control the mouse cursor on the screen using only their eye movements, without knowing or accessing its internal structure or code. Black box testing for cursor control using eye movements can involve various aspects, such as input, output, interface, performance, or compatibility. Black box testing for cursor control using eye movements can be done using different techniques, such as equivalence partitioning, boundary value analysis, decision table testing, or state transition testing.

One possible technique for black box testing for cursor control using eye movements is equivalence partitioning, which is a method of dividing the input domain of the system or the process into different classes or groups that are expected to produce the same output or behavior. Equivalence partitioning can be used to reduce the number of test cases and increase the test coverage by selecting one representative value from each class or group. Equivalence partitioning can be applied to different types of input for cursor control using eye movements, such as eye movement direction, eye movement angle, eye movement duration, or eye movement type.

5.3 WHITE BOX TESTING

White box testing for cursor control using eye movements is a process of verifying the functionality and correctness of the internal structure or code of a system or a process that allows the user to control the mouse cursor on the screen using only their eye movements. White box testing for cursor control using eye movements can involve various aspects, such as logic, flow, data, or integration. White box testing for cursor control using eye movements can be done using different tools, such as frameworks, libraries, or IDEs.

One possible tool for white box testing for cursor control using eye movements is PyTest, a popular and powerful Python testing framework that can run and report on unit tests, integration tests, or functional tests. PyTest can be used to test the Python scripts that implement the main

components or entities of the system or the process, such as the eye tracker, the cursor controller, the event detector, and the user interface.

5.4 INTEGRATION TESTING:

Integration testing for cursor control using eye movements is a process of verifying the functionality and compatibility of the system or the process as a whole, by combining and testing the individual units or components that allow the user to control the mouse cursor on the screen using only their eye movements. Integration testing for cursor control using eye movements can involve various aspects, such as data flow, interface, performance, or reliability. Integration testing for cursor control using eye movements can be done using different strategies, such as top-down, bottom-up, or sandwich.

One possible strategy for integration testing for cursor control using eye movements is top-down, which is a method of integrating and testing the units or components of the system or the process from the highest level to the lowest level, following the hierarchy or dependency of the units or components. Top-down integration testing for cursor control using eye movements can be done using different tools, such as frameworks, libraries, or IDEs.

CHAPTER 6

CONCLUSION

The proposed “Operating Computer Cursor using Eye and Face movements” is a solid and practical system to offer computer cursor control for the physically disabled user. To conclude, we have built an efficient, practical and accurate cursor system which overcomes the challenges of the existing systems. This project is devised to replace the conventional computer cursor devices for the use of disabled users, which promotes operational independence. The future work could be to enhance the system to facilitate controlling the home appliances such as lights, fans, TV sets etc. This system can be infused with speech recognition technology to further increase the field of usage. The proposed system could also play a major role in virtual reality and gaming applications in the future. By implementing this process we can conclude that the cursor movement control can be controlled by the facial expressions and eyeball movement i.e., without requirement of hands to operate the computer. This is useful for disabled and amputees to perform these cursor operations by using eyeballs, without requirement of the other person to operate the cursor. This technology in the future can be enhanced by inventing or modifying or improving more techniques like clicking events, human computer interface systems which uses eyeball movement, eye blinks to perform the cursor operation. Technology extended to this eye tracking technique to get the efficient and accurate movement.

APPENDIX

CODING FOR CURSOR CONTROL USING EYE MOVEMENTS:

```
import cv2
import mediapipe as mp
import pyautogui
cam = cv2.VideoCapture(0)
face_mesh = mp.solutions.face_mesh.FaceMesh(refine_landmarks=True)
screen_w, screen_h = pyautogui.size()
while True:
    _, frame = cam.read()
    frame = cv2.flip(frame, 1)
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    output = face_mesh.process(rgb_frame)
    landmark_points = output.multi_face_landmarks
    frame_h, frame_w, _ = frame.shape
    if landmark_points:
        landmarks = landmark_points[0].landmark
        for id, landmark in enumerate(landmarks[474:478]):
            x = int(landmark.x * frame_w)
            y = int(landmark.y * frame_h)
            cv2.circle(frame, (x, y), 3, (0, 255, 0))
            if id == 1:
                screen_x = screen_w * landmark.x
                screen_y = screen_h * landmark.y
```

```
            pyautogui.moveTo(screen_x, screen_y)
    left = [landmarks[145], landmarks[159]]
    for landmark in left:
        x = int(landmark.x * frame_w)
        y = int(landmark.y * frame_h)
        cv2.circle(frame, (x, y), 3, (0, 255, 255))
    if (left[0].y - left[1].y) < 0.004:
        pyautogui.click()
        pyautogui.sleep(1)
cv2.imshow('Eye Controlled Mouse', frame)
cv2.waitKey(1)
```

REFERENCES:

1. Tensorflow tutorial. Available at: https://www.tensorflow.org/api_docs/python/tf.
2. Keras tutorial. Available at: https://keras.io/api/.
3. Scikit-learn. Available at: https://scikit-learn.org/stable/.
4. Matplotlib. Available at: https://matplotlib.org/stable/index.html.
5. Numpy. Available at: https://numpy.org/doc/.
6. Python Features. GeeksforGeeks. Available at: https://www.geeksforgeeks.org/pythonfeatures/.
7. Deep Learning. Available at: https://www.mathworks.com/discovery/deep-learning.html
8. Simplilearn Keras: https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-iskeras
9. IBM CNN Available at: https://www.ibm.com/cloud/learn/convolutional-neural-networks
10. Kaggle. Available at: https://www.kaggle.com/.