MINI PROJECT REPORT

ON

EYE TRACKING MOUSE

SUBMITTED IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS OF

DEGREE OF

BACHELOR OF ENGINEERING

BY

ANMOL J. MESHRAM

YASH V. KOYANDE

OMKAR R. DHANAWADE

VAIBHAV G. BHALE

SUPERVISOR

Prof. ASHWINI VARMA

DEPARTMENT OF INFORMATION TECHNOLOGY

PILLAI HOC COLLEGE OF ENGINEERING AND TECHNOLOGY,

PILLAI'S HOCL EDUCATIONAL CAMPUS, HOCL COLONY,

RASAYANI, TAL: KHALAPUR, DIST RAIGAD 410207

UNIVERSITY OF MUMBAI

[2024-25]

Mahatma Education Society's

Pillai HOC College of Engineering and Technology, Rasayani-410207

[2024-25]

Certificate

This is to certify that the Mini Project -1 B entitled Project Title is a bonafide work of Anmol J. Meshram, Yash V. Koyande, Vaibhav G. Bhale, Omkar R. Dhanawade submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the degree of "**Undergraduate**" in "**Information Technology**".

Prof. Ashwini Varma	Prof. Trapti Soni
(Supervisor)	(Project Coordinator)
Prof. Srijita Bhattacharjee	Dr. J. W. Bakal
(Head of Department)	(Principal)

Mini Project-1B Report Approval

This project report entitled **EYE TRACKING MOUSE** submitted by Anmol J. Meshram, Yash V. Koyande, Vaibhav G. Bhale, Omkar R. Dhanawade is approved for the degree of **Bachelor of Engineering in Information Technology**.

Examiners

1. _____

2. _____

Date:

Place:

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included. We have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will because for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

Anmol J. Meshram

_____

Yash V. Koyande

_____

Omkar Ratnakar Dhanawade

_____

Vaibhav G. Bhale

Date:

Abstract

The Eye-Tracking Mouse Control system is a hands-free alternative to traditional computer navigation, aimed at improving accessibility for individuals with limited mobility. Developed using Python, OpenCV, MediaPipe, and PyAutoGUI, the system leverages computer vision to track real-time eye movements and translate them into cursor actions using a standard webcam. It also detects eye blinks to simulate mouse clicks, offering an intuitive and seamless user experience. By eliminating the need for external hardware, it remains both cost-effective and easy to implement. Future enhancements may include AI-based gaze prediction and advanced calibration to further improve accuracy and responsiveness.

Keywords: Eye-Tracking, Accessibility, Cost-effective, Python, PyAutoGUI, OpenCV, MediaPipe, Real-time eye movements.

List of Figures

Figure No.	Figure Name	Page No.
Figure 4.1	System Architecture	10
Figure 5.1	Right Click	14
Figure 5.2	Left Click	14

TABLE OF CONTENTS

	Abstract	**I**
	List of Figures	**II**
1.	**Introduction**	**1**
	1.1 Introduction	2
	1.1 Background	2
	1.2 Motivation	2
2.	**Literature Survey**	**3**
	2.1 Basic Terminologies	4
	2.2 Existing System	5
	2.3 Problem Statement	5
3.	**Requirement Gathering**	**6**
	3.1 Software and Hardware Requirements	7
4.	**Plan of Project**	**9**
	4.1 Proposed System Architecture	10
	4.2 Methodology	11
5.	**Result Analysis**	**13**
	5.1 Results and discussion	14
6.	**Conclusion**	**15**
	References	**17**

Chapter 1
Introduction

1.1 Introduction

The Eye-Tracking Mouse Control system is an innovative hands-free alternative to traditional computer navigation, designed to enhance accessibility for individuals with limited mobility. This project utilizes computer vision and machine learning techniques to track eye movements and translate them into cursor movements on the screen. Built using Python, OpenCV, MediaPipe, and PyAutoGUI, the system captures real-time facial landmarks to determine gaze direction, allowing precise cursor control. Additionally, it detects eye blinks to simulate mouse clicks, enabling a seamless interaction experience. This technology addresses the challenges posed by conventional input devices, making computing more inclusive and intuitive. The system operates using a standard webcam, requiring no additional hardware, making it cost-effective and easy to implement.

1.2 Background

Traditional computer input devices such as a mouse and keyboard require fine motor skills and hand coordination, making them inaccessible for individuals with disabilities or mobility impairments. Over the years, researchers have explored alternative input methods, such as voice control and gesture recognition, but these often require expensive hardware or are not precise enough for daily use Eye-tracking technology provides a hands-free solution, allowing users to navigate their computers using gaze movements.

1.3 Motivation

The primary motivation behind this project is to enhance accessibility for individuals who struggle with conventional input devices due to physical disabilities, injuries, or medical conditions such as paralysis, ALS, or muscular dystrophy. Many assistive technologies available today are either too expensive, require additional hardware, or lack precision, making them impractical for daily use. This project aims to create a cost-effective, software-based alternative that allows users to interact with computers using only their eyes. By leveraging computer vision algorithms, we can offer a seamless and intuitive experience without requiring any additional hardware beyond a standard webcam.

Chapter 2
Literature Survey

2.1 Basic Terminologies

- **Eye Tracking:**

Technique used to measure eye position and movement. Helps determine where a user is looking on the screen.

- **Gaze Estimation:**

Computational method to identify the exact point of focus. Uses eye landmarks to map gaze direction to screen coordinates.

- **Blink Detection:**

Identifies voluntary eye closures to simulate click actions. Relies on EAR (Eye Aspect Ratio) to distinguish between open and closed eyes.

- **Eye Aspect Ratio (EAR):**

A mathematical ratio of vertical vs. horizontal eye landmark distances. Lower EAR values indicate eye closure; used for blink-based input.

- **Facial Landmarks:**

Key reference points on the face (e.g., corners of eyes, lips, nose). Used to track facial movements and expressions.

- **MediaPipe Face Mesh:**

Machine learning framework by Google. Detects 468 detailed 3D facial landmarks in real-time.

- **OpenCV (Open Source Computer Vision Library):**

Captures and processes video frames from the webcam. Provides real-time image processing support for the application.

- **PyAutoGUI:**

A Python library for simulating mouse and keyboard actions. Allows the system to move the cursor and click based on gaze and blink detection.

- **Cursor Mapping:**

Translates eye gaze coordinates from camera space to screen coordinates. Ensures accurate cursor placement based on where the user is looking.

- **Real-time Processing:**

Ensures all operations (video capture, facial tracking, actions) occur instantly. Makes the system smooth and responsive for real-world interaction.

2.2 Existing System

High-end eye-tracking systems rely on infrared-illuminated cameras, specialized sensors, and dedicated processing units to achieve millisecond-level accuracy and robust tracking under varying lighting conditions. Screen-based trackers can cost between $1,000 and over $10,000, while wearable eye-tracking glasses from leading vendors often exceed $30,000 once hardware, software, and support are bundled. These solutions offer binocular tracking, multi-point calibration, and sampling rates from 60 to over 250 Hz, as well as integrated scene cameras and comprehensive SDKs for research and commercial use.

2.3 Problem Statement

Traditional input devices are often inaccessible to individuals with severe physical disabilities, limiting their ability to interact with computers. Existing eye-tracking systems tend to be expensive and rely on specialized hardware, making them less accessible. Many hands-free alternatives lack the precision required for accurate cursor control, and blink-based click detection is frequently unreliable. Additionally, most accessible technologies are not user-friendly or affordable for everyday use.

Chapter 3
Requirement Gathering

3.1 Software and Hardware Requirements

3.1.1. Software Requirements

1.Python 3.x

- Version: Python 3.7 or later is recommended for compatibility with MediaPipe, and PyAutoGUI.

2. Operating System

- Windows 10/11
- macOS Monterey or later
- Ubuntu 20.04 LTS or later

3. IDE (Integrated Development Environment)

- PyCharm
- Visual Studio Code (VS Code)
- Jupyter Notebook (for prototyping)

3.1.2. Hardware Requirements

1. Computer System

- Processor: Intel Core i5 (8th Gen or higher) / AMD Ryzen 5

- RAM: Minimum 4GB (Recommended: 8GB or more)

- Storage: At least 5MB of free disk space for software installations and logs

- Graphics Card: Integrated GPU is sufficient, but a dedicated GPU (NVIDIA/AMD) can improve performance.

2. Webcam

- Resolution: Minimum 720p HD (1280×720 pixels) for better accuracy

- Frame Rate: 30 FPS or higher for smooth tracking

- Camera Type: Built-in webcam or USB external webcam

3. Backup Storage (Optional)

- Options: External HDD/SSD or Cloud Storage (Google Drive, OneDrive, etc.)

Chapter 4
Plan of Project

4.1 Proposed System Architecture

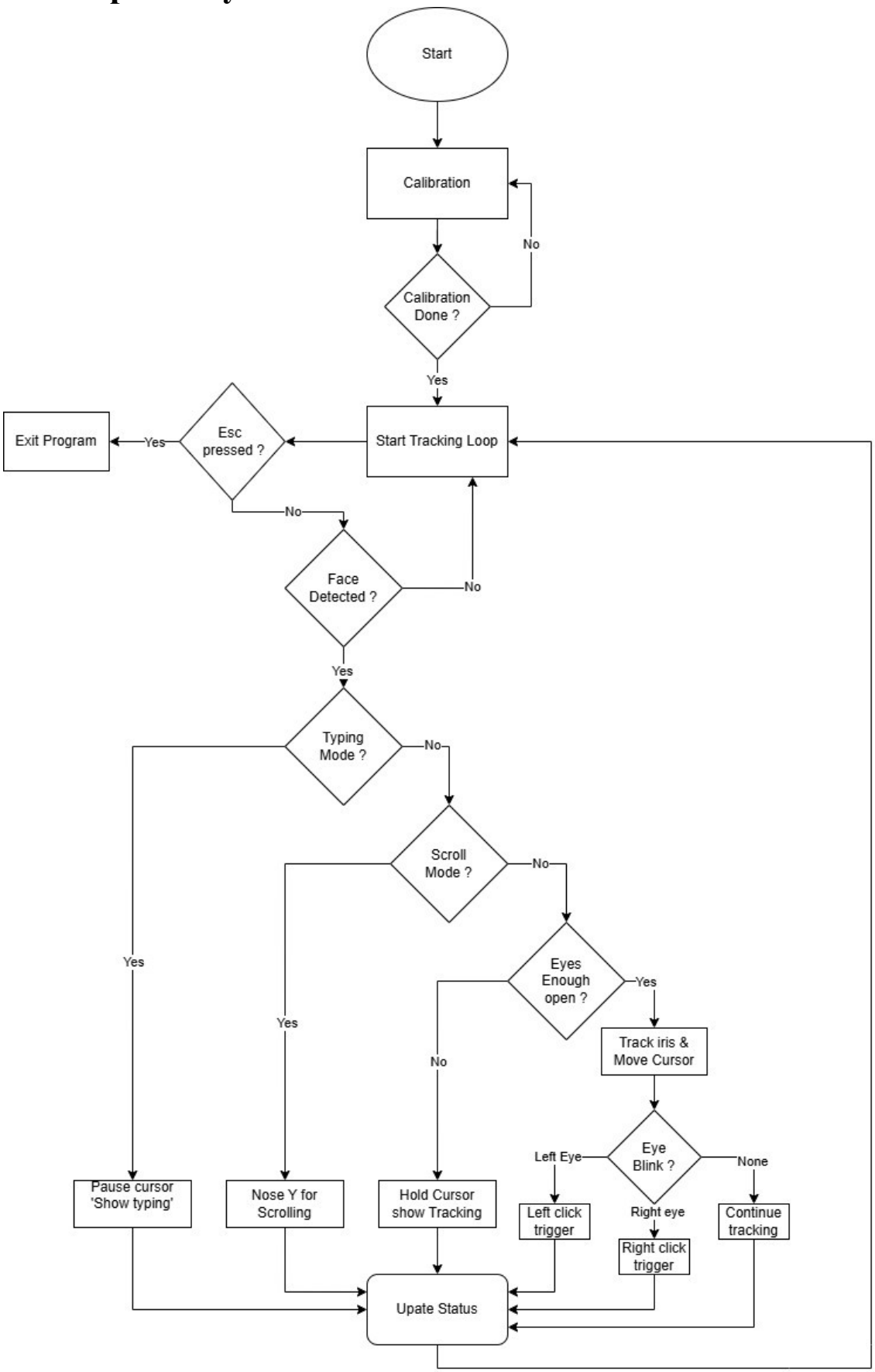

Figure 4.1 System Architecture

4.2 Methodology

In **Core Algorithms & Logic Used in the Project:**

1. **Facial Landmark Detection:**
    - Utilizes MediaPipe's Face Mesh CNN model.
    - Detects real-time 3D landmarks including iris, eyelids, lips, and nose points.
2. **Multi-point Calibration:**
    - Based on 9 known screen points for accurate gaze mapping.
    - Applies linear delta-scaling interpolation to translate raw pupil positions to screen coordinates.
    - Averages multiple pupil samples at each calibration point.
3. **Cursor Smoothing:**
    - Implements an Exponential Moving Average (EMA) filter.
    - Uses $\alpha = 0.2$ to smooth cursor movement and reduce jitter.
4. **Click & Scroll Mode Detection:**
    - Uses threshold-based classifiers:
        - Blink detection: Eyelid openness < 0.1 for 1.5s $\rightarrow$ triggers click.
        - Mouth openness: > 0.05 held for 3s $\rightarrow$ toggles scroll mode.
5. **Scroll Gesture Recognition:**
    - Activates in scroll mode only.
    - Monitors nose Y-axis displacement exceeding ±20 pixels.
    - Includes debounce logic with 0.2s delay to control scroll rate.
6. **Calibration Validation:**
    - Enforces head-centering logic.
    - Uses a tolerance of ±0.05 around the nose's normalized (x, y) position during calibration and tracking.
7. **Click Action Logic:**
    - Employs a parity-toggle mechanism:
        - Alternates between left and right clicks on successive blinks of either eye.

8. **System State Transitions:**
    - Operates as a finite state machine with the following modes:
        - Tracking Mode – Active gaze and click control.
        - Typing Mode – Triggered by recent keyboard input, suspends tracking.
        - Scroll Mode – Triggered via mouth gesture, enables vertical scrolling.
    - Designed to prioritize user intent and prevent unintended actions.

Chapter 5
Result Analysis

5.1 Results and discussion

The Eye-Tracking Mouse system was successfully implemented and tested, demonstrating accurate and responsive real-time cursor control using only eye movements and intentional blinks. The use of MediaPipe for facial landmark detection and PyAutoGUI for cursor actions ensured smooth operation with minimal latency. The system performed well under normal lighting conditions and required only a standard webcam, making it cost-effective and accessible. Blink detection reliably simulated mouse clicks, and users found the interface intuitive with minimal setup. Overall, the project achieved its goal of providing a practical, hands-free navigation tool for individuals with physical disabilities.

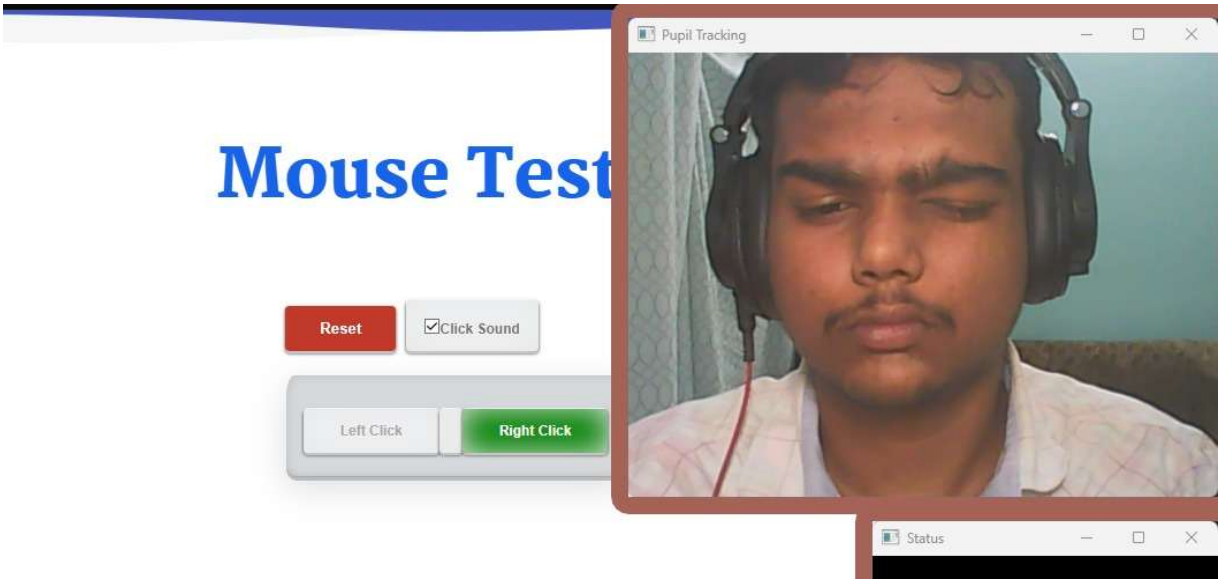

Figure 5.1 Right Click

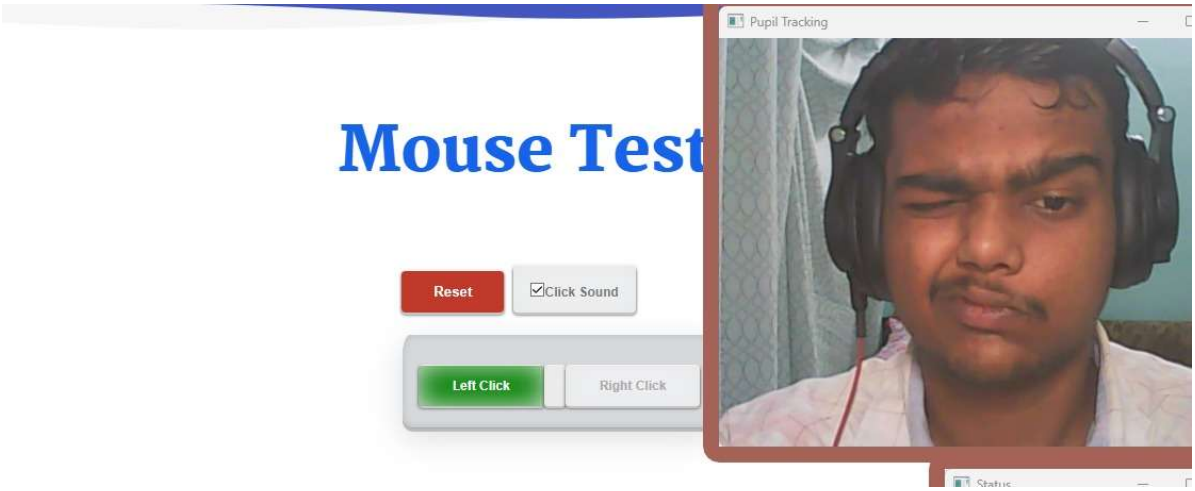

Figure 5.2 Right Click

Chapter 6
Conclusion

Conclusion

The Eye-Tracking Mouse System enables hands-free cursor control using eye movements and blinks. Built with Python and MediaPipe, it uses a standard webcam for real-time tracking. It's a cost-effective solution for accessibility and assistive tech, with future potential in AI and mobile integration.

References

Research Papers

1. **"Eye Controlled Human Computer Interface for Disabled Persons"**

- DOI: 10.1109/EmbeddedSys.2014.6953173
- This paper discusses using eye-tracking for cursor control and is aligned with your project's goals for accessibility.

2. **"Gaze-based Interaction in Human-Computer Interaction"**

- International Journal of Computer Applications
- DOI: 10.1145/1040830.1040843
- Covers fundamentals of gaze tracking and interaction techniques.

Web Resources

1. **MediaPipe Face Mesh Documentation (Google)**

- https://google.github.io/mediapipe/solutions/face_mesh
- Official resource for understanding how to extract facial landmarks, particularly eye points.

2. **PyAutoGUI Documentation**

- https://pyautogui.readthedocs.io/en/latest/
- Covers the library used for cursor movement and mouse actions.

3. **OpenCV Eye Tracking Tutorial**

- https://docs.opencv.org/4.x/d7/d00/tutorial_meanshift.html
- Good for learning eye motion tracking and frame processing techniques.

Video Tutorials

1. **"Build Eye-Controlled Mouse with Python & OpenCV" – Murtaza's Workshop**

- https://www.youtube.com/watch?v=k3PcVruvZCs
- A step-by-step visual explanation of creating an eye-tracking mouse system.

2. **"Blink Detection using Eye Aspect Ratio (EAR)" – PyImageSearch**

- https://www.youtube.com/watch?v=U9T6YkEDkMo
- Shows how to distinguish intentional blinks for input detection.