A Project Report

On

CLICK – AI Virtual Mouse

submitted for partial fulfillment of the requirements

for the award of the degree of

Bachelor of Technology

in

Computer Science

Submitted by

Sparsh Dagar (2000290120163)

Suryansh Shukla (2000290120171)

Sumit Agrawal (2000290120167)

Under supervision of

Mr. Vivek Kumar Sharma

Assistant Professor

KIET Group of Institutions, Ghaziabad

Dr. A.P.J. Abdul Kalam Technical University, Lucknow
May, 2024

DECLARATION

We hereby declare that this submission is our work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Sparsh Dagar (2000290120163)

Suryansh Shukla (2000290120171)

Sumit Agrawal (2000290120167)

Date:-   /   /2024

CERTIFICATE

This is to certify that the Project Report entitled "CLICK – AI Virtual Mouse" which is submitted by Sparsh Dagar, Suryansh Shukla and Sumit Agrawal in partial fulfillment of the requirement for the award of degree B. Tech. in the Department of Computer Science of Dr. A.P.J. Abdul Kalam Technical University, Lucknow is a record of the candidates own work carried out by them under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

Date: / /2024 **Supervisor**

Mr. Vivek Kumar Sharma

Assistant Professor

Department of Computer Science

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B.Tech. Final Year. We owe a special debt of gratitude to **Mr. Vivek Kumar Sharma, Department of Computer Science, KIET, Ghaziabad,** for his constant support and guidance throughout our work. His sincerity, thoroughness, and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavours have seen the light of the day.

We also take the opportunity to acknowledge the contribution of **Dr. Ajay Kumar Shrivastava, Head of the Department of Computer Science, KIET, Ghaziabad,** for his full support and assistance during the development of the project. We also do not like to miss the opportunity to acknowledge the contribution of all the department's faculty members for their kind assistance and cooperation during the development of our project.

Last but not least, we acknowledge our friends for their contribution to the completion of the project.

Sparsh Dagar (2000290120163)

Suryansh Shukla (2000290120171)

Sumit Agrawal (2000290120167)

Date: / /2024

ABSTRACT

The project introduces an AI-powered virtual mouse, a transformative technology poised to redefine the landscape of human-computer interaction (HCI). Utilizing advanced Artificial Intelligence (AI) and computer vision techniques, the virtual mouse interprets eye & lip gestures and movements, enabling precise and natural control of on-screen cursor dynamics. Through artificial intelligence, the virtual mouse intelligently adapts to user behaviour, tailoring its responses and interactions to individual preferences, thereby enhancing overall usability and productivity. Emphasizing inclusivity, the virtual mouse integrates accessibility features, accommodating a diverse user base. Moreover, its cross-platform compatibility ensures a seamless user experience across various operating systems and devices. This research presents the AI virtual mouse as a groundbreaking HCI tool, demonstrating its potential to revolutionize the way users interact with and navigate the digital realm.

TABLE OF CONTENTS

Page No.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Acronym	Meaning
HCI	Human Computer Interaction
CV	Computer Virtualization
AI	Artificial Intelligence
IDE	Integrated Development Environment
VR	Virtual Reality

CHAPTER 1
INTRODUCTION

1.1 Introduction to Project

1.1.1 Overview

Traditional computer mouse and touchpad interfaces can be limiting in terms of precision and accessibility for various users. The objective of this project is to develop an AI-powered virtual mouse system that can offer a more intuitive, precise, and adaptable pointing and clicking experience for users across different devices and scenarios. This AI virtual mouse should address issues related to input accuracy, user comfort, and accessibility, ultimately enhancing user interaction with digital interfaces. The primary objective of this invention is to accurately identify users eye movement and project the movement into the screen through virtual arrow movement of mouse, implemented by OpenCV (python library). The research demonstrates how models make predictions for the target classes. Real-time detection of users is pivotal for the effective working of the technology. The development of this system was inspired by comparative observations of real life events around the world. Using technological advances, we aim to develop an accurate and flexible framework for a virtual mouse which can be operated without any physical touch. With advancing technologies everything is becoming software oriented rather than hardware inclined. Also, with upsurge in COVID 19 cases globally in the past few years, hands free technologies are rising substantially. Therefore, this technology aims to provide virtual mouse movement, with the help of Python IDE – PyCharm and various python libraries, such as – Numpy, mediapipe, OpenCV.

1.1.2 Design

1. The system opens the camera of desktop and detects the eye and lip gesture of user using OpenCV modules,
2. Then, the cursor which is connected to system is traced and moves accordingly as the user moves.
3. If the user moves his/her lips and says “click”, the left click function of mouse is performed.
4. Finally, if the user moves out of bound, the system stops.

1.2 Project Category

Creating software that allows users to interact virtually with their desktop computers is the goal of CLICK – AI Virtual Mouse. This means creating and deploying user-friendly user interfaces, utilizing artificial intelligence and Computer virtualization to comprehend and react to commands from the user precisely, and control the mouse cursor using eye and lip gestures. The project aims to create a smooth and effective user experience while utilizing AI to improve accessibility and productivity in humancomputer interaction. It covers several software development stages, including requirements gathering, system design, implementation, testing, and maintenance.

1.3 Objective

The main objective of this project is to developing an AI virtual mouse system encompasses several key features aimed at revolutionizing user interaction with computing devices. Firstly, precision is paramount; the system must offer unparalleled accuracy in pointing and clicking, surpassing conventional input methods. Moreover, accessibility is crucial, ensuring inclusivity for users with physical disabilities through customizable input options and compatibility with alternative devices. Incorporating gesture recognition elevates user experience, enabling intuitive control via eye and lip movements for common mouse functions. Adaptive sensitivity further enhances versatility, dynamically adjusting to user preferences and task requirements, whether it be graphic design, gaming, or text editing. Finally, prioritizing natural interaction minimizes user fatigue and discomfort during prolonged use by mimicking organic eye and lip movements, fostering a seamless and ergonomic computing experience.

1.4 Structure of Report

1. **Introduction:** In this section, we provide a overview of our virtual mouse project, outlining its objectives and significance. We highlight the project's aim to develop an efficient virtual mouse system and its potential impact on user convenience and productivity.
2. **Literature Review:** The literature review delves into existing research and discourse surrounding virtual mouse. We analyze the evolution of virtual mouse technology, discussing advancements, user experiences, and pertinent security and privacy concerns.

3. **Proposed System:** This chapter outlines our proposed virtual mouse system, detailing its functionalities and the technologies utilized for its implementation. We elucidate the system's capabilities and how it addresses user requirements.
4. **Requirement Analysis and System Specification:** Here, we conduct a feasibility study of our proposed system and provide a detailed software requirement specification. We discuss the chosen SDLC model and its relevance to the project's development.
5. **Implementation:** In this section, we present an overview of the tools, and technologies employed for implementing the virtual mouse system. We delve into the libraries and algorithms utilized, offering insights into the implementation process and key modules.
6. **Testing and Maintenance:** This chapter focuses on testing techniques and methodologies utilized to ensure the functionality and reliability of our virtual mouse system. We discuss unit testing, integration testing, functional testing, usability testing, and performance testing, along with details of the test environment.
7. **Results and Discussions:** Here, we provide a summary of the various modules comprising the virtual mouse system and discuss the outcomes of our project. We analyze the results, interpret their implications, and offer insights into the findings.
8. **Conclusion:** In the conclusion, we encapsulate the project's objectives, achievements, and contributions. We reflect on key findings, discuss potential future research directions, and underscore the significance of our work.
9. **References:** The references section includes citations of relevant research papers, articles, and sources used throughout the report, ensuring transparency and academic integrity.

CHAPTER 2
LITERATURE REVIEW

2.1 Literature Review

A thorough Systematic Literature Review (SLR) was conducted for this study. In recent years, advancements in technology, particularly in the fields of human-computer interaction (HCI), computer vision, and gesture recognition, have catalyzed innovative approaches to real-time interaction and control systems. These technologies have garnered significant attention due to their potential to revolutionize user interfaces and accessibility across various domains.

One notable area of research focuses on cursor control systems utilizing hand gesture recognition. The seminal work by Nilesh J Uke (2013) and L. Somas (2018) has laid the foundation for this field. Uke's research introduces a system that allows users to control mouse cursor functions in real time using hand gestures captured by a camera. This approach not only enhances user-friendliness but also addresses the challenges posed by traditional pointing devices, particularly in mobile settings. Somas' extension of this work, known as the Digital Canvas, showcases the transformative potential of gesture recognition technology in digital art creation. By enabling artists to use hand gestures as brushes, the Digital Canvas opens up new avenues for creative expression and accessibility in the art world.

Furthermore, the integration of artificial intelligence (AI) and computer vision has led to the development of AI virtual mouse systems. S. Shriram's research (2021) represents a significant milestone in this area, presenting an AI virtual mouse system that utilizes hand gestures for cursor control. By leveraging deep learning techniques and computer vision algorithms, the proposed system achieves high accuracy and usability compared to traditional mouse input methods. Moreover, it addresses practical concerns such as the transmission of infectious diseases by eliminating the need for physical contact with input devices, thereby contributing to public health efforts such as the prevention of COVID-19 spread.

In parallel, the review conducted by J. Katona (2021) provides valuable insights into the broader landscape of HCI and virtual reality (VR) research within the cognitive infocommunications (CogInfoCom) domain. This comprehensive analysis highlights emerging trends and key contributions in HCI and VR technologies, underscoring their significance in enhancing human-computer interaction and immersive experiences.

In summary, these research endeavors collectively demonstrate the transformative potential of technology in addressing practical challenges and improving user experiences. By leveraging computer vision, gesture recognition, and AI, researchers have developed innovative solutions for cursor control systems, digital art creation, and immersive gaming experiences. These advancements not only enhance accessibility but also contribute to broader societal goals, such as public health and education.

2.2 Research Gaps

Identified research gaps:

1. **Interface Optimization**: Further research can refine gesture recognition algorithms to enhance accuracy and responsiveness, improving user experience.
2. **Accessibility:** Investigating how gesture-based interfaces can accommodate users with diverse abilities ensures equitable access to digital art tools.
3. **Healthcare Applications:** More studies are needed to tailor AI virtual mouse systems for specific healthcare needs like patient monitoring and surgical navigation.
4. **Security:** Research on potential vulnerabilities in gesture-based interfaces and development of robust security mechanisms is necessary.
5. **User Acceptance:** Understanding factors influencing user perceptions and barriers to adoption can improve the usability of gesture-based interfaces

2.3 Problem Formulation

1. **Integration Challenges**: Harmonizing the diverse methodologies from the literature surveys, including OpenCV, gesture recognition, and hardware interfaces, might pose difficulties in ensuring seamless compatibility and interoperability among different components of the system.

2. **User Adaptation and Learning Curve:** Users may encounter challenges in acclimating to the novel interaction paradigms introduced by the project, whether it involves mastering hand gesture based cursor control or navigating virtual environments. Overcoming the learning curve and fostering intuitive user experiences are pivotal for widespread acceptance and usage.

3. **Hardware and Software Compatibility:** Ensuring smooth integration across various hardware setups (e.g., cameras, VR headsets) and software platforms (e.g., operating systems, development environments) is essential for the successful deployment and functionality of the project across diverse computing environments.

4. **Performance Optimization:** Achieving optimal performance of real-time computer vision algorithms, particularly in resource-constrained environments or computationally intensive applications, is paramount for delivering responsive and accurate gesture recognition, cursor control, and virtual reality interactions.

5. **Ethical and Privacy Concerns:** Leveraging computer vision technologies for tasks like gesture recognition and user tracking necessitates addressing ethical considerations surrounding privacy, consent, and data security. Implementing robust privacy protocols and transparent data handling practices is imperative to safeguard user trust and mitigate potential risks.

6. **Validation and Evaluation:** Conducting thorough validation and evaluation in real-world contexts is indispensable for assessing the effectiveness and usability of the project across diverse user demographics and application scenarios. Rigorous user testing, feedback collection, and iterative refinement are pivotal steps toward ensuring the project's success and impact

CHAPTER 3 PROPOSED SYSTEM

3.1 Proposed System

The proposed system is an eye-controlled mouse interface that allows users to interact with a computer without using traditional input devices such as a mouse or keyboard. Using a webcam, the system detects and tracks the user's eye movements in real-time. By analysing specific landmarks on the face, such as the eyes and mouth, the system interprets gestures made by the user's eyes, such as blinks or gaze directions. These eye gestures are then mapped to corresponding cursor movements and actions on the computer screen. Additionally, the system can detect mouth gestures, such as blinks, to perform additional actions, such as mouse clicks. By leveraging computer vision techniques and machine learning models, combined with automation libraries like PyAutoGUI, the proposed system offers a hands-free and intuitive method for users to control the cursor and interact with the computer interface solely through their eye movements and gestures. This system has potential applications in accessibility, assistive technology, and human-computer interaction, offering an alternative input method for users with mobility impairments or those seeking a more natural and efficient way to interact with computers.

The Technology used in the invention is as follows:

- Windows-10 OS
- Python Programing Language
- PyCharm – python IDE
- OpenCV
- Mediapipe
- PyautoGUI
- Numpy

3.2 Unique Features of the System

The system helps in performing basic cursor functions using eye and lip gesture. The systems functionality can be summarized as –

- Implementing the system installation on computers.
- Employing precise cameras to detect the facial movements.

- Alerting the user if face out of the screen.
- Detecting the minute eye and lips movement to provide the user with well-functioning arrow movement of virtual mouse.
- The sensitivity of the arrow movement could be set according to the user’s comfort.

CHAPTER 4
REQUIREMENT ANALYSIS AND SYSTEM SPECIFICATION

4.1 Feasibility Study

The purpose of the feasibility study is to evaluate the possibility and viability of putting in place a facial recognition-based automated cursor movement system. By recognizing and tracking the facial expressions the trained project aims to operate the cursor handsfree. To ascertain the viability of the project, the technical, financial, operational, and scheduling factors are assessed.

1. **Technical Feasibility -**

   Facial Recognition Technology: Assess the scalability, accuracy, and dependability of software and algorithms for facial recognition.

   Hardware Requirements: Examine the cameras and processing units, among other hardware elements, required for facial recognition.

   Integration: Examine how well the facial recognition system integrates with the real time camera of the user.

2. **Economic Feasibility -**

   Cost analysis: Calculate the out-of-pocket expenses related to software development, system integration, and hardware acquisition.

   Return on Investment (ROI): Figure out how much you could save by having less administrative work and more accurate attendance records.

   Total Cost of Ownership (TCO): Examine the enduring expenses, including upkeep, modifications, and assistance.

3. **Operational Feasibility –**

   User Acceptance: Determine whether users i.e. students, staff members, and administrators are willing to accept and utilize facial recognition technology to operate cursor.

   Training Requirements: Find out what kind of training users need to do in order to properly engage with the system.

   Privacy and Security: Talk about worries about data security and privacy related to face recognition technologies.

4.2 Software Requirement Specification

4.2.1 Data Requirement

(a) The system should be capable of accessing the desktop camera.

(b) It should detect the user's eye and lip gesture.

(c) It should have the access to desktop's cursor and it's basic functionalities.

4.2.2 Functional Requirements

(a) Camera: The system should be able to capture real time video of user.

(b) Command Interpretation and Execution: The system should analyze the eye and lip gesture and control the cursor movement in real time. Upon recognizing gesture, the system should execute the corresponding action, such as moving cursor and performing basic cursor functions.

(c) Response Generation: The system should be able to move the cursor accordingly and perform the click function of mouse.

4.2.3 Performance Requirements

(a) Response Time: The system should provide prompt responses to user inputs, aiming for minimal latency in processing and execution.

(b) Accuracy: Capturing eye and lip gesture and perform the cursor functions with high accuracy.

(c) Resource Efficiency: The system should utilize system resources efficiently to ensure smooth operation without excessive CPU or memory usage.

4.2.4 Maintainability Requirements

(a) Modularity: The system architecture should be modular, allowing for easy maintenance, updates, and scalability.

(b) Documentation: Comprehensive documentation should be provided to facilitate understanding, troubleshooting, and future enhancements.

(c) Error Handling: If any error occurs then system should be able to tackle that error efficiently and give user a interactive and smooth experience.

4.2.5 Security Requirements

(a) **Data privacy:** User data, including video inputs and personal information, should be handled securely and in compliance with privacy regulations.

(b) **Access Control:** Access to sensitive functionalities or data within the system should be appropriately restricted to authorized users or roles.

(c) **Secure Communication:** Any communication between the virtual mouse system and external services or APIs should be encrypted to prevent unauthorized access or data interception.

4.3 SDLC Model

Iterative Model: The Iterative model is a software development approach where the project is divided into small increments or iterations, each building upon the previous one. Here are some key characteristics:

(a) Incremental Development: The project progresses through a series of iterations, with each iteration adding new features, enhancements, or improvements to the product. Each iteration results in a working version of the software.

(b) Feedback Loop: Iterative development incorporates feedback from stakeholders, users, and team members at the end of each iteration. This feedback informs the next iteration, guiding the direction of development and ensuring alignment with requirements.

(c) Flexible Planning: Iterative projects have a flexible planning approach, with requirements and priorities evolving. This flexibility allows for adjustments based on feedback, lessons learned, and changing business needs.

(d) Progressive Refinement: Each iteration builds upon the previous one, gradually refining the product and adding new features or improvements. This incremental approach enables early delivery of value while maintaining focus on quality and usability.

(e) Continuous Validation: Throughout the iterative development process, the product is continuously validated against user needs, expectations, and acceptance criteria. This validation ensures that the product meets quality standards and delivers value to stakeholders.

(f) Parallel Development: Iterative development often involves parallel workstreams, where multiple iterations may be in progress simultaneously. This parallelism allows for efficient use of resources and accelerates the pace of development.

4.4 System Design

4.4.1 Detail Design

This system will work in the following ways -

- Implementing the system installation on computers.
- Employing precise cameras to detect the facial movements.
- Alerting the user if face out of the screen.
- Detecting the minute eye and lips movement to provide the user with well functioning arrow movement of virtual mouse.

4.4.2 System Design using DFD

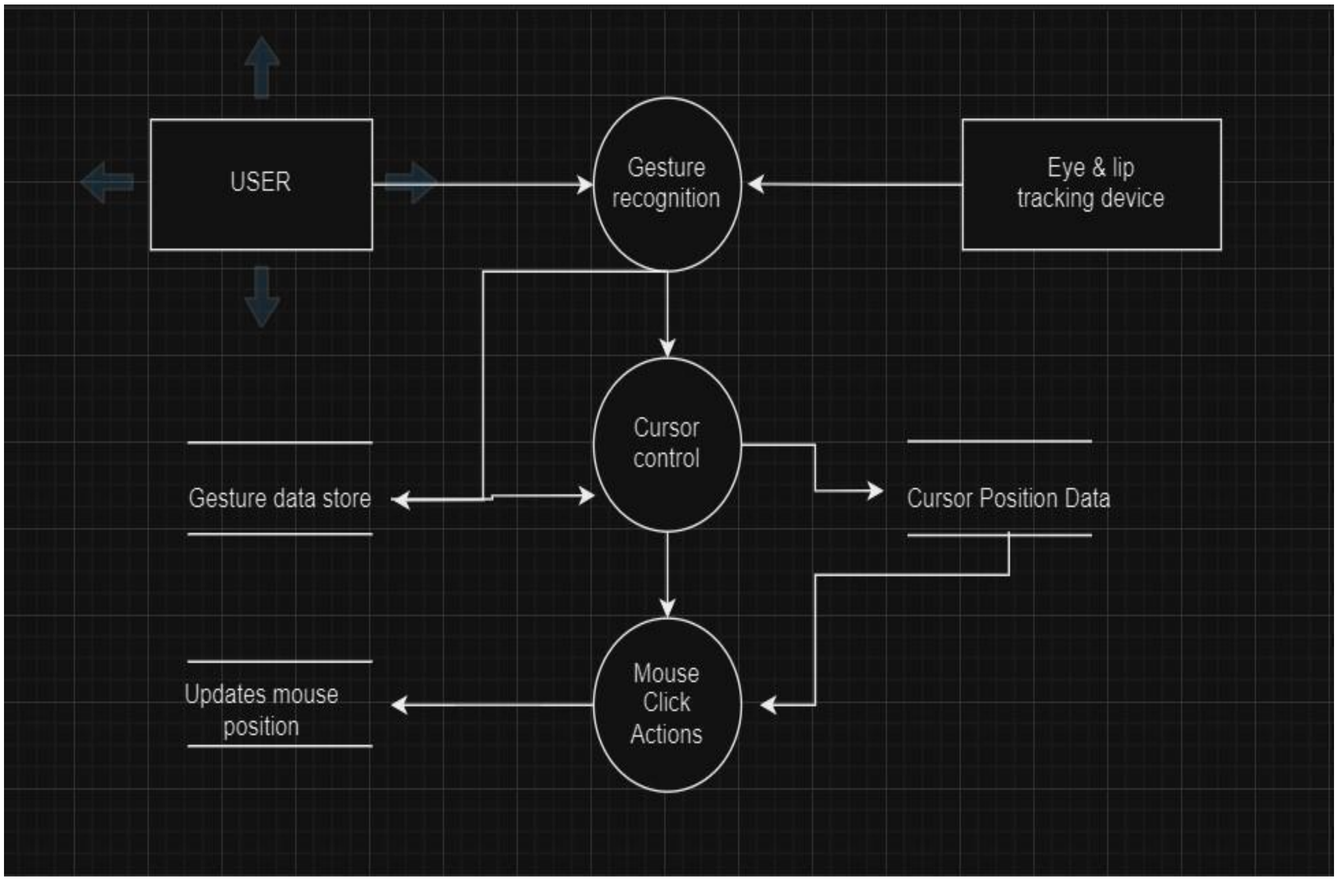


Fig 4.1 DFD

The flowchart shows the steps of listening and recognizing the command, performing the task, which is divided into system actions and cam actions, and then waiting for the next command. This diagram is likely used to explain how virtual mouse systems process and respond to user commands.

4.4.3 Use Case Diagram

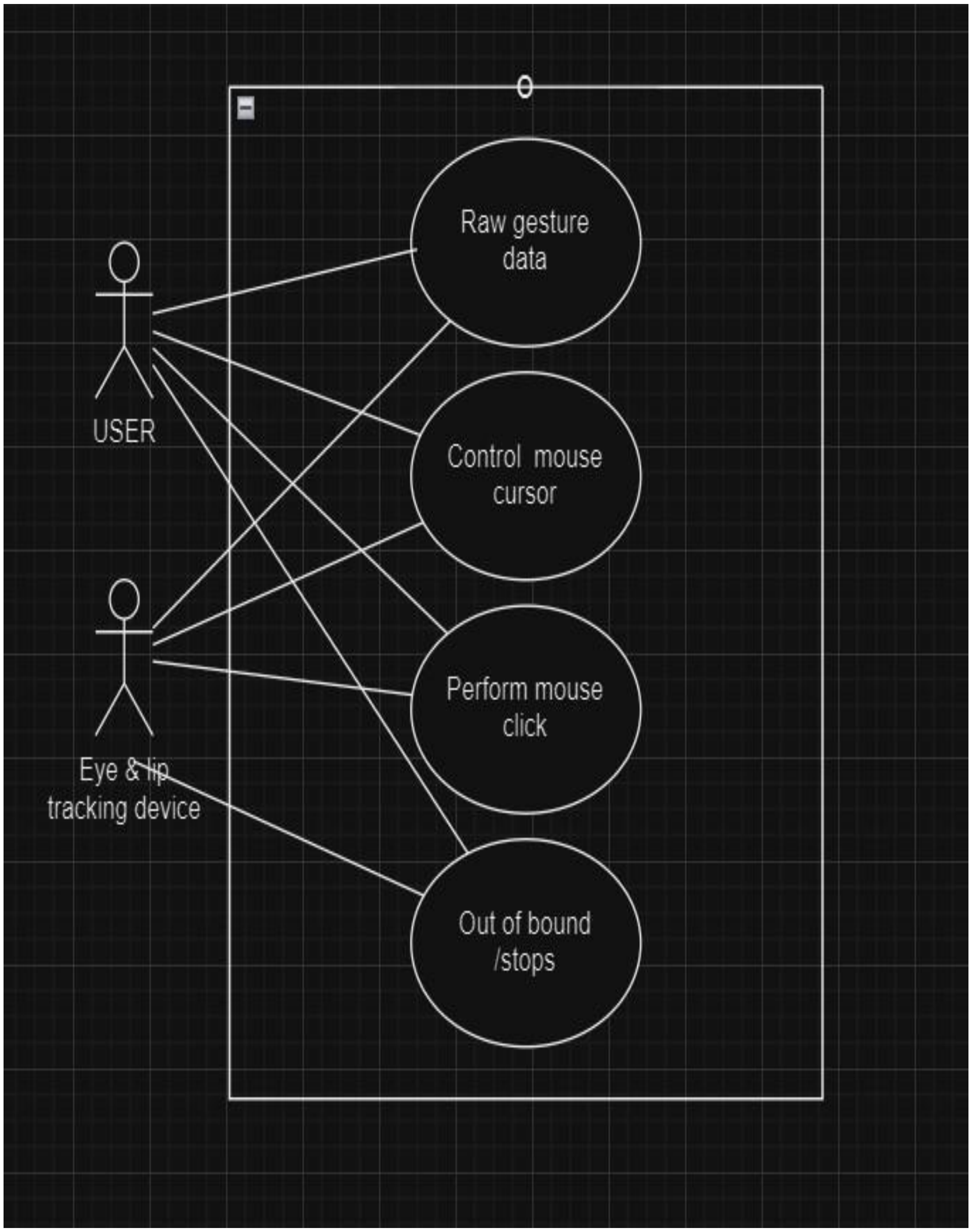


Fig 4.2 Use Case Diagram

4.4.4 Flow Chart

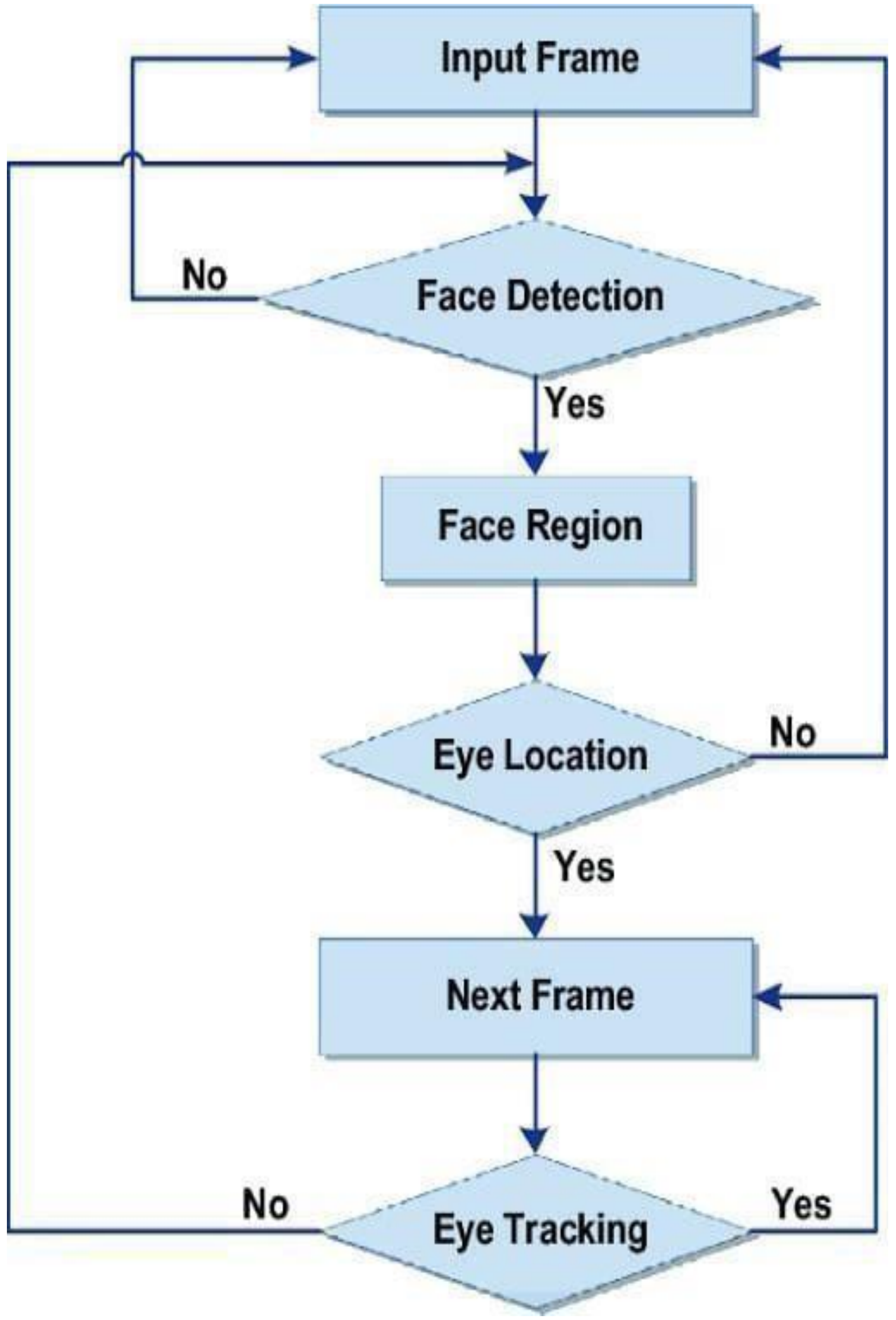


Fig 4.3 FlowChart

4.4.5 Entity-relationship diagram

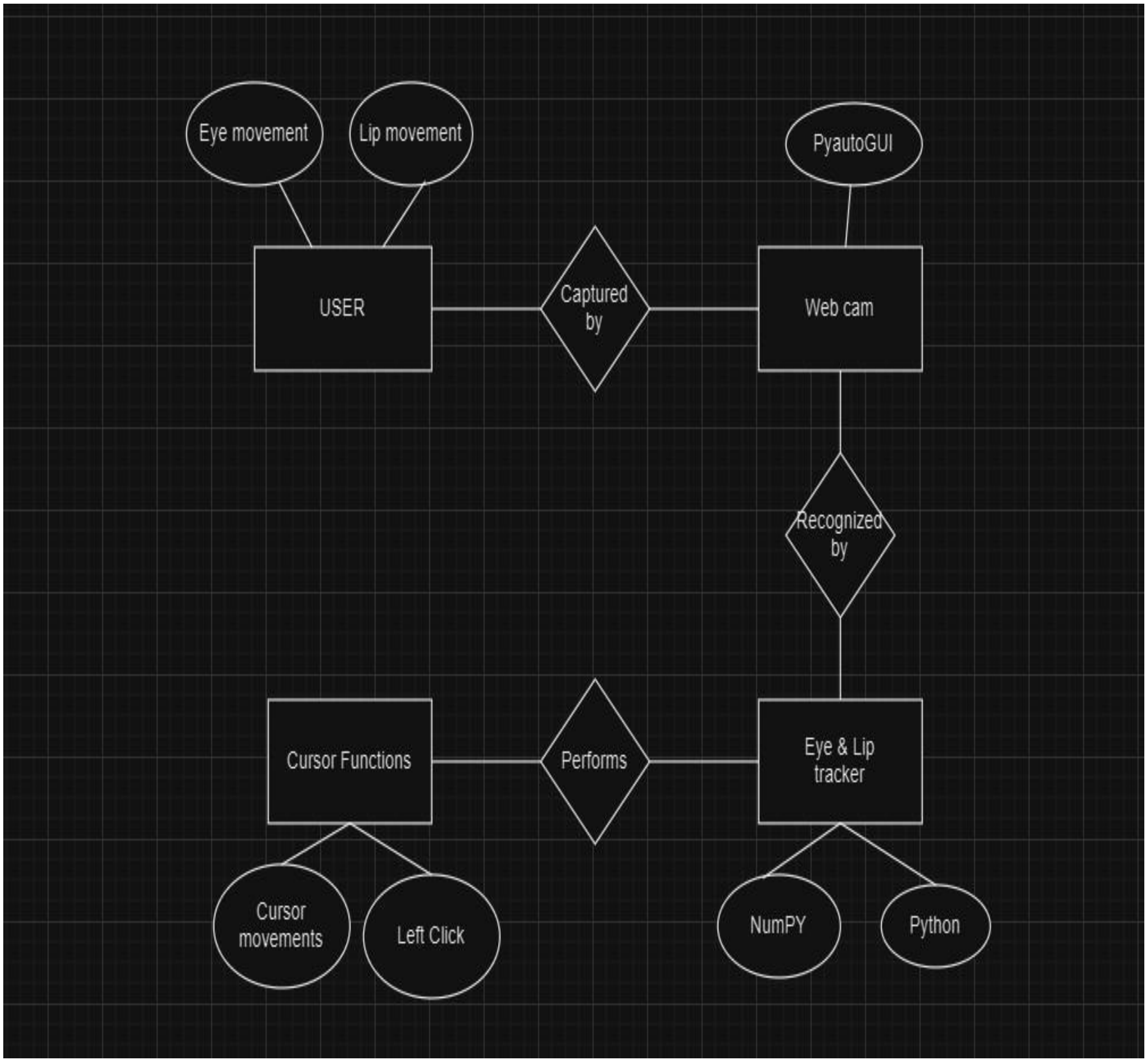


Fig 4.4 ER Diagram

CHAPTER 5 IMPLEMENTATION

5.1 Introduction to Languages, Tools & Technologies

5.1.1 Python

Python is the primary programming language used in our project for implementing the eye-controlled mouse interface. It serves as the foundation for integrating various libraries and modules necessary for the system's functionality. Here's how Python is used in our project:

1. **Library Integration:** Python allows us to seamlessly integrate different libraries such as OpenCV, Mediapipe, PyAutoGUI, and NumPy. These libraries provide essential functionalities for tasks like webcam access, facial landmark detection, gesture recognition, mouse control, and numerical computations.
2. **Algorithm Implementation:** Python facilitates the implementation of algorithms for processing video frames, detecting facial landmarks, analyzing eye gestures, and mapping them to cursor movements and actions on the screen. The flexibility and ease of use of Python make it ideal for prototyping and experimenting with different approaches.
3. **Real-time Processing:** Python enables real-time processing of video frames captured from the webcam, allowing us to continuously analyze and interpret the user's eye movements and gestures. With its efficient performance and support for multi-threading, Python ensures smooth and responsive interaction with the computer interface.
4. **User Interface:** Python can be used to develop a simple graphical user interface (GUI) for displaying visualizations, debugging information, and user feedback. While not explicitly mentioned in the provided code snippet, Python's libraries like Tkinter or PyQt could be employed for this purpose.

5.1.2 Libraries Used

1. **OpenCV (Open Source Computer Vision Library):**

   Description: OpenCV is an open-source computer vision and machine learning software library. It provides a wide range of functionalities for processing images and videos, including image manipulation, object detection and tracking, facial recognition, and more.

   Key Features:

   - Image and video I/O: OpenCV can read, write, and display images and videos in various formats.

- Image processing: It offers a plethora of functions for tasks such as filtering, edge detection, image transformations, and morphology.
- Object detection and tracking: OpenCV provides algorithms and pre-trained models for detecting and tracking objects in images and videos.
- Machine learning support: It includes tools for machine learning tasks such as classification, clustering, and regression.
- Applications: OpenCV is widely used in various fields, including robotics, augmented reality, surveillance, medical imaging, and more.

2. **Mediapipe:**

Description: MediaPipe is an open-source framework developed by Google for building machine learning pipelines for audio, video, and multimodal applications. It offers pre-built solutions and components for tasks such as hand tracking, pose estimation, face detection, and gesture recognition.

Key Features:

- Hand tracking: MediaPipe provides a robust and efficient hand tracking solution that can accurately detect and track hand movements in real-time.
- Pose estimation: It includes pose estimation models that can detect human body keypoints and estimate poses from images or videos.
- Face detection and recognition: MediaPipe offers components for detecting faces in images or videos and recognizing facial landmarks.
- Applications: MediaPipe is commonly used in applications such as virtual try-on, gesturebased interfaces, sign language recognition, and augmented reality filters.

3. **PyAutoGUI:**

Description: PyAutoGUI is a Python library that enables programmatically controlling the mouse and keyboard to automate tasks on a computer. It provides cross-platform support for simulating mouse movements and clicks, keyboard inputs, and screen interactions.

Key Features:

- Mouse control: PyAutoGUI allows developers to move the mouse cursor, click mouse buttons, and scroll the mouse wheel programmatically.
- Keyboard control: It enables sending keystrokes, key combinations, and keyboard shortcuts to interact with applications and input text.
- Screen interaction: PyAutoGUI can capture screenshots, locate images on the screen, and perform actions based on visual patterns.
- Applications: PyAutoGUI is used for automating repetitive tasks, testing GUI applications, creating bots for games or simulations, and building accessibility tools for users with disabilities.

4. **NumPy (Numerical Python):**

Description: NumPy is a fundamental package for scientific computing in Python. It provides support for multidimensional arrays, mathematical functions, linear algebra operations, random number generation, and more.

Key Features:

- N-dimensional array support: NumPy offers a powerful array object that enables efficient manipulation of large datasets and multidimensional numerical computations.
- Mathematical functions: It includes a wide range of mathematical functions for arithmetic operations, trigonometry, logarithms, exponents, and more.
- Linear algebra: NumPy provides functions for matrix operations, such as matrix multiplication, inversion, eigenvalue decomposition, and solving linear equations.
- Random number generation: It offers facilities for generating random numbers and sampling from probability distributions.
- Applications: NumPy is extensively used in scientific computing, data analysis, machine learning, signal processing, image processing, and computational mathematics.

These libraries collectively provide a robust foundation for implementing computer vision-based projects, enabling tasks such as image processing, object detection, gesture recognition, automation, and scientific computing in Python.

5.1.3 Algorithm Used

5.1.3.1 Eye Gesture Recognition Algorithm:

- Capture and process eye movements using computer vision techniques, possibly leveraging OpenCV for image processing.
- Detect and classify specific eye gestures, such as blinks, gazes, or movements, using predefined algorithms or machine learning models.
- Extract relevant features from the detected eye gestures, such as direction, duration, or intensity, to interpret user intent accurately.

5.1.3.2 Gesture Mapping and Cursor Control Algorithm:

- Map recognized eye gestures to corresponding cursor movements and actions, such as cursor positioning, clicking, or dragging.
- Update the cursor's position on the screen dynamically based on the recognized eye gestures and user input.
- Simulate mouse movements and clicks in real-time to interact with graphical user interfaces or applications, ensuring seamless control and responsiveness.

5.1.3.3 Real-time Interaction:

- Enable users to navigate and interact with the computer interface using eye gestures captured from the camera feed.
- Provide an intuitive and hands-free method for controlling the cursor and executing actions on the screen, enhancing accessibility and user experience.
- Ensure robustness and accuracy of the system in real-world scenarios, considering factors such as lighting conditions, user variability, and environmental noise for reliable interaction.

CHAPTER 6
TESTING AND MAINTENANCE

6.1 Testing Techniques & Test Cases Used

6.1.1 Testing Techniques

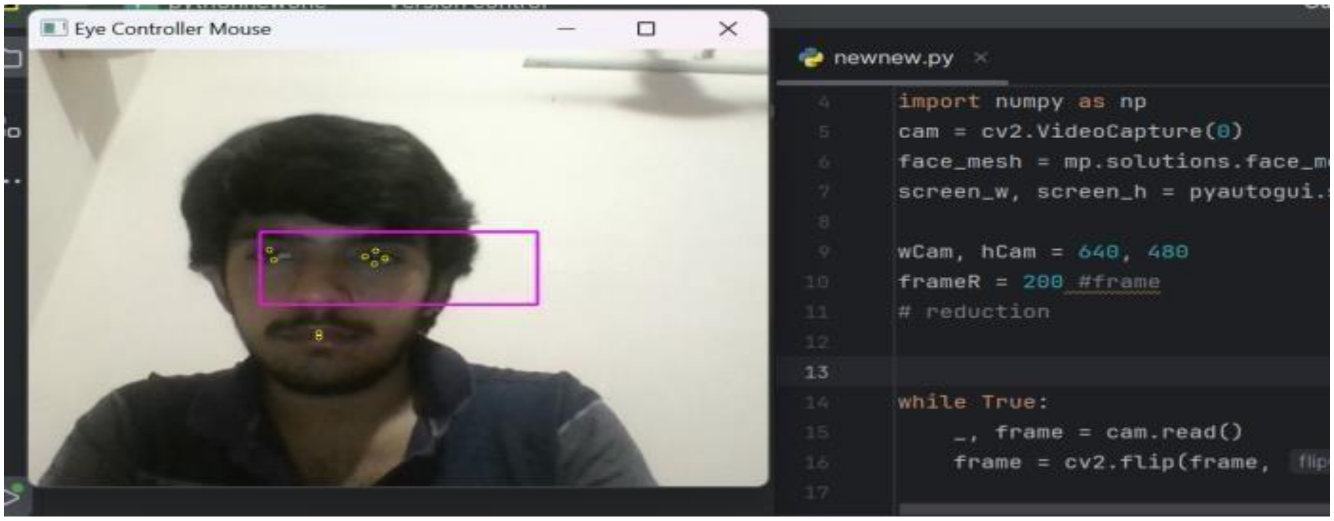
Manual testing is performed for the eye gestures detection and lip detection have been tested in various illumination conditions and also been tested with different distances from the webcam for tracking of the eye gesture and lip detection. An experimental test has been conducted to summarize the results. The test was performed 25 times by a person and this test has been made in different conditions and at different distances from the screen, and person tested the AI virtual mouse system.

In each set of test, 12 actions are performed and actual results are recorded and then analyzed against the expected output and a remark is given as fail or pass. If a majority of these actions are marked as Pass, then only the test is considered as passed. The same set of test is repeated 25 times and each time a test is performed, we perform tap function 2 times to test the cursor funtions.

This is an example set of actions, expected output, actual output and the remark, each time the system is tested.

Out of 25 times, the system was able to detect the eye gesture 24 times i.e. accuracy of 96% and moved the cursor in the right direction, out of 50 times saying “tap” the system was able to perform click function 46 times i.e. the accuracy is 92%.

- **Successful detection of eye gestures**.



```
import numpy as np
cam = cv2.VideoCapture(0)
face_mesh = mp.solutions.face_m
screen_w, screen_h = pyautogui.

wCam, hCam = 640, 480
frameR = 200 #frame
# reduction


while True:
    _, frame = cam.read()
    frame = cv2.flip(frame,
```

Fig 4.5 Eye detection

- **Moving out of the screen.**

  This message is shown in console -

  File"C:\Users\dagar_okcnktp\PycharmProjects\pyth
  onnewone\.venv\lib\sitepackages\pyautogui\__init__.py", line 1734, in failSafeCheck raise
  FailSafeException(pyautogui.FailSafeException: PyAutoGUI fail-safe triggered from mouse moving to a corner of the screen. To disable this fail-safe, set pyautogui. FAILSAFE to False. DISABLING FAIL-
  SAFE IS NOT RECOMMENDED.
  Process finished with exit code 1.

- **Detection of lip movement.**

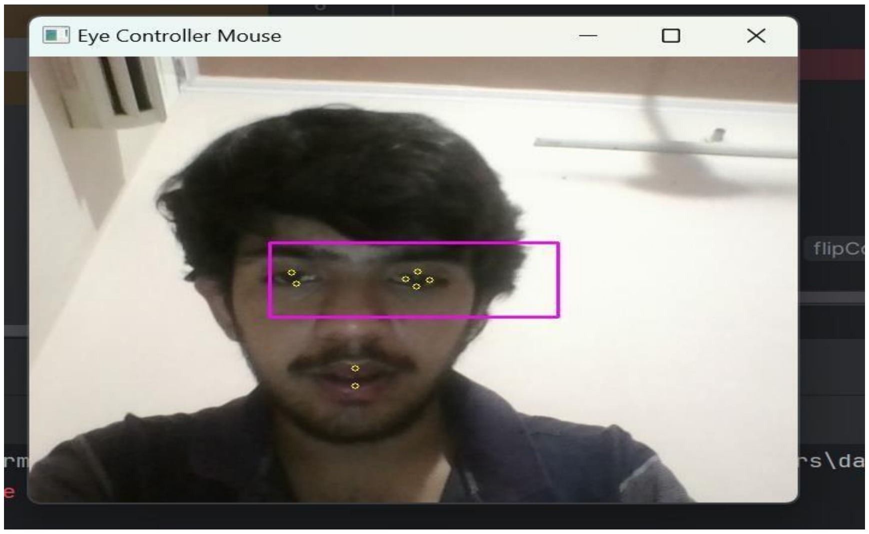
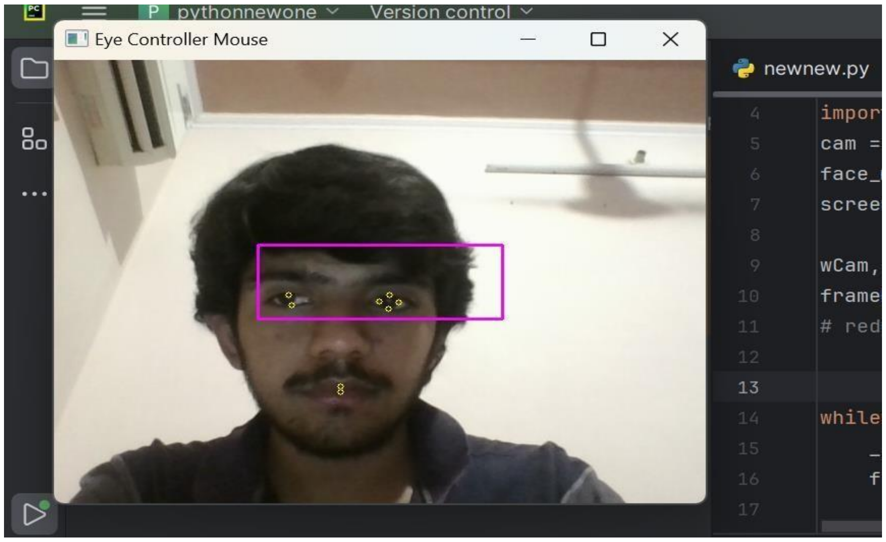


Fig 4.6 Lip movement detection

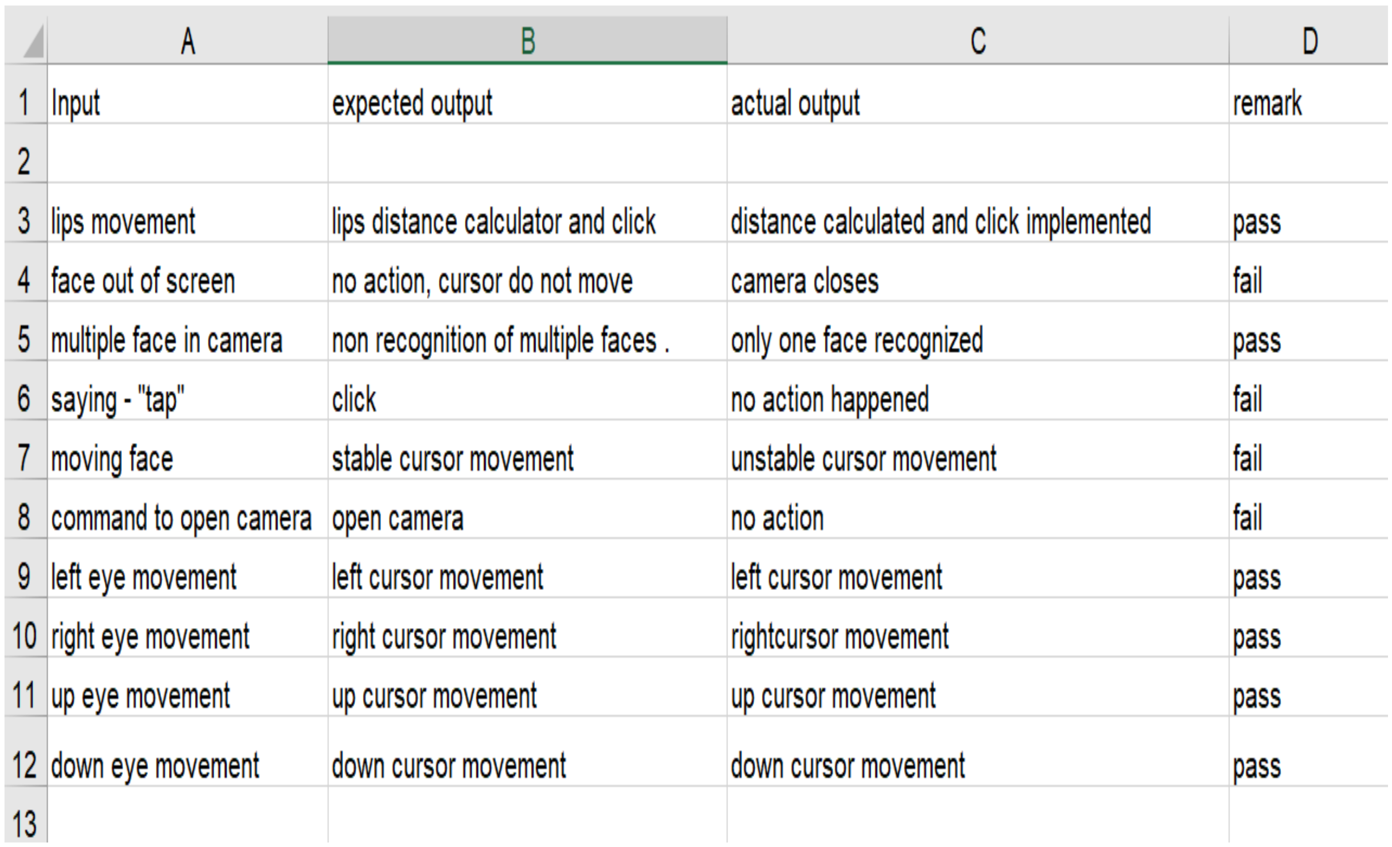


```
impor
cam =
face_
scree

wCam,
frame
# red


while
    _
    f
```

Fig 4.7 Close lip detection

When the distance between both the lips increases by certain distance then the “Click” function is performed.

6.1.2 Test Cases Used –



	A	B	C	D
1	Input	expected output	actual output	remark
2				
3	lips movement	lips distance calculator and click	distance calculated and click implemented	pass
4	face out of screen	no action, cursor do not move	camera closes	fail
5	multiple face in camera	non recognition of multiple faces .	only one face recognized	pass
6	saying - "tap"	click	no action happened	fail
7	moving face	stable cursor movement	unstable cursor movement	fail
8	command to open camera	open camera	no action	fail
9	left eye movement	left cursor movement	left cursor movement	pass
10	right eye movement	right cursor movement	rightcursor movement	pass
11	up eye movement	up cursor movement	up cursor movement	pass
12	down eye movement	down cursor movement	down cursor movement	pass
13				

Table 6.1 Testing table

6.2 Test Environment

It mentions the minimum **hardware** requirements that will be used to test the Application. The following **software is** required in addition to client-specific software.

- Windows 10 and above
- Minimum 4GB RAM
- Minimum Intel Core i3 or above.
- Microphone for input.
- Latest VS Code (Updated)
- Chrome, Mozilla, or Edge is preferred over non-chromium-based browsers.

CHAPTER 7
RESULTS AND DISCUSSIONS

7.1 Brief Description of Various Modules

1. CV2 (OpenCV):

OpenCV is a popular open-source computer vision library used for image and video processing tasks, including reading/writing image files, webcam access, and various image manipulation operations.

2. mediapipe:

Mediapipe is a framework developed by Google for building machine learning pipelines for audio, video, and multimodal applications. In this code, it's specifically used for facial landmark detection with the FaceMesh model.

3. pyautogui:

PyAutoGUI is a Python library for programmatically controlling the mouse and keyboard to automate tasks on a computer. It's used in this code to simulate mouse movements, clicks, and keyboard inputs.

4. numpy (np):

NumPy is a fundamental package for numerical computing in Python. It provides support for multidimensional arrays, mathematical functions, linear algebra operations, and random number generation. It's used in this code for numerical computations, particularly for interpolation.

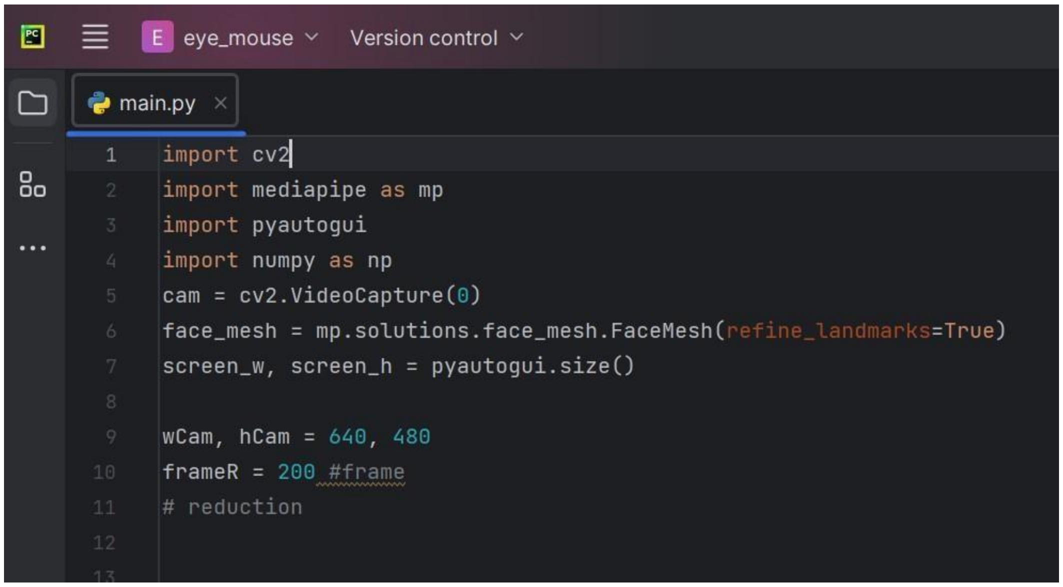
7.2 Snapshots of System Features

1. Importing Libraries:

- The code begins by importing the necessary libraries:
- cv2: OpenCV library for computer vision tasks.
- mediapipe: Mediapipe library for facial landmark detection.
- pyautogui: PyAutoGUI library for simulating mouse and keyboard inputs.
- numpy: NumPy library for numerical computations.

2. Initializing Variables:

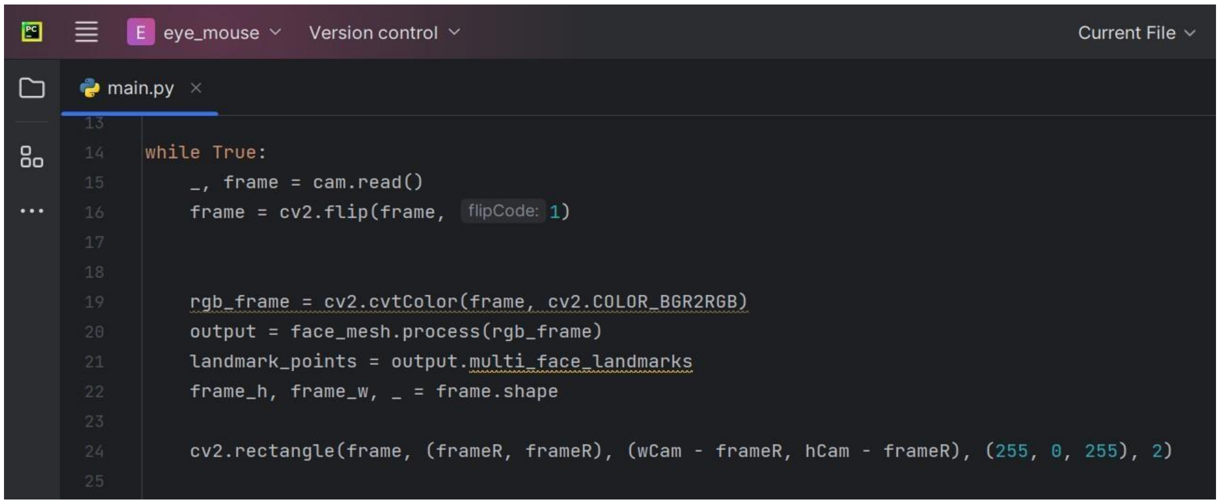
- cam: Initializes the webcam for capturing video frames.
- face_mesh: Initializes the FaceMesh model from the Mediapipe library for detecting facial landmarks.
- screen_w, screen_h: Retrieves the screen resolution using PyAutoGUI's size() function.
- wCam, hCam: Defines the width and height of the webcam feed.
- frameR: Defines the size of the frame reduction for processing.



```python
import cv2
import mediapipe as mp
import pyautogui
import numpy as np
cam = cv2.VideoCapture(0)
face_mesh = mp.solutions.face_mesh.FaceMesh(refine_landmarks=True)
screen_w, screen_h = pyautogui.size()

wCam, hCam = 640, 480
frameR = 200 #frame
# reduction
```

3. Main Loop:

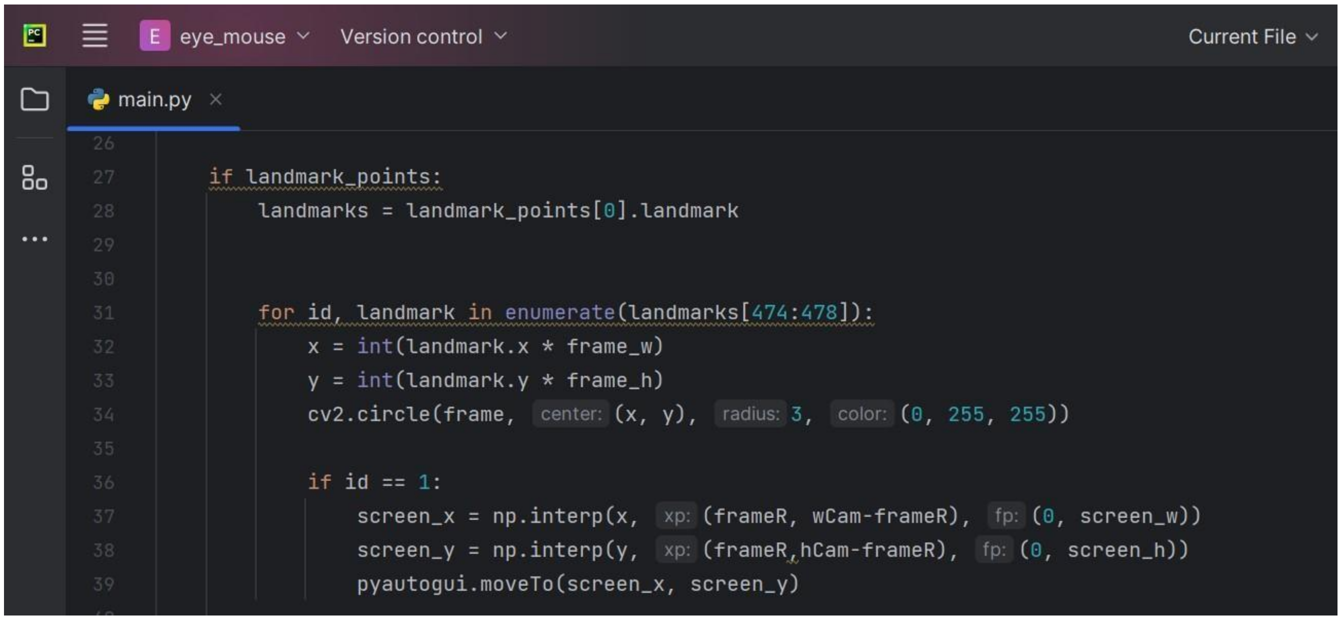
- The code enters a continuous loop (while True) to capture and process video frames from the webcam.



```python
while True:
    _, frame = cam.read()
    frame = cv2.flip(frame,  flipCode: 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

    cv2.rectangle(frame, (frameR, frameR), (wCam - frameR, hCam - frameR), (255, 0, 255), 2)
```

4. Frame Processing:

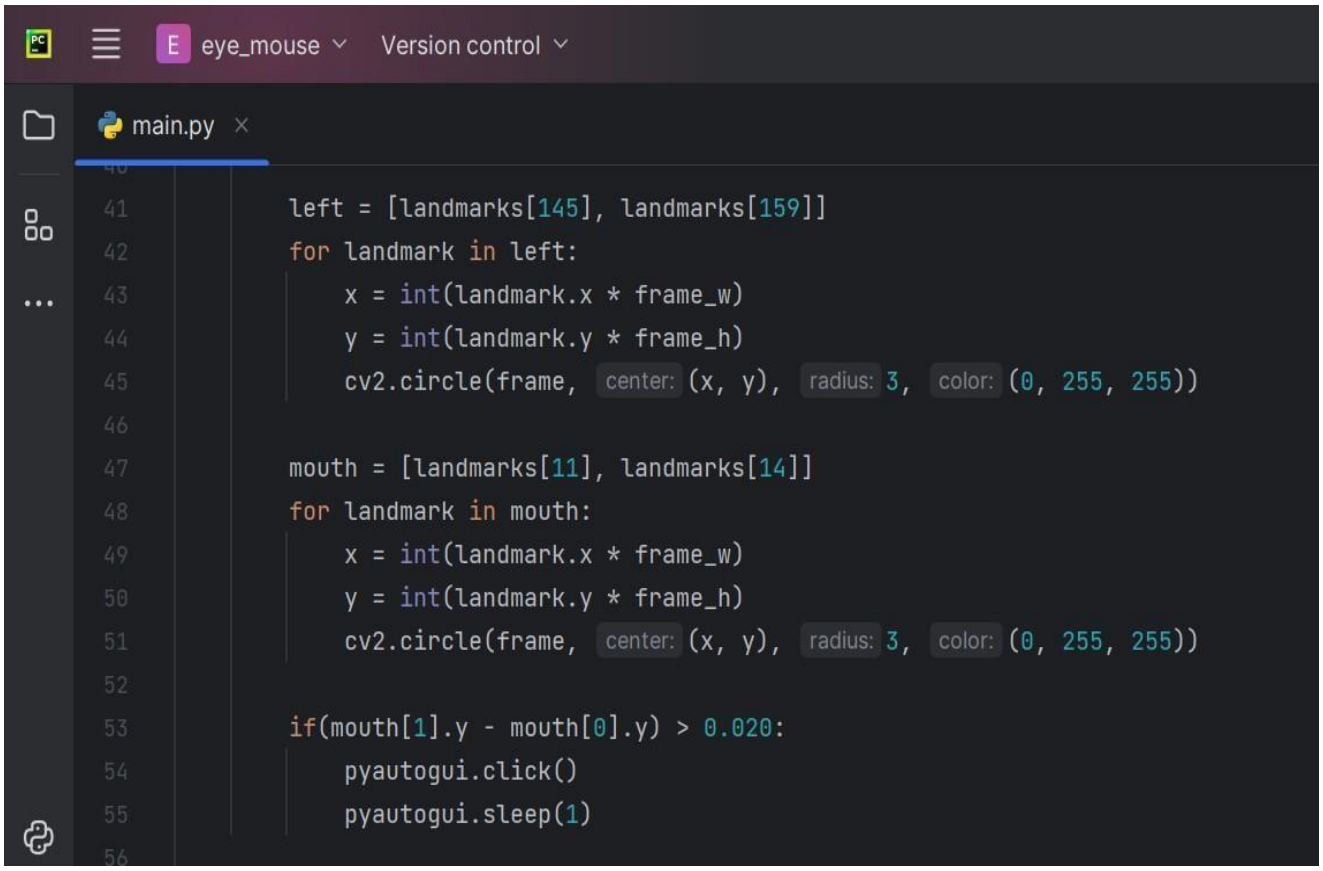
- Within the loop, it reads a frame from the webcam using cam.read() and flips it horizontally to correct for the mirror effect using cv2.flip().
- The frame is converted from BGR to RGB color space using cv2.cvtColor() for processing with Mediapipe.
- The FaceMesh model detects facial landmarks in the RGB frame using face_mesh.process(), and the resulting landmark points are stored in landmark_points.



```
    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, center: (x, y), radius: 3, color: (0, 255, 255))

            if id == 1:
                screen_x = np.interp(x, xp: (frameR, wCam-frameR), fp: (0, screen_w))
                screen_y = np.interp(y, xp: (frameR,hCam-frameR), fp: (0, screen_h))
                pyautogui.moveTo(screen_x, screen_y)
```

5. Gesture Detection and Cursor Control:

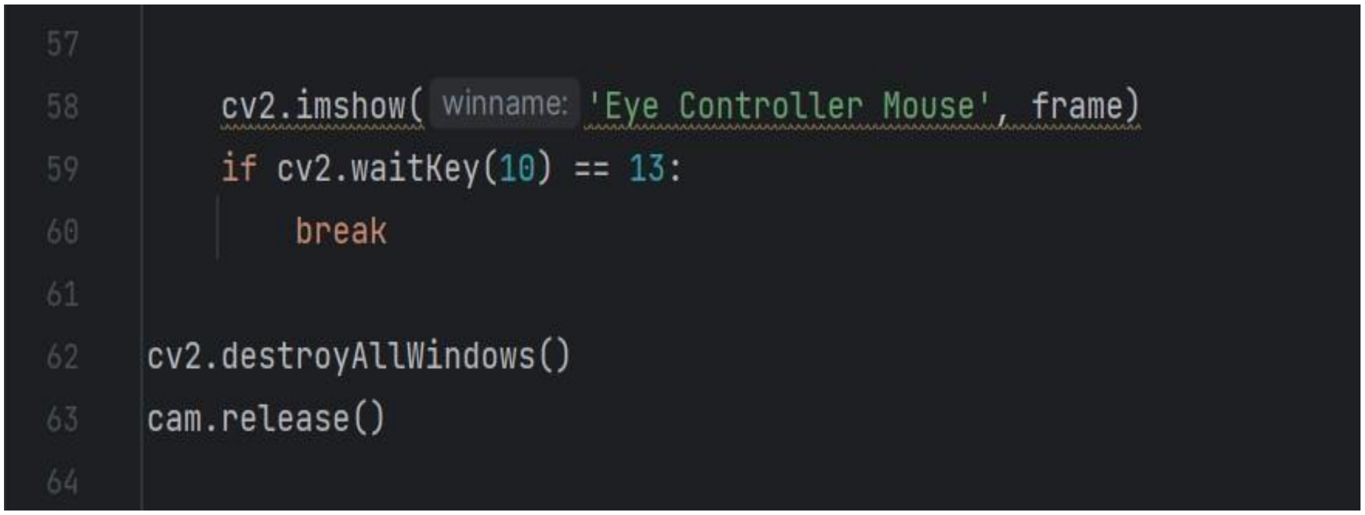
- The code calculates the screen coordinates based on specific eye landmarks and maps them to the screen resolution using np.interp().
- It moves the mouse cursor to the calculated screen coordinates using pyautogui.moveTo(), enabling cursor control based on eye movements.
- Additionally, it detects mouth gestures corresponding to blinks based on the vertical distance between mouth landmarks. If a blink gesture is detected, it simulates a mouse click using pyautogui.click().



```python
        left = [landmarks[145], landmarks[159]]
        for landmark in left:
            x = int(landmark.x * frame_w)
            y = int(landmark.y * frame_h)
            cv2.circle(frame, center: (x, y), radius: 3, color: (0, 255, 255))

        mouth = [landmarks[11], landmarks[14]]
        for landmark in mouth:
            x = int(landmark.x * frame_w)
            y = int(landmark.y * frame_h)
            cv2.circle(frame, center: (x, y), radius: 3, color: (0, 255, 255))

        if(mouth[1].y - mouth[0].y) > 0.020:
            pyautogui.click()
            pyautogui.sleep(1)
```

6. Visualization:

- The code visualizes the detected landmarks and cursor movement on the frame using cv2.circle() to draw circles around specific landmarks.

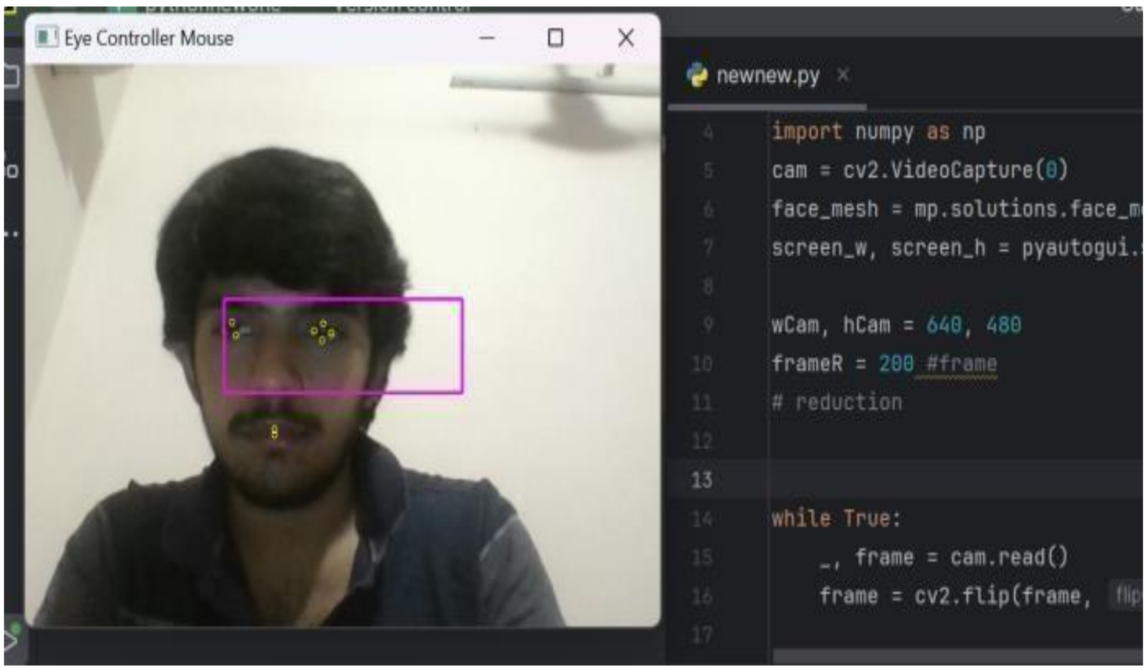


```python

    cv2.imshow( winname: 'Eye Controller Mouse', frame)
    if cv2.waitKey(10) == 13:
        break

cv2.destroyAllWindows()
cam.release()
```

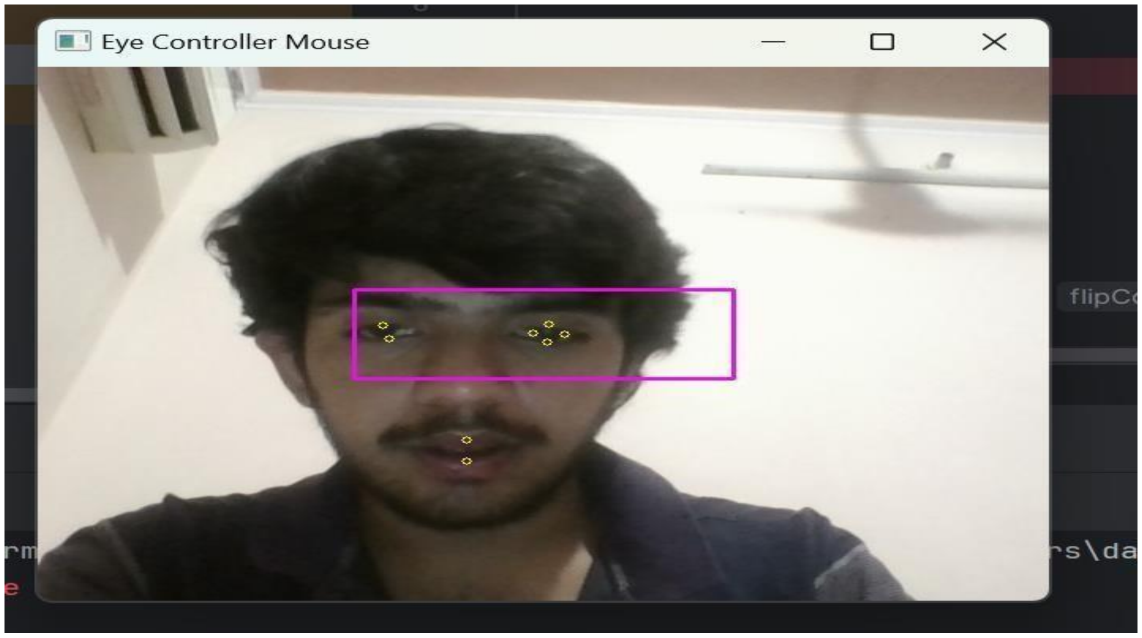
7. Display and Termination:

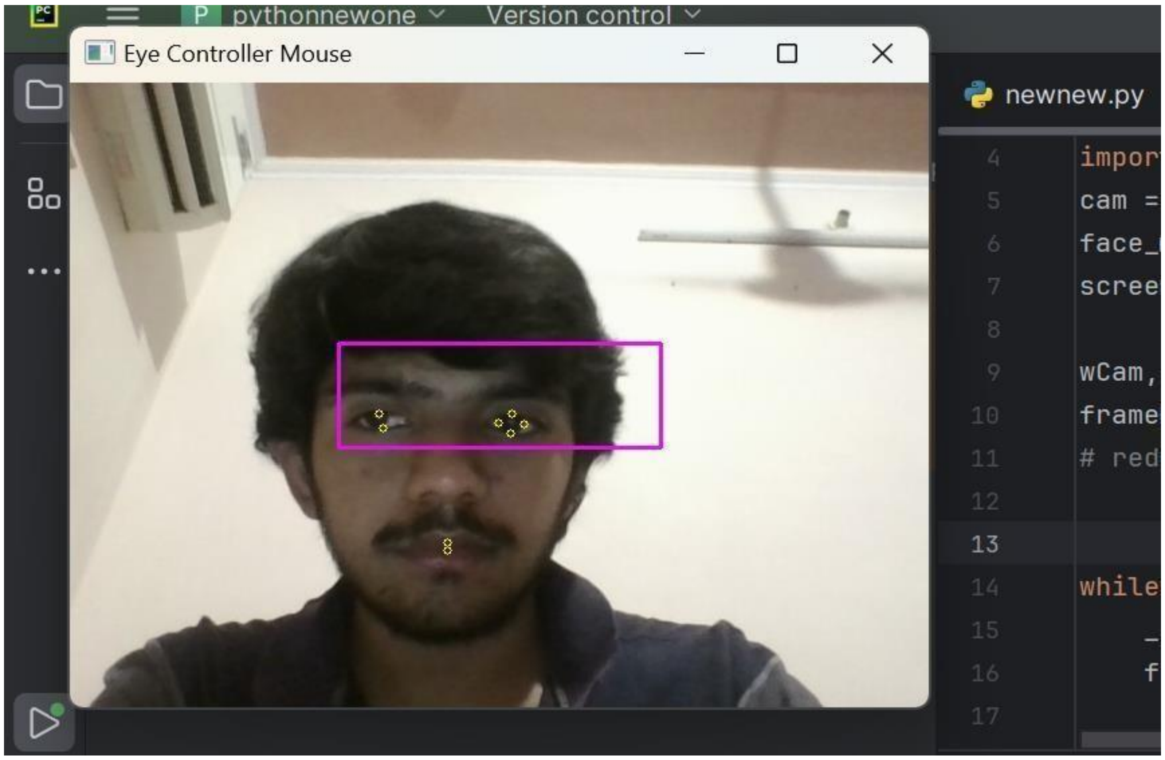
- The processed frame with visualizations is displayed in a window titled "Eye Controller Mouse" using cv2.imshow().
- The loop continues until the user presses the Enter key (keyCode 13), at which point the program exits, and the webcam is released using cv2.destroyAllWindows() and cam.release().

7.3 Output



```
import numpy as np
cam = cv2.VideoCapture(0)
face_mesh = mp.solutions.face_m
screen_w, screen_h = pyautogui.s

wCam, hCam = 640, 480
frameR = 200 #frame
# reduction


while True:
    _, frame = cam.read()
    frame = cv2.flip(frame,  flip
```





```
impor
cam =
face_
scree

wCam,
frame
# red


while
    _
    f

```

CHAPTER 8
CONCLUSION & FUTURE SCOPE

8.1 Conclusion

In conclusion, the AI virtual mouse represents a transformative advancement in human computer interaction (HCI). Through the integration of advanced artificial intelligence and computer vision technologies, it offers users an intuitive, precise, and adaptable means of interacting with digital interfaces. The virtual mouse's ability to interpret natural eye & lip gestures and continuously learn from user behaviour enables a personalized experience, enhancing efficiency and productivity across a range of tasks. The potential applications of the AI virtual mouse span from everyday computing for general users to specialized fields like graphic design and healthcare. By addressing the need for efficient, hygienic, and intuitive interaction, this technology sets the stage for a future where AIdriven interfaces play a central role in shaping how we interact with the digital world. As we continue to advance in HCI, the AI virtual mouse exemplifies the promise of AI in creating more intuitive and intelligent user centric computing experiences.

8.2 Future Scope

1. Daily use for disabled persons.
2. Accessibility and inclusivity enhancement.
3. Immersive virtual reality and augmented reality experiences.
4. Productivity boost in various work healthcare.
5. Integration into everyday devices for seamless interaction.

REFERENCES

[1] S. Shriram, ”Deep Learning-Based Real-Time AI Virtual Mouse System Using Computer Vision to Avoid COVID-19 Spread,” 25 Oct 2021.

[2] S. U. Dudhane, “Cursor control system using hand gesture recognition,” IJARCCE, vol. 2, no. 5, 2013.

[3] L. Somas, “Virtual mouse using hand gesture,” International Research Journal of Engineering and Technology (IRJET, vol. 5, no. 4, 2018.

[4] J. Katona, “A review of human–computer interaction and virtual reality research fields in cognitive InfoCommunications,” Applied Sciences, vol. 11, no. 6, p. 2646, 2021.

[5] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov, “Realtime computer vision with openCV,” Queue, vol. 10, no.4, pp. 40–56, 2012.]

[6] J. Jaya and K. Thanushkodi, “Implementation of classification system for medical images,” *European Journal of Scientific Research*, vol. 53, no. 4, pp. 561–569, 2011.

[7] L. Thomas, “Virtual mouse using hand gesture,” *International Research Journal of Engineering and Technology (IRJET*, vol. 5, no. 4, 2018.

[8] V. Bazarevsky and G. R. Fan Zhang. On-Device, MediaPipe for Real-Time Hand Tracking.

[9] J. T. Camillo Lugaresi, “MediaPipe: A Framework for Building Perception Pipelines,” 2019, https://arxiv.org/abs/1906.08172.

[10] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov, “Realtime computer vision with openCV,” *Queue*, vol. 10, no. 4, pp. 40–56, 2012.

[11] K. P. Vinay, “Cursor control using hand gestures,” *International Journal of Critical Accounting*, vol. 0975–8887, 2016.

[12] V. Bazarevsky and G. R. Fan Zhang. On-Device, MediaPipe for Real-Time Hand Tracking.

[13] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov, "Realtime computer vision with openCV," *Queue*, vol. 10, no. 4, pp. 40–56, 2012.

[14] A. Haria, A. Subramanian, N. Asokkumar, S. Poddar, and J. S. Nayak, "Hand gesture recognition for human computer interaction," *Procedia Computer Science*, vol. 115, pp. 367– 374, 2017.

[15] K. H. Shibly, S. Kumar Dey, M. A. Islam, and S. Iftekhar Showrav, "Design and development of hand gesture based virtual mouse," in *Proceedings of the 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, pp. 1– 5, Dhaka, Bangladesh, May 2019.