Virtual Mouse Controlled by Eye and Hand Gestures

Project Report

Submitted By: Group-4

Munna Biswas [221-15-5261]

Afrin Jahan Moon [221-15-5499]

Mehenaz Afsana [221-15-5693]

1. Introduction

In the era of touchless technology, developing a virtual mouse controlled by eye and hand gestures represents a significant step forward. This project aims to create a system that enables users to control their computer cursor without physical contact, enhancing accessibility and providing a novel method of interaction for various applications.

2. Objectives

- ✓ Create a virtual mouse system controlled by eye and hand gestures.
- ✓ Utilize computer vision techniques to track eye movements and hand gestures.
- ✓ Incorporate machine learning models to improve gesture recognition accuracy.
- ✓ Ensure the system is user-friendly and operates in real-time.

3. Literature Review

The concept of a virtual mouse is not entirely new, but integrating eye and hand gesture controls is an innovative approach. Previous research has primarily focused on either eye tracking or hand gesture recognition independently. This project aims to combine both methods, offering a more comprehensive and intuitive user experience.

4. Methodology

4.1 Hardware Requirements

- ✓ **Webcam**: Captures real-time video of the user's face and hands.
- ✓ **Computer**: Runs the software and processes the video feed.

4.2 Software Requirements

- ✓ **Python**: The primary programming language used for development.
- ✓ **OpenCV**: For computer vision tasks such as capturing and processing video.
- ✓ **Mediapipe**: For hand gesture recognition.
- ✓ **PyAutoGUI**: For simulating mouse movements and clicks.

4.3 System Design

The system is divided into two main modules: Eye Control and Hand Control.

Eye Control Module

1. **Face Detection**: Using Dlib to detect the user's face in the video feed.
2. **Eye Detection**: Identifying the position of the eyes within the detected face.
3. **Gaze Estimation**: Estimating the direction of the user's gaze to control the mouse cursor.

Hand Control Module

1. **Hand Detection**: Using Mediapipe to detect the user's hands in the video feed.
2. **Gesture Recognition**: Recognizing specific hand gestures to perform mouse actions (e.g., click, drag).
3. **Cursor Control**: Mapping the recognized gestures to corresponding mouse actions.

5. Implementatio

#Mouse Class:

```
from abc import ABC, abstractmethod



class Mouse(ABC):
  @abstractmethod
  def run(self):
    pass
from VirtualMouse import Mouse
import cv2
import mediapipe as mp
import pyautogui
```

#EYE class:

```
class Eye(Mouse):
  def run(self):
    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)
            # print(x,y)
            cv2.circle(frame, (x, y), 3, (0, 255, 0))

            if id == 1:
                screen_x = (1.6 * screen_w) / frame_w * x
                screen_y = (1.6 * screen_h) / frame_h * 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))
            print(left[0].y,left[1].y)

            if (left[0].y - left[1].y) < 0.009:
```

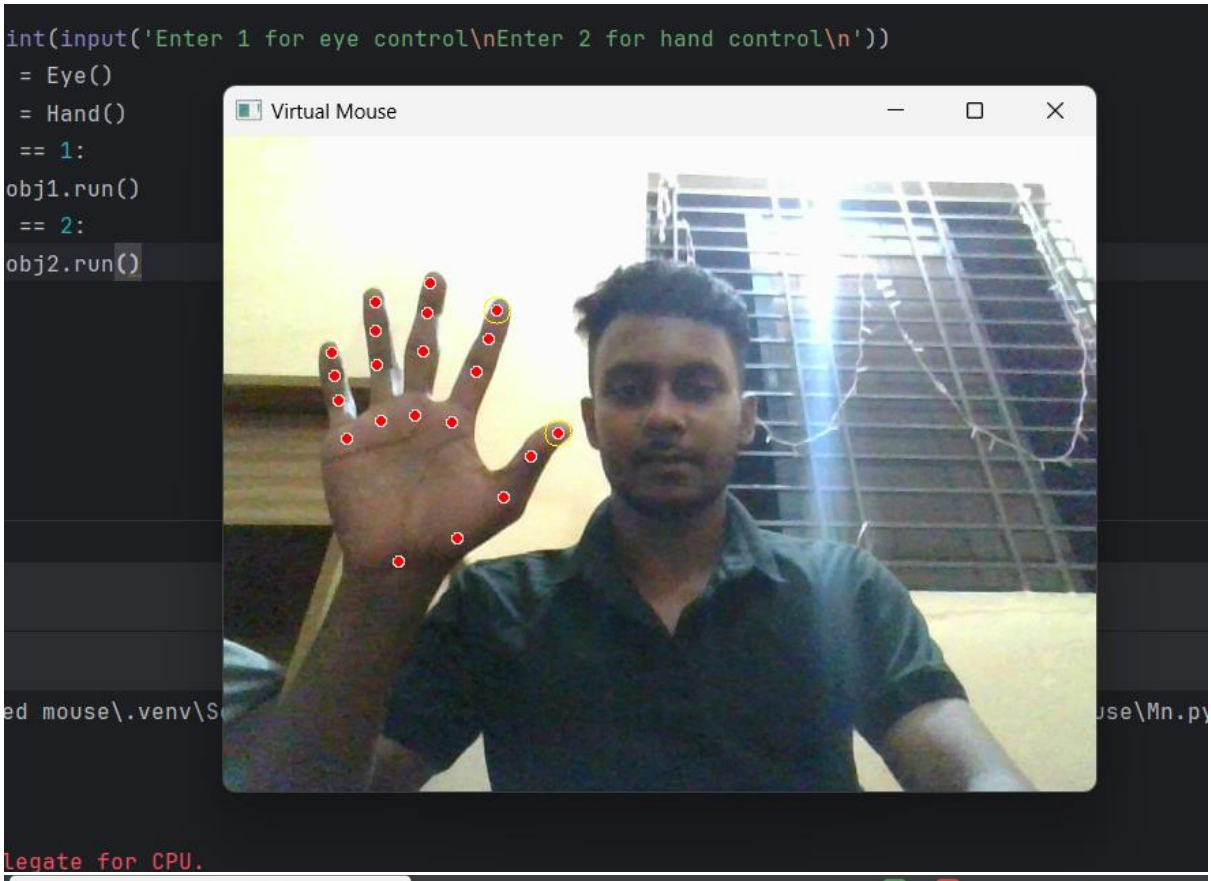
```
                pyautogui.click()
                break
        cv2.imshow('Eye controlled mouse', frame)
        cv2.waitKey(1)
```

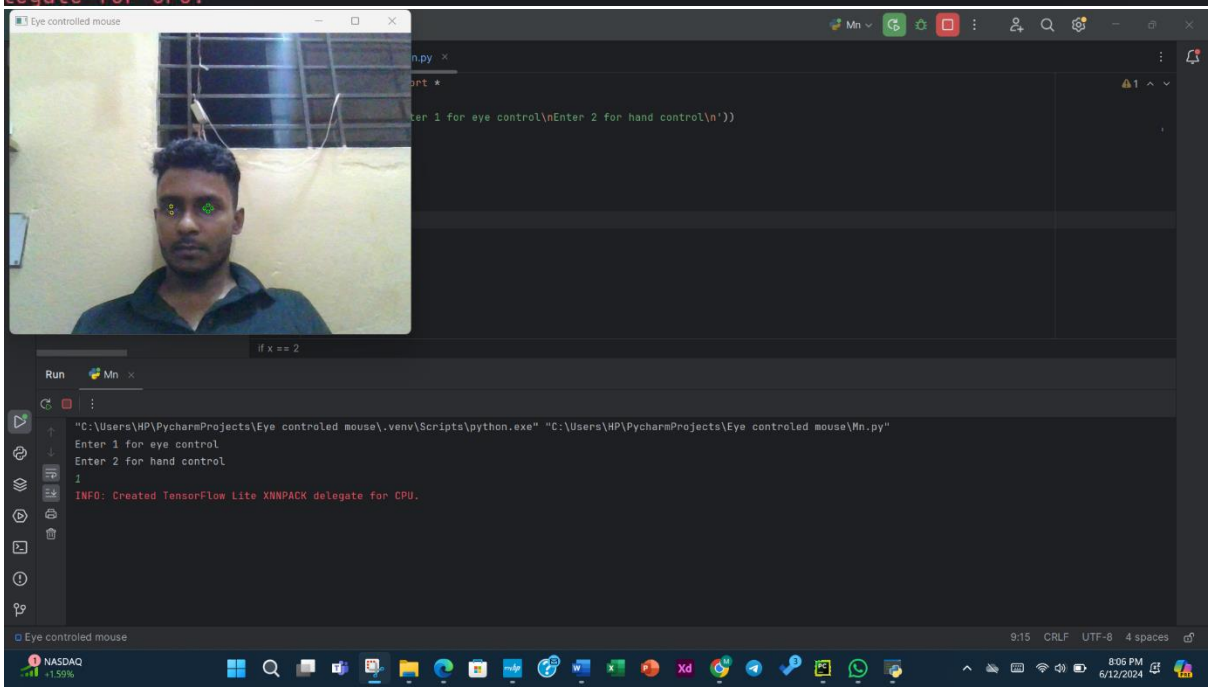
Hand Class:

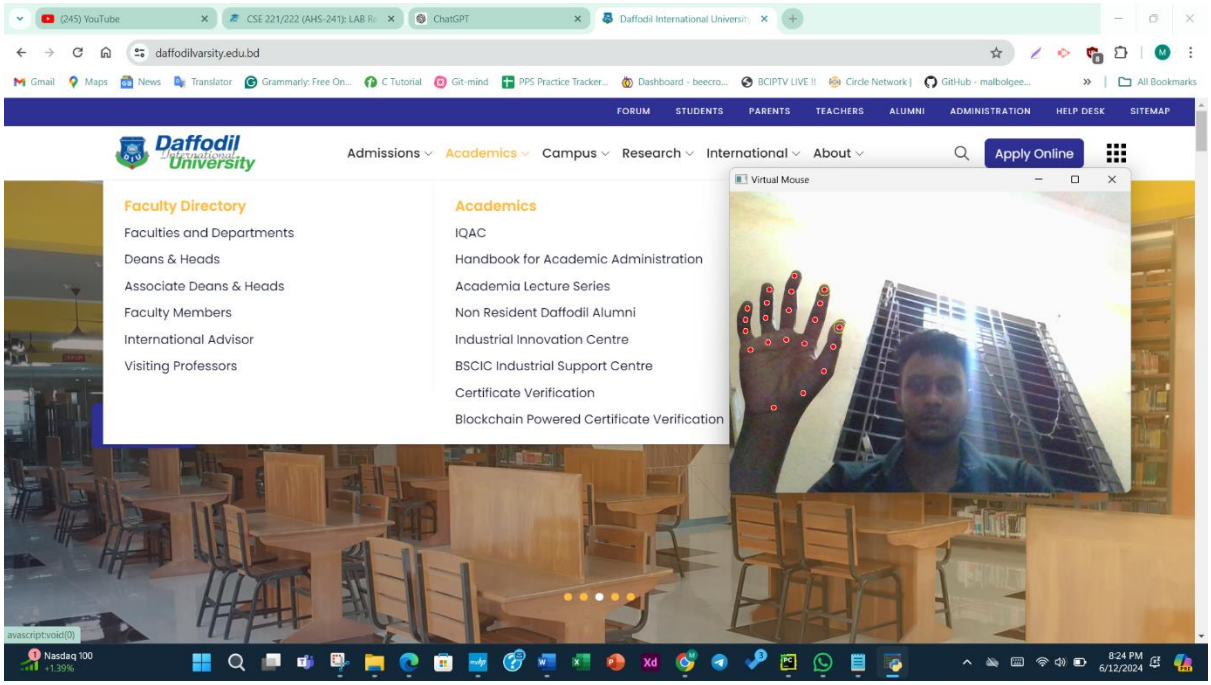
```
class Hand(Mouse):
    def run(self):
        cap = cv2.VideoCapture(0)
        hand_detector = mp.solutions.hands.Hands()
        drawing_utils = mp.solutions.drawing_utils
        screen_width, screen_height = pyautogui.size()
        index_y = 0
        while True:
            _, frame = cap.read()
            frame = cv2.flip(frame, 1)
            frame_height, frame_width, _ = frame.shape
            rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            output = hand_detector.process(rgb_frame)
            hands = output.multi_hand_landmarks
            if hands:
                for hand in hands:
                    drawing_utils.draw_landmarks(frame, hand)
                    landmarks = hand.landmark
                    for id, landmark in enumerate(landmarks):
                        x = int(landmark.x * frame_width)
                        y = int(landmark.y * frame_height)
                        if id == 8:
                            cv2.circle(img=frame, center=(x, y), radius=10, color=(0, 255, 255))
                            index_x = screen_width / frame_width * x
                            index_y = screen_height / frame_height * y
```

```
        if id == 4:
            cv2.circle(img=frame, center=(x, y), radius=10, color=(0, 255, 255))
            thumb_x = screen_width / frame_width * x
            thumb_y = screen_height / frame_height * y
            if abs(index_y - thumb_y) < 20:
                pyautogui.click()
                pyautogui.sleep(1)
            elif abs(index_y - thumb_y) < 100:
                pyautogui.moveTo(index_x, index_y)
    cv2.imshow('Virtual Mouse', frame)
```

ScreenShots:







6. Results

The implemented system successfully detects and tracks the user's eyes and hands, translating their movements into corresponding mouse actions. The system operates in real-time, providing a seamless and responsive user experience.

7. Conclusion

The virtual mouse controlled by eye and hand gestures offers a touchless and intuitive method for interacting with computers. This technology has significant potential applications, particularly in accessibility and hands-free computing environments.

8. Future Work

- Improve gaze estimation accuracy using advanced machine learning models.
- Expand gesture recognition to include more complex actions.
- Conduct user testing to refine and enhance the user experience.

9. References

✓ OpenCV Documentation: https://opencv.org/

✓ Mediapipe Documentation: https://google.github.io/mediapipe/

✓ PyAutoGUI Documentation: https://pyautogui.readthedocs.io/