Eye Controlled Mouse Using Blink and Gaze Detection

Name: [Your Name]

Mentor: [Mentor Name]

Institution: [Institution Name]

Date: [Date]

# Abstract

This project explores a hands-free human-computer interaction system using eye tracking and blink detection for mouse control. Utilizing computer vision libraries such as OpenCV and MediaPipe, the system allows users to control the mouse pointer and simulate clicks based on eye gaze and blink duration. This system benefits users with mobility impairments and also serves as an experimental human-computer interface for broader accessibility solutions.

# Table of Contents

[To be auto-generated by Word Table of Contents feature]

# 1. Introduction

With the rise in demand for touchless and accessible technology, computer vision provides opportunities for users to interact with systems in non-traditional ways. One such method is eye-controlled interaction where the eyes can move the mouse cursor, and blinks can simulate mouse clicks.  
This project demonstrates an eye-controlled mouse system built using Python, OpenCV, and MediaPipe.

# 2. Objectives

- Develop a computer vision-based system to detect eye movement and blinks.  
- Translate eye movements into mouse pointer control.  
- Use quick and long blinks to simulate mouse clicks (left click, right click, drag).  
- Provide a seamless hands-free user experience.

# 3. Technologies Used

- Python 3.x  
- OpenCV: For video processing and frame capture.  
- MediaPipe: For facial landmark detection and eye tracking.  
- PyAutoGUI: For controlling the mouse pointer and performing click actions.

# 4. System Requirements

- Python 3.10 or compatible version  
- Webcam (built-in or external)  
- Visual Studio Code or any Python IDE  
- Required libraries: opencv-python, mediapipe, pyautogui

# 5. Methodology

1. Capture real-time video feed using the webcam.  
2. Use MediaPipe to detect facial landmarks, focusing on the eye region.  
3. Calculate eye position and map it to screen coordinates.  
4. Detect blink duration to distinguish between dot (quick blink) and dash (long blink).  
5. Translate actions to cursor movement and mouse clicks using PyAutoGUI.

# 6. System Architecture

The system includes:  
- Input: Webcam capturing user face and eye movement.  
- Processing Unit: Python script using MediaPipe to extract and process eye data.  
- Output: Mouse control via PyAutoGUI based on eye gaze and blink patterns.

# 7. Implementation

- The right eye iris is used to track gaze position.  
- The left eye is used for blink detection:  
 \* Quick blink = left click  
 \* Long blink = right click  
- Dragging is done by holding a long blink followed by quick blinks.  
- Scroll functionality is added by vertical eye movement.

# 8. Results

- Accurate eye tracking under good lighting conditions.  
- Blink detection performs well with less than 0.3 seconds distinguishing between short and long blinks.  
- Mouse pointer follows gaze direction with high sensitivity.  
- Successfully simulates left click, right click, and scroll operations.

# 9. Limitations

- Performance drops in low light or with occlusions (glasses, poor camera).  
- Needs user calibration for precision.  
- Fatigue may affect performance during long usage.

# 10. Applications

- Assistive technology for differently-abled individuals.  
- Hands-free interaction in sterile environments (labs, medical rooms).  
- Experimental HCI projects.

# 11. Future Enhancements

- Add calibration interface for more accuracy.  
- Enable typing using eye movement and blink combinations.  
- Build cross-platform desktop integration.  
- Use machine learning to improve blink classification.

# 12. Conclusion

This project demonstrates a successful implementation of an eye-controlled mouse system using only a webcam and Python libraries. It offers a promising assistive technology for users with motor disabilities and opens avenues for future touchless interaction systems.

# 13. References

1. Gonzalez, R. C., & Woods, R. E. (2018). Digital Image Processing.  
2. MediaPipe Documentation, Google Research.  
3. Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement.  
4. Rosebrock, A. (2020). Practical Python and OpenCV.  
5. PyAutoGUI Documentation.