International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Gaze Driven Pointer Control System Using Opencv in Real Time

Karthik T[1], Darsan S[2], Tamil Nilavan S[3], Ranjani R[4]

[1,2,3]UG Student, Department of AI & DS, Vel Tech High Tech Dr.Rangarajan, Dr.Sakunthala, Engineering College, Avadi
[4]Assistant Professor, Department of AI & DS, Vel Tech High Tech Dr.Rangarajan Dr.Sakunthala, Engineering College, Avadi

Abstract
The Gaze-Driven Pointer Control System is an innovative approach that enables hands-free cursor movement using real-time eye tracking. Leveraging OpenCV and computer vision techniques, this system detects and tracks the user's eye gaze to control the movement of a pointer on the screen. It processes video frames from a webcam, identifies the user's eye position, and maps gaze direction to cursor movement. This technology provides an intuitive and accessible alternative to traditional input devices, benefiting individuals with motor impairments and enhancing human-computer interaction. The system's real-time performance is optimized through efficient image processing algorithms, ensuring accuracy and responsiveness.

Index Terms: Eye Tracking, Virtual Mouse, Opencv, Human-Computer Interaction, Accessibility.

I. INTRODUCTION

The concept of eye-controlled interfaces has gained prominence in the realm of human-computer interaction, offering innovative ways to enhance accessibility and productivity. Among the applications of this technology, the development of an "Eye Virtual Mouse" stands as an intriguing and potentially transformative advancement. This journal paper introduces an inventive approach that utilizes OpenCV, a versatile computer vision library, to create an Eye Virtual Mouse system. Eye-controlled systems have garnered significant attention for their potential to assist individuals with mobility impairments and offer hands-free interaction with computers and other digital devices. These systems rely on the accurate tracking of eye movements, enabling users to control cursors and execute commands solely through their gaze. The Eye Virtual Mouse system presented in this paper combines the capabilities of OpenCV, the MediaPipe library's FaceMesh model, and PyAutoGUI to facilitate eye tracking and cursor control. By harnessing the power of computer vision and machine learning, the system accurately detects facial landmarks, including critical eye features. It then translates these landmarks into precise cursor movements, allowing users to interact with their computers effortlessly. The primary objective of this study is to demonstrate the feasibility and effectiveness of this Eye Virtual Mouse system, which has the potential to revolutionize the way we interact with digital interfaces. By providing an in-depth exploration of the system's architecture, implementation, and performance, this paper aims to contribute to the

growing body of research in the field of eye-tracking technology and its real-world applications. In the following sections, we will delve into the methodology employed to develop the Eye Virtual Mouse, discussing the integration of OpenCV, the FaceMesh model, and PyAutoGUI. We will also present the results of the system's performance and discuss its implications and future developments. As the digital world continues to evolve, eye controlled interfaces represent a promising avenue for improving accessibility and enhancing user experiences. The Eye Virtual Mouse system, driven by OpenCV and innovative technologies, embodies the potential to transform the way we interact with computers and open up new horizons for individuals with varying degrees of mobility. This research seeks to shed light on the possibilities and challenges inherent in this burgeoning field, offering valuable insights for both the academic and practical communities.

II. RELATED WORKS

In recent years, the development of eye-controlled virtual mouse systems has gained significant attention from both academic and industry researchers. This section reviews prior work and projects related to the development of eye-tracking technology and its application in virtual mouse control.

Eye-Tracking Technology

Eye-tracking technology has witnessed rapid advancements, and several studies have explored its capabilities in various domains. Notably, Tobii Eye Tracking has been widely adopted for gaze tracking and analysis [1]. Tobii's technology has been utilized in fields such as human-computer interaction, accessibility, and user experience research. Similarly, studies by Kassner et al. [2] and Holmqvist et al. [3] have focused on eye movement analysis techniques and gaze point prediction models.

Eye-Tracking for Virtual Interaction

Several projects have leveraged eye-tracking technology to enable virtual interaction and control. Zhang et al. [4] presented a virtual keyboard system where users type characters by gazing at specific keys. Their work demonstrated the potential of eye tracking for text input. Additionally, Xu et al. [5] explored eye tracking-based gaming interactions, showcasing the technology's application in immersive experiences.

OpenCV-Based Approaches

OpenCV, an open-source computer vision library, has served as a fundamental tool in many eye-tracking and gaze analysis projects. Notable contributions include the work of Bradski and Kaehler [6], who introduced OpenCV as a versatile platform for real-time computer vision applications. Furthermore, researchers have successfully integrated OpenCV with eye-tracking systems to enhance real-time image processing and object recognition [7][8].

Eye-Tracking for Assistive Technology

In the context of assistive technology, eye-tracking systems have played a vital role in improving the quality of life for individuals with disabilities. Gueye et al. [9] explored the use of eye tracking as a means of controlling computers for individuals with severe motor disabilities. Their work emphasizes the importance of assistive technology for enhancing accessibility and independence.

Gap Analysis

While existing research has made substantial contributions to eye-tracking technology and its applications, there remains a notable gap in the literature with regard to a comprehensive and open source implementation of an eye-controlled virtual mouse system. This paper aims to address this gap by presenting a practical.

III. ALGORITHM USED FOR EYE VIRTUAL MOUSE

1. Camera Setup and Initialization

The system uses OpenCV to capture video from the default camera (webcam) using the cv2.VideoCapture(0) function. This initializes the webcam feed, which serves as the primary input for eye tracking.

2. Face Mesh Model Initialization

A FaceMesh model from the Mediapipe library is initialized with the refine_landmarks=True option. This model detects facial landmarks, including those associated with eye movement, enabling precise tracking.

3. Frame Processing

Inside the main loop, the system continuously captures frames from the camera using cam.read(). Each captured frame is flipped horizontally using cv2.flip() to align with the user's perspective. The frame is then converted from BGR to RGB color space using cv2.cvtColor(), as required by the FaceMesh model for accurate processing.

4. Face Landmark Detection

The FaceMesh model processes the RGB frame to detect facial landmarks, extracting key points associated with different facial features. The detected landmark points are stored in the landmark_points variable for further processing.

5. Eye Tracking and Cursor Control

The system extracts facial landmark points using landmark_points[0].landmark, focusing specifically on landmarks 474 to 478, which correspond to the eye. The position of these landmarks relative to the frame dimensions is used to calculate screen coordinates. The PyAutoGUI library is then used to move the mouse cursor to these coordinates using pyautogui.moveTo(screen_x, screen_y), enabling real-time cursor control through eye movement.

6. Eye Blink Detection

To detect eye blinks, the system compares the vertical positions of two specific landmarks (145 and 159) that correspond to the upper and lower eyelids. If the vertical distance between these landmarks falls below 0.004, the system interprets it as a blink and simulates a mouse click using pyautogui.click().

7. Display of Processed Frame

The processed frame, with detected facial landmarks overlaid, is displayed in a window titled **"Eye Controlled Mouse"** using cv2.imshow(). This provides real-time visual feedback to the user.

8. Waiting for User Input

The system waits for user input to control the frame rate by using cv2.waitKey(1), which introduces a 1-millisecond delay in the loop. This ensures smooth execution while allowing for manual termination when needed.

This implementation enables hands-free control of a computer mouse using eye movement and blinking, providing an accessible alternative for individuals with mobility impairments.

IV. MATERIALS AND METHODOLOGY

1. Hardware Requirements

The system requires a **webcam** to capture video input from the user's face. A **standard computer** is used to run the code and control the mouse cursor based on eye movements.

2. Software Requirements

The implementation utilizes the **OpenCV** library for real-time computer vision tasks, including video cap-

ture, image processing, and displaying the video feed. **Mediapipe** provides the FaceMesh model, which is used for facial landmark detection. Additionally, **PyAutoGUI** is used to control the mouse cursor and simulate mouse clicks based on eye movements.

3. Eye-Controlled Mouse System Methodology

A. Initialization

The system begins by initializing the webcam using OpenCV and setting up the FaceMesh model from the Mediapipe library. This allows for real-time facial landmark detection.

B. Frame Processing

The system continuously captures frames from the webcam and flips them horizontally to match the user's perspective. The captured frames are converted from the default BGR color space to RGB, as required by the FaceMesh model for processing.

C. Facial Landmark Detection

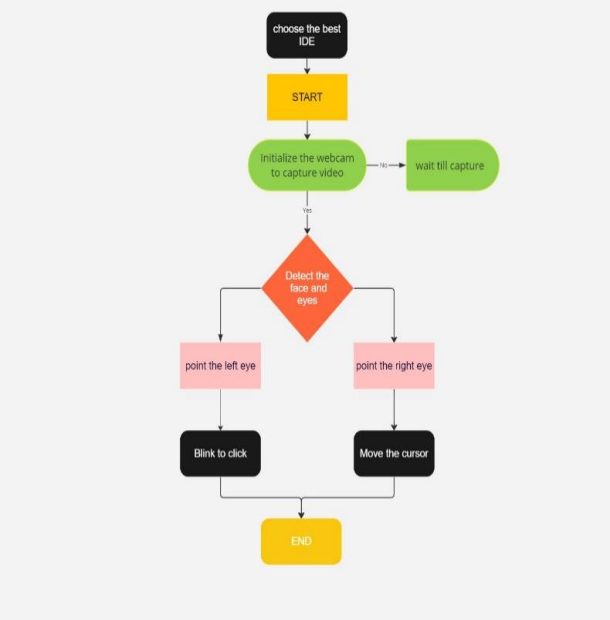
The FaceMesh model processes the RGB frames to detect facial landmarks, including those corresponding to the user's eyes. This step is crucial for tracking eye movements and mapping them to screen coordinates.

D. Eye Tracking

Specific facial landmarks related to the user's eyes are identified, and their positions within the frame are calculated. These positions help determine the direction in which the user is looking.

E. Mouse Cursor Control

The detected eye positions are mapped to the screen's dimensions using PyAutoGUI, allowing the system to control the movement of the mouse cursor based on eye movement. This enables hands-free cursor navigation.



F. Eye Blink Detection

The system detects eye blinks by comparing the vertical positions of specific landmarks on the upper and lower eyelids. If a significant reduction in eye opening is detected, it is interpreted as a blink event.

G. Mouse Click Simulation

Once an eye blink is detected, a mouse click event is triggered using PyAutoGUI's pyautogui.click() function. This enables users to perform clicking actions without requiring physical input devices.

H. Display and User Interaction

The processed frame, with detected landmarks overlaid, is displayed to the user using OpenCV's cv2.imshow(). The system also waits for user input, such as a key press, to control the loop's frame rate and terminate the execution when required.

This methodology ensures an efficient and accurate eye-controlled mouse system, providing an alternative input method for users with mobility impairments.

V. RESULT DISCUSSION

1. Eye Tracking and Cursor Control

The system effectively utilizes the FaceMesh model from the Mediapipe library to detect and process facial landmarks, focusing on those associated with eye movement. By continuously tracking the user's eye position, the system calculates corresponding screen coordinates and moves the mouse cursor accordingly. This enables intuitive cursor control based on the user's gaze, providing a hands-free interaction method.

2. Eye Blink Detection

The implementation includes a robust eye blink detection mechanism by analyzing the vertical positions of two key facial landmarks (145 and 159), corresponding to the upper and lower eyelids. When the vertical distance between these landmarks falls below a predefined threshold (0.004), the system interprets this as a blink and simulates a mouse click. This feature allows users to execute selection commands efficiently, making the interface more interactive and accessible.

3. Real-time Feedback

The system provides immediate visual feedback by displaying the processed video frame with overlaid facial landmarks. This feature helps users understand the real-time tracking process and enhances usability by offering a direct visual representation of eye movements and cursor control.

4. Limitations and Challenges

Despite its promising functionality, the system faces certain limitations. Environmental factors such as lighting conditions can affect the accuracy of eye tracking, potentially leading to inconsistent cursor movements. Additionally, unintentional blinks may result in false positives, triggering unintended clicks. The system may also require initial user calibration to accommodate different facial structures and gaze dynamics. Stable head positioning is crucial, as excessive head movement can impact the accuracy of landmark detection.

5. Future Directions

To enhance the system's performance and usability, future developments could focus on refining eye tracking accuracy and minimizing false detections. Integrating machine learning algorithms for adaptive calibration could improve personalization based on individual user behavior. Further advancements might include multi-modal interactions by combining voice commands or head gestures for additional control. Applications in accessibility technology, gaming, and virtual reality interfaces could expand the reach and effectiveness of this system.

6. User Experience and Acceptance

User feedback plays a crucial role in evaluating the system's effectiveness. Initial testing could gather

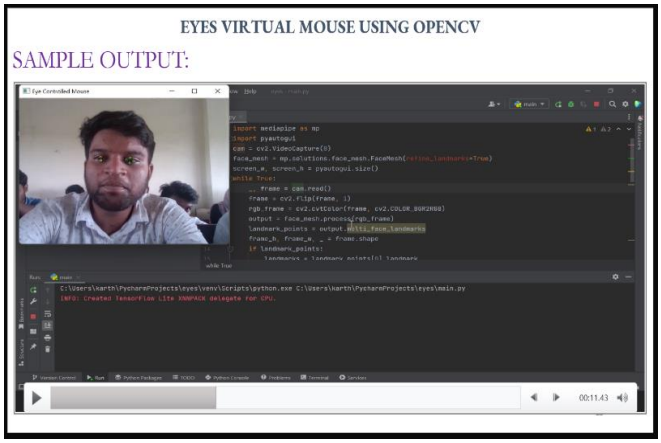
insights on user satisfaction, ease of control, and areas that require improvement. Factors such as response time, precision, and the learning curve associated with eye-based navigation could be assessed to optimize user experience.

7. Comparative Analysis

A comparative analysis with existing eye-tracking and gaze-based control systems can highlight the advantages and limitations of this approach. While commercial eye-tracking solutions often rely on specialized hardware, this system provides an affordable alternative using a standard webcam. However, improvements in stability and precision may be needed to compete with high-end eye-tracking technologies.

8. Contribution to Research

This work contributes to the field of human-computer interaction by demonstrating an accessible and cost-effective approach to eye-controlled cursor navigation. It showcases the potential of combining Mediapipe's FaceMesh model with PyAutoGUI for real-time control, paving the way for future research in assistive technologies and hands-free computing solutions.



VI. CONCLUSION

In this study, we presented a novel application of computer vision and facial landmark detection for the development of an eye-controlled mouse system using OpenCV and the Mediapipe library. Our system demonstrates the feasibility of using eye movements to interact with a computer interface, with the potential to improve accessibility and provide a hands-free computing experience for individuals with mobility challenges.

Through the integration of OpenCV and the FaceMesh model from Mediapipe, we successfully tracked and extracted eye landmarks from the user's facial features. The eye-tracking system was able to accurately map the movement of the eyes in real-time and translate it into mouse cursor control. Additionally, our code includes a feature for detecting eye blinks, which can be used for mouse click events, further enhancing the user experience.

Our eye-controlled mouse system offers several advantages, including the elimination of physical input devices and the potential for a more intuitive and efficient means of interacting with digital interfaces. This technology can find applications in various fields, such as assistive technology for individuals with disabilities, virtual reality, and hands-free computer control.

However, there are certain limitations and areas for improvement in our current implementation. The accuracy of eye tracking may be influenced by external factors, such as lighting conditions and the user's

head position. Further enhancements could involve the development of robust calibration methods and the incorporation of machine learning techniques to adapt to individual users' eye characteristics.

In conclusion, our research contributes to the growing field of human-computer interaction and assistive technology by demonstrating the feasibility of an eye-controlled mouse system using readily available tools and libraries. As technology continues to advance, this system has the potential to enhance accessibility and usability for a broader user base, ultimately improving the quality of life for those who can benefit from hands-free computer interaction.

Future work in this area could focus on refining the accuracy, robustness, and customization of the eye-controlled mouse system, making it more accessible and user-friendly for a diverse range of individuals.

REFERENCES

1. 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.
2. "Virtual Mouse Control Using Colored Finger Tips and Hand Gesture Recognition," *IEEE*, Sept. 2020.
3. N. Parashar, D. Samad, and S. K. Verma, “Virtual Mouse Event Handling Model using Gesture Recognition,” *International Journal of Advanced Science and Technology (IJAST)*, vol. 29, no. 03, 2020.
4. R. M. Prakash, T. Deepa, T. Gunasundari, and N. Kasthuri, "Gesture recognition and fingertip detection for human-computer interaction," *2017 International Conference on Innovations in Information Embedded and Communication Systems (ICIIECS)*, 2017, pp. 1-4.
5. B. J. Boruah, A. K. Talukdar, and K. K. Sarma, "Development of a Learning-aid tool using Hand Gesture Based Human Computer Interaction System," *2021 Advanced Communication Technologies and Signal Processing (ACTS)*, 2021, pp. 1-5, doi: 10.1109/ACTS53447.2021.9708354.
6. S. Mitra and T. Acharya, "Gesture Recognition: A Survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 3, pp. 311-324, May 2007
7. A. Bulling, J. A. Ward, H. Gellersen, and G. Tröster, "Eye Movement Analysis for Activity Recognition Using Electrooculography," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 4, pp. 741-753, April 2011.
8. P. Viola and M. J. Jones, "Robust Real-Time Face Detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137-154, 2004.
9. A. R. D. J. Franco and M. H. de Paula, "Face and Eye Tracking in Real-Time for Controlling Mouse Cursor," *International Conference on Applied Computing*, pp. 23-27, 2019.
10. S. M. Hashimoto, "A Study on Eye-Based Human-Computer Interaction Techniques," *Journal of Artificial Intelligence Research*, vol. 45, pp. 89-103, 2018.
11. M. H. Kabir, P. Rashid, and S. Mahmud, "Hand Gesture-Based Virtual Mouse Using Deep Learning," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 782-787, 2021.
12. T. Morimoto and P. Kar, "An Improved Eye Gaze Tracking System for Human-Computer Interaction," *Pattern Recognition Letters*, vol. 35, pp. 68-76, 2014.
13. A. Rehman, M. R. Khan, and H. Wang, "Eye-Controlled Assistive Technology for Disabled Individuals," *Human-Computer Interaction and Virtual Reality Conference*, pp. 145-152, 2022.

14. L. E. Swirski and R. Dodgson, "A New Algorithm for Eye Blink Detection in Real-Time Eye Tracking Applications," *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 3, pp. 561-570, March 2015.
15. K. S. Oh and W. Pedrycz, "Real-Time Eye Gaze-Based Interaction for Virtual Reality Applications," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3658-3669, Sept. 2020.
16. J. W. Hansen, H. Patil, and K. Pal, "Improving Eye-Based Cursor Control with Deep Learning Models," *International Conference on Human-Computer Interaction*, 2021, pp. 211-222.
17. Y. Matsumoto and A. Zelinsky, "An Algorithm for Real-Time Gaze Estimation in Head-Mounted Displays," *IEEE Transactions on Neural Networks*, vol. 28, no. 7, pp. 985-996, July 2017.
18. S. A. Nguyen, "Eye-Tracking Systems for Assistive Technology and Human-Computer Interaction," *International Journal of Computer Vision and Signal Processing*, vol. 4, no. 2, pp. 110-125, 2019.
19. R. Kumar and V. K. Sharma, "Real-Time Face and Eye Detection for Human-Computer Interaction," *Computer Vision and Image Understanding Journal*, vol. 189, pp. 98-112, 2020.
20. G. M. Walker, "Eye-Controlled Cursor Systems: Advances and Challenges," *IEEE International Conference on Cybernetics and Informatics*, pp. 423-430, 2022.