Eye-Controlled Mouse System Using Facial Landmarks and Gesture Recognition

Srudhan Eerla
MTech SCOPE
Vellore Institute of Technology
Chennai, INDIA
srudhaneerla@gmail.com

Dr. S. Deepa Nivethika
Professor
Vellore Institute of Technology
Chennai, INDIA
deepanivethika.s@vit.ac.in

***Abstract*— Human–computer interaction (HCI) has seen significant advancements in recent years, aiming to make technology more accessible and adaptive. Traditional input devices such as the mouse and keyboard are effective but can be limiting for users with physical disabilities. To address this, a vision-based mouse control system using facial landmarks and eye gestures is proposed. The system captures live video through a webcam and detects facial features using Dlib's 68-point landmark predictor. Key parameters such as the Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) are calculated in real time to identify eye blinks and mouth movements. These gestures are then mapped to corresponding mouse actions including left-click, right-click, double-click, and scroll. The proposed model is implemented using Python libraries such as OpenCV, Dlib, Imutils, NumPy, and PyAutoGUI. The system achieves reliable gesture detection under normal lighting conditions and provides a contactless, user-friendly interface. This approach offers a low-cost and effective alternative for hands-free computer control, particularly beneficial for assistive technologies and accessibility applications.**

Keywords—*Eye tracking, Facial landmarks, Human–Computer Interaction (HCI), OpenCV, Dlib, PyAutoGUI, Cursor control, Eye gestures.*

INTRODUCTION

Modern computers have become an essential part of daily life, yet the ways in which users interact with them have changed very little over the years. Devices such as the mouse and keyboard remain the most common tools for communication with computers. However, for individuals with physical impairments or limited hand movement, using these devices can be difficult or even impossible. This challenge has encouraged the development of alternative methods that allow interaction through natural human gestures such as eye blinks, facial movements, or head orientation. These systems make computing more inclusive and intuitive by removing the need for direct physical contact.

Eye-based control systems have gained increasing attention as a promising hands-free solution for operating computers. With the progress of image processing and machine learning, it is now possible to detect and analyze subtle facial features using only a standard webcam. By tracking the eyes and facial landmarks, the computer can interpret specific gestures and map them to corresponding actions. While some commercial eye-tracking devices already exist, they often rely on expensive infrared sensors or complex hardware setups, making them unsuitable for everyday users. Hence, there is a need for a low-cost, software-based solution that provides accurate and real-time gesture detection without additional equipment.

In this research, an eye-controlled mouse system is proposed using facial landmarks and eye gestures to perform cursor operations. The system utilizes Dlib's 68-point facial landmark predictor to locate the eyes and mouth in real-time video frames. The Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) are calculated to detect gestures such as blinking and mouth opening, which are then converted into mouse actions like left-click, right-click, double-click, and scroll through the PyAutoGUI library in Python. The proposed system offers a simple, reliable, and cost-effective approach to computer control. It aims to assist people with disabilities and also serve as an innovative, hands-free tool for regular users, promoting greater comfort, accessibility, and user independence in computer interaction.

RELATED WORK

Technology that enables computers to understand human gestures has become an active area of research in recent times. Instead of depending on hardware-based inputs such as the mouse or keyboard, modern systems are now being designed to respond to natural human movements, especially those of the eyes and face. This shift has been made possible by improvements in computer vision and real-time image processing techniques. Researchers across the world are focusing on making these systems more accurate, affordable, and responsive, with the goal of providing an easier and more inclusive way to interact with computers particularly for users with limited physical mobility.

Early research in this field mainly relied on geometric and rule-based methods to detect and interpret facial and eye movements. Mygapula et al. [1] developed an iris-based computer control system using OpenCV and Python that converted eye movements into cursor actions. Their model achieved over 90% accuracy and worked effectively even under varying lighting conditions. Vasisht et al. [10] proposed an eye-controlled interface using Dlib-based facial landmarks to identify eye blinks and mouth movements. They utilized the Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) to recognize gestures and perform mouse clicks in real time. Similarly, Sreeni et al. [8] built a system using Haar Cascade and Adaboost algorithms to detect facial motion, where head movement controlled cursor direction and eye blinks were interpreted as clicks. While these traditional systems provided reliable performance in controlled settings, they often struggled under poor lighting or when the user's face was partially tilted.

To address these challenges, later studies introduced hybrid methods that combined classical image processing with deep learning techniques. Saradha et al. [2] developed a hybrid system using Histogram of Oriented Gradients (HOG) for face detection and a Convolutional Neural Network (CNN) for recognizing gaze direction. This combination improved detection accuracy and stability across different lighting conditions. Kanakaprabha et al. [6] proposed a regression-assisted framework that applied the Hough Transform for iris localization and regression analysis for predicting gaze points. Miah et al. [7] presented a Mediapipe-based deep learning model that integrated CNN features with EAR-based blink detection, resulting in better accuracy and faster processing. These hybrid and deep learning systems achieved higher precision and adaptability while maintaining real-time performance.

Recent studies have shifted toward multimodal approaches that combine multiple gestures for improved interaction. Jayalakshmi et al. [9] introduced a virtual mouse system that responded to both hand and eye gestures using OpenCV, Mediapipe, and PyAutoGUI, allowing users to perform actions such as cursor movement, clicking, and volume control. Murthy et al. [5] proposed an eye gesture–based mouse that used Haar Cascade classifiers for face and eye detection, proving that standard webcams can achieve accurate gaze tracking without the need for special sensors. Likewise, Abiyev and Arslan [4] developed a CNN-driven interface that recognized both head and eye gestures, improving system responsiveness and reliability for assistive use.

From these studies, it is evident that research on eye and facial gesture-based control has evolved from simple geometric models to advanced hybrid and multimodal systems. Despite these improvements, many existing methods still depend on high computational power and perform inconsistently under varied environmental conditions. The system proposed in this paper aims to overcome these issues through a lightweight, real-time, software-based approach using Python, OpenCV, and Dlib. By leveraging natural eye blinks and mouth gestures, the model offers a cost-effective, accessible, and hands-free method of computer control suitable for all users.

METHODOLOGY

To develop the proposed system and achieve accurate, real-time control, several advanced Python libraries and computer vision techniques were employed. The system is implemented entirely through software and does not require any additional hardware apart from a standard webcam. Key libraries such as OpenCV are used for capturing and processing live video frames, while Dlib is applied for detecting and tracking facial landmarks that define the positions of the eyes and mouth. The PyAutoGUI library is then utilized to convert recognized facial gestures into corresponding mouse actions such as clicks or scrolling. All these components work together in a synchronized manner to create a smooth and contactless interaction between the user and the computer. The system's main goal is to provide a lightweight, low-cost, and accessible alternative to traditional input devices while maintaining real-time responsiveness and reliability. The complete workflow of the proposed system is shown in Figure 1.

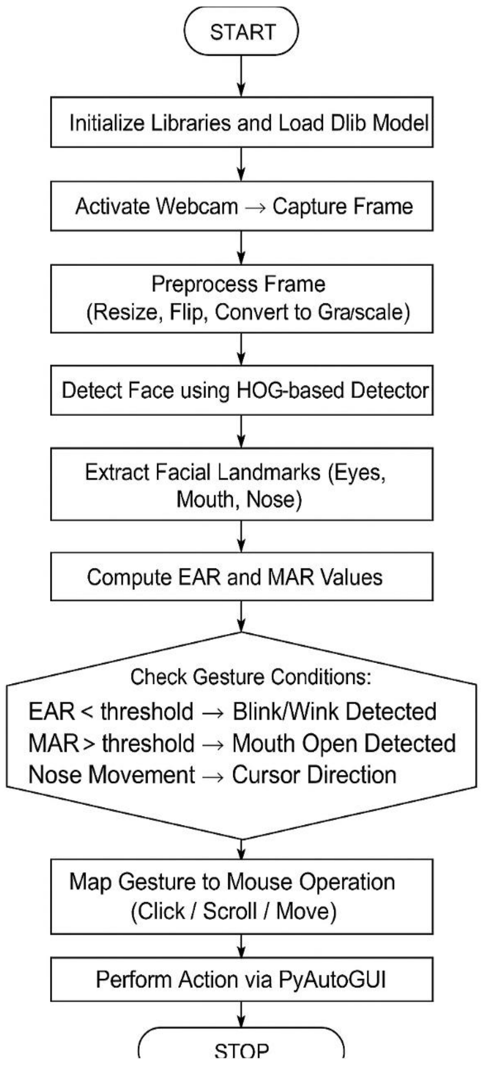
START
Initialize Libraries and Load Dlib Model
Activate Webcam → Capture Frame
Preprocess Frame (Resize, Flip, Convert to Grayscale)
Detect Face using HOG-based Detector
Extract Facial Landmarks (Eyes, Mouth, Nose)
Compute EAR and MAR Values
Check Gesture Conditions:
EAR < threshold → Blink/Wink Detected
MAR > threshold → Mouth Open Detected
Nose Movement → Cursor Direction
Map Gesture to Mouse Operation (Click / Scroll / Move)
Perform Action via PyAutoGUI
STOP

Fig. 1. Flowchart

A. *Initialization of Libraries and Loading Dlib Model*

The proposed system begins with the initialization of all required Python libraries and the loading of the pre-trained Dlib facial landmark model. This step sets up the foundation for real-time facial detection and gesture recognition. Several Python libraries are employed, each serving a distinct function in the workflow. OpenCV (cv2) is used for capturing and processing live video frames, performing essential operations such as resizing, grayscale conversion, and image enhancement. Dlib provides robust face detection and facial landmark prediction capabilities through its pre-trained 68-point landmark predictor model (shape_predictor_68_face_landmarks.dat), which is loaded at the start of the program. NumPy supports mathematical operations such as calculating Euclidean distances between facial landmark coordinates, while Imutils simplifies frequent image-processing tasks like resizing and rotation. PyAutoGUI enables the system to control the cursor and perform actions such as clicks, double-clicks, and scrolling based on detected gestures. The libraries are imported and initialized only once at the beginning to reduce computational overhead during execution. This ensures that the system operates efficiently in real time using only a standard webcam, without the need for any additional hardware or complex setup.

B. Frame capture

Once all libraries are initialized and the facial landmark model is loaded, the system activates the webcam to begin capturing live video frames from the user's face. Each frame acts as an image input that is processed individually to detect and analyze facial features in real time. The continuous video stream is handled using OpenCV, which provides efficient access to the webcam and frame-by-frame data retrieval. To improve computational performance, each captured frame is resized to a fixed dimension and converted to grayscale. Since facial detection primarily depends on structural intensity rather than color, this conversion helps reduce unnecessary data, ensuring faster and more accurate processing.

Before moving to the detection phase, the frames are also flipped horizontally to create a mirror-like view that aligns with the user's natural movement, enhancing usability. A proper frame rate is maintained to prevent lag between the actual gesture and the corresponding cursor action. The system continuously reads new frames in a loop until the program is terminated, ensuring uninterrupted tracking of the user's facial movements. Additionally, basic preprocessing techniques such as noise reduction and contrast normalization may be applied to maintain consistency under different lighting conditions.

The frame capture stage serves as the backbone of the entire system, as every subsequent module such as face detection, landmark localization, and gesture recognition relies on the quality and stability of the frames obtained here. Smooth and consistent frame capture ensures that subtle facial gestures like eye blinks or mouth openings are recognized accurately and without delay, resulting in a responsive and user-friendly interface.

C. Fece Detection

After capturing the continuous video frames from the webcam, the next essential step is detecting the user's face within each frame. Accurate face detection is the foundation Dlib's efficient face detector, the proposed approach achieves reliable, real-time face localizationestures will be analyzed. The proposed system performs several preprocessing operations before initiating face detection to enhance accuracy and consistency. Each captured frame is resized to a standard resolution to maintain uniformity and reduce computational overhead. The frame is then converted to grayscale, since color information is not required for face detection. Grayscale conversion simplifies the image by focusing on intensity variations, which helps the detection algorithm recognize edges and facial structures more efficiently. Additionally, filters such as Gaussian blur may be applied to minimize image noise and smooth out small variations that could interfere with detection under unstable lighting conditions.

Once preprocessing is completed, the system uses Dlib's frontal face detector, which employs the Histogram of Oriented Gradients (HOG) algorithm combined with a linear classifier for detecting facial regions. The HOG-based approach works by analyzing the gradients and edge orientations in the image, identifying characteristic facial features such as the outline of the eyes, nose, and mouth. This method offers strong robustness to variations in facial expression, moderate lighting changes, and small head rotations, making it suitable for real-time applications. When a face is successfully detected, a rectangular boundary is drawn around it, marking the Region of Interest (ROI) for subsequent processing. By restricting further analysis to this ROI, the system significantly reduces background interference and computational cost, improving both speed and reliability.

The face detection process is executed continuously for every frame in the video stream. As the user moves or changes position, the detector dynamically updates the ROI to ensure that the face remains accurately tracked. In case the face is momentarily lost due to occlusion or sudden movement, the system quickly reinitializes detection when the face reappears in view. This continuous detection and tracking mechanism ensures stability, allowing the system to maintain consistent performance even during real-world usage. By combining preprocessing steps with Dlib's efficient face detector, the proposed approach achieves reliable, real-time face localization laying a strong foundation for accurate facial landmark prediction in the next stage.

D. Face Landmark Detection

Once the user's face is successfully detected and marked with a bounding box, the next stage involves identifying key feature points on the face that define its structure. This process, knowlocalizingl landmark detection, plays a vital role in locating and tracking regions such as the eyes, mouth, nose, and jawlanalyze the proposed system, Dlib's pre-trained 68-point facial landmark predictor is used for this purpose. The model, stored as shape_predictor_68_face_ landmarks.dat, is based on an ensemble of regression trees and is capable of detecting specific facial landmarks with high precision. These landmarks are represented as a set of (x, y) coordinates corresponding to key locations on the user's face, enabling the system to analyze facial expressions and gestures in detail.

Before applying the landmark predictor, each detected face is slightly expanded around its bounding box to ensure that no important features fall outside the Region of Interest (ROI). The grayscale frame obtained from the preprocessing stage is passed to the predictor, which scans the ROI and outputs 68 landmark points. These points map distinct facial features such as the corners of the eyes, the eyebrows, the outline of the lips, and the shape of the jaw. For this project, only the landmarks corresponding to the eyes and mouth are utilized, as they are essential for computing the Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) that determine blinking and mouth-open gestures. Figure 2, shows the user's face marked accurately on the key facial regions.

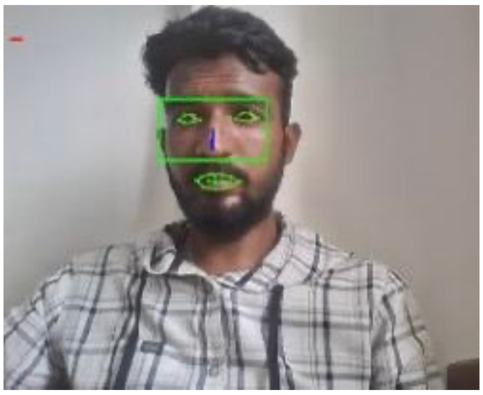

Fig. 2. Detection of facial landmarks using Dlib model

Once the landmarks are detected, they are drawn as small circular markers on the frame to provide visual confirmation during execution. The system then extracts the coordinates of the eye and mouth regions and stores them temporarily for mathematical computation in the next stage. This detection process operates continuously on each frame, adapting dynamically to the user's movement and facial orientation. Dlib's predictor offers stable landmark estimation even under minor variations in lighting and head position, making it suitable for real-time gesture recognition. By accurately localizing these feature points, the facial landmark detection stage provides a reliable geometric foundation for calculating aspect ratios and identifying user gestures with high precision.

E. ***Feature Extraction***

After the facial landmarks are detected, the next step is to extract the regions of interest (ROIs) that are relevant for gesture recognition. In this system, the focus is mainly on the eyes and mouth, as these areas carry the most expressive features required for cursor control. Each region is defined using landmark coordinates obtained from Dlib's facial landmark predictor. For the eyes, six points are selected to outline the upper and lower eyelids and the corners of each eye, while for the mouth, twelve points are chosen to represent the outer lip boundary. These coordinates form the geometric foundation for calculating the Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) in the subsequent stage of the system.

Once these coordinates are identified, the corresponding regions are extracted from the face frame using OpenCV functions. The selected areas are cropped and isolated to minimize background interference and improve processing speed. To visually represent the tracking process, contours or convex hulls are drawn around the eyes and mouth. Light preprocessing techniques such as contrast normalization or smoothing may also be applied to handle changes in illumination and maintain accuracy under varying conditions. By focusing only on these essential regions, the system reduces computational complexity and ensures that even subtle movements like blinking or mouth opening are detected accurately. This selective extraction of meaningful features strengthens both the performance and real-time responsiveness of the gesture recognition process.

F. ***Calculation od Eye and Mouth aspect ratios***

After the eyes and mouth regions are extracted from the face frame, the next step is to calculate two geometric parameters the Eye Aspect Ratio (EAR) and the Mouth Aspect Ratio (MAR). These values play a crucial role in determining the user's eye and mouth states, which are later used for gesture recognition. The EAR measures the degree of eye openness by evaluating the distance between the eyelids relative to the horizontal width of the eye. For each eye, six landmark points are used: two for the corners and four for the upper and lower eyelids. The vertical and horizontal distances between these points are calculated using the Euclidean distance formula. The EAR is then obtained using the following mathematical expression:

$$EAR = \frac{(\|p^2 - p^6\| + \|p^3 - p^5\|)}{(2 \times \|p^1 - p^4\|)}$$

Here, ||pi – pj|| represents the Euclidean distance between two facial landmark points. The numerator computes the sum of the vertical distances between the eyelids, and the denominator represents the horizontal distance between the corners of the eye. When the eyes are open, the EAR value remains high; as the eyes close, the vertical distances shrink, causing the EAR value to drop below a certain threshold. This change in EAR helps the system detect eye blinks in real time with high accuracy.

Similarly, the Mouth Aspect Ratio (MAR) is calculated to determine whether the mouth is open or closed. The MAR is computed using the vertical and horizontal distances between specific mouth landmarks extracted from the outer lip contour. The formula used is given by:

$$MAR = \frac{(\|p^{51} - p^{59}\| + \|p^{53} - p^{57}\| + \|p^{49} - p^{55}\|)}{(3 \times \|p_{49} - p_{55}\|)}$$

A high MAR value indicates that the mouth is open, while a lower value corresponds to a closed mouth. The system continuously monitors changes in MAR and compares them against a defined threshold to detect intentional mouth movements such as opening the mouth to activate scroll mode. Both EAR and MAR calculations are performed for each frame, allowing the system to track subtle changes in facial gestures accurately. Small variations caused by lighting or minor head movements are filtered out to ensure stable detection. Together, these two aspect ratios provide an efficient, non-intrusive way to interpret facial gestures for hands-free computer control.

Gesture recognition

Once the Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) values are calculated for each frame, the system interprets them to identify the user's gestures. These gestures are mapped to different mouse control actions such as left click, right click, double click, and scroll mode. To achieve this, the calculated EAR and MAR values are continuously compared against predefined threshold values. When the EAR value of the left or right eye drops below its respective threshold, the system identifies it as a blink. A left-eye blink is interpreted as a left-click action, while a right-eye blink triggers a right-click. If both eyes close simultaneously, the system performs a double-click operation. Similarly, when the MAR value rises above the mouth threshold, the system recognizes the mouth as open, which activates the scroll mode for navigating the screen vertically.

To ensure reliable performance, the system uses a frame validation mechanism to confirm gestures. Instead of reacting to every minor change in EAR or MAR, a gesture is confirmed only if the ratio remains consistent across a few consecutive frames. This prevents false detections caused by natural eye movements, facial twitches, or variations in lighting. Additionally, adaptive threshold tuning can be implemented to handle differences between users, ensuring that the system performs accurately for various face shapes and expressions. Once a gesture is recognized, a corresponding command is sent to the PyAutoGUI library, which executes the assigned mouse action instantly. This frame-by-frame analysis and threshold-based recognition make the system efficient, lightweight, and capable of real-time performance, allowing users to interact with the computer naturally without physical contact.

Cursor Control

After a facial gesture is successfully recognized, the corresponding mouse operation is executed using the PyAutoGUI library. This stage serves as the interface between the gesture recognition system and the computer's operating environment. The PyAutoGUI module allows direct control of the mouse cursor by simulating physical movements and actions such as left click, right click, double click, and scrolling. Once a gesture is identified, the system sends an appropriate command to PyAutoGUI, which performs the respective mouse action instantly. For example, a left-eye blink results in a left-click command, a right-eye blink triggers a right click, both eyes closed together initiate a double click, and mouth opening activates scroll mode. The actions are carried out with minimal delay, ensuring real-time interaction and a smooth user experience.

To maintain natural usability, the cursor control is designed to be highly responsive yet stable. The system includes a brief delay mechanism between consecutive commands to prevent unintentional repeated clicks caused by quick blinks or facial twitches. The movement of the cursor and the timing of each action are carefully synchronized with the gesture detection process to avoid overlap or missed inputs. Since the system operates entirely through software and requires only a standard webcam, it provides a cost-effective and accessible solution for hands-free computer interaction. This integration of facial gesture recognition with PyAutoGUI enables an efficient, contactless interface that can assist users with limited mobility while also offering a new, intuitive way for anyone to control a computer using simple facial expressions.

Results and Discussions

The proposed system was tested in real time to evaluate its ability to detect facial gestures and execute corresponding mouse operations accurately. During testing, the webcam successfully captured continuous video frames and processed them using the integrated OpenCV and Dlib models. The system consistently identified facial landmarks such as the eyes and mouth, confirming stable tracking and landmark localization. Figure 4.1 displays the system during this initialization phase, showing the detected face and corresponding landmarks. The output verifies that the system can read and process real-time video input effectively, laying the foundation for gesture-based cursor control.

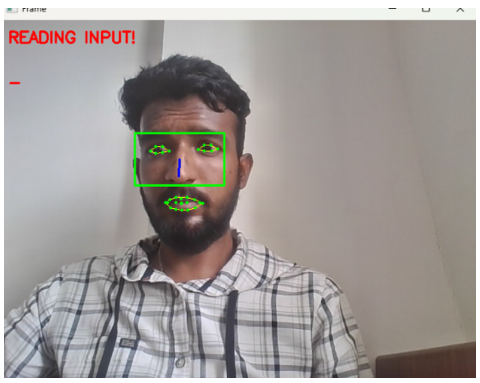

Fig. 3. Output showing system initialization

Once the face and landmarks were detected, the system's gesture recognition logic was evaluated for different facial movements. A blink of the left eye was interpreted as a left-click, while a blink of the right eye triggered a right-click. When both eyes were closed simultaneously, the system executed a double-click operation. These actions were accurately identified through variations in the Eye Aspect Ratio (EAR), which dropped below predefined thresholds during blinks. The results, shown in Figures 4.2 and 4.3, confirm that the system could detect single and double eye blinks with minimal delay and without false triggering. The response time between gesture detection and the corresponding click action was nearly instantaneous, ensuring smooth and practical usability.

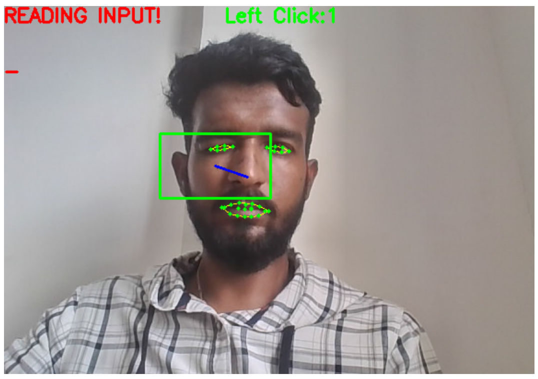

Fig. 4. Left Eye Wink for Left Click

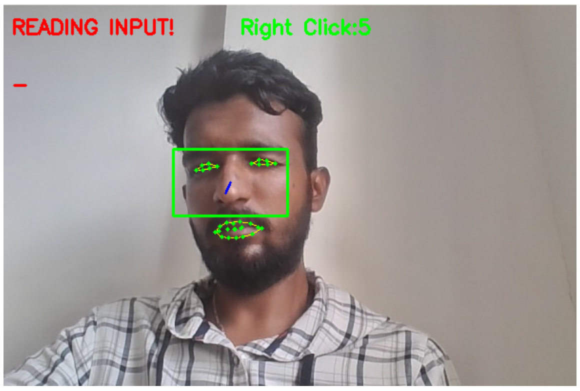

Fig. 5. Right Eye Wink for Right Click

In addition to eye gestures, mouth movement was successfully used for scroll control. When the Mouth Aspect Ratio (MAR) exceeded its threshold, the system detected the mouth as open and activated scroll mode, allowing the user to navigate vertically on the screen. Figure 4.4 illustrates this mode, showing the system's detection of an open-mouth gesture. The implementation of MAR-based scrolling was particularly useful for hands-free page navigation, further extending the usability of the proposed system. Throughout the tests, the system maintained high accuracy even under moderate changes in lighting and head orientation. Occasional performance variations occurred under poor illumination or when the user moved too rapidly, but overall stability remained consistent.

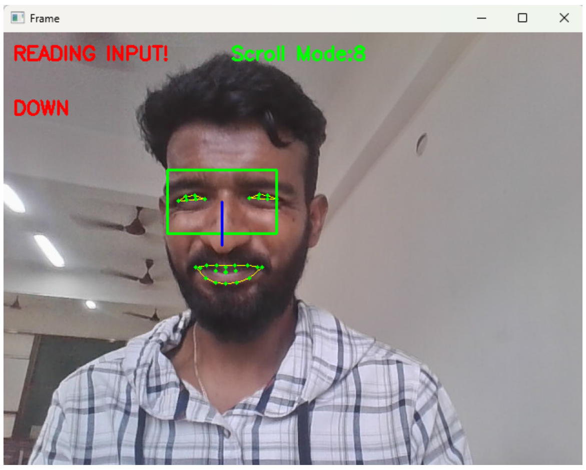

Fig. 6. Scroll mode Activated

These results demonstrate that the proposed eye-controlled mouse system performs efficiently, providing real-time responsiveness and precision without requiring specialized hardware. The combination of OpenCV, Dlib, and PyAutoGUI libraries proved effective in implementing a fully software-based solution. The EAR and MAR computations were reliable in differentiating intentional gestures from normal facial movements, minimizing false activations. Overall, the outcomes validate the system's capability as a cost-effective, accessible, and user-friendly alternative to conventional mouse devices, particularly beneficial for users with limited mobility or those seeking contactless control.

Conclusion

The proposed eye-controlled mouse system successfully demonstrates how facial landmarks and gestures can be used as an efficient and contactless alternative to conventional input devices. By integrating Python optimization such as OpenCV, Dlib, and PyAutoGUI, the system accurately detects real-time facial movements, interprets them through aspect ratio computations, and translates them into corresponding mouse actions. The implementation achieves reliable performance in recognizing blinks and mouth openings, providing smooth and responsive control without requiring any additional hardware. The results validate that simple geometric calculations combined with facial landmark tracking can create an intuitive human–computer interaction framework, particularly beneficial for individuals with physical disabilities or users seeking hands-free control.

While the current system performs effectively in normal lighting and background conditions, its accuracy can be affected by sudden illumination changes or extreme head movements. Future improvements can focus on incorporating adaptive thresholding and deep learning–based models to enhance robustness across various environments and facial orientations. Additionally, expanding the system to support more complex gestures such as drag-and-drop actions or dynamic cursor movement based on gaze direction—can further enhance usability. Integrating this technology with assistive applications, gaming, or virtual reality environments also presents promising directions for future development.

Overall, the study highlights the potential of vision-based computing as a low-cost, software-driven, and accessible solution for achieving natural and contactless human–computer interaction. With continued optimization and adaptability, such systems can significantly improve digital accessibility and reshape the way users interact with computers in the near future.

REFERENCES

[1] S. P. K. Mygapula, M. L. Saini, C. S. R. Dheeraj, S. Maji and D. Gupta, "Controlling Mouse Cursor through Eye Movement," 2024 International Conference on Signal Processing and Advance Research in Computing (SPARC), Lucknow, India, 2024, pp. 1–5, doi: 10.1109/SPARC61891.2024.10828631.

[2] S. K. R., P. R., A. N. S. and M. Rathishree, "Cursor Control Based on Eyeball Movement Using Deep Learning," 2023 Intelligent Computing and Control for Engineering and Business Systems (ICCEBS), Chennai, India, 2023, pp. 1–5, doi: 10.1109/ICCEBS58601.2023.10448773.

[3] R. H. Abiyev and M. Arslan, "Head Mouse Control System for People with Disabilities," IEEE Access, 2019.

[4] V. S. Vasisht, S. Joshi, S. Shashidhar, S. Shreedhar and C. Gururaj, "Human Computer Interaction Based Eye Controlled Mouse," Proceedings of the Third International

Conference on Electronics Communication and Aerospace Technology (ICECA), 2019.

[5] M. Jayalakshmi, T. Pardha Saradhi, S. M. Rahil Azam, S. Durga Sai Sriram and S. Fazil, “Multi-model Human-Computer Interaction System with Hand Gesture and Eye Gesture Control,” 2024 5th International Conference on Innovative Trends in Information Technology (ICITIIT), 2024.

[6] S. Kanakaprabha, A. Prathibha, R. Priyanka and K. Vijay, “Mouse Cursor Controlled by Eye Movement for Individuals with Disabilities,” Proceedings of the 7th International Conference on Intelligent Computing and Control Systems (ICICCS), 2023.

[7] P. Miah, M. R. Gulshan and N. Jahan, “Mouse Cursor Movement and Control using Eye Gaze – A Human Computer Interaction,” 2022 International Conference on Artificial Intelligence of Things (ICAIoT), 2022.

[8] S. Sreeni, M. Sabeel, E. Sai Kumar, V. H. Vardhan and K. Chandrakala, “Mouse Cursor Control Using Facial Movements – An HCI Application,” International Journal of Techno-Engineering (IJTE), vol. XV, no. II, pp. 270–273, Apr. 2023.

[9] M. S. Murthy, D. Anitha, N. M. Chowdary and S. S. S. Kumar, “Mouse Cursor Control with Eye Gestures,” IEEE International Conference on Inventive Computation Technologies (ICICT), 2024.

[10] M. Dhanaraju, S. Mekala, A. H. V. Rao, C. P. Kumar and R. Lokesh, “Human-Eye Controlled Virtual Mouse,” International Journal for Research in Applied Science and Engineering Technology (IJRASET), vol. 10, no. VI, pp. 1100–1106, Jun. 2022.