CursorPilot: Eye and Head Movement Based Cursor System

Maanav Anand Kumar - 116437327

Abstract

This report presents *CursorPilot*, a hands-free interaction system that uses eye and head movements to control a computer cursor. Designed for enhanced accessibility and improved multitasking, the system provides features such as eye-controlled cursor movement and head-tilt-based scrolling and video playback. Implemented using technologies like OpenCV, DLib, and PythonAutoGUI, *CursorPilot* offers an intuitive graphical user interface for easy mode switching and control. The system aims to reduce reliance on traditional input devices, making digital interactions more inclusive and ergonomic for users with physical limitations and professionals managing complex workflows.

1 Introduction and Motivation

The way we interact with computers has evolved significantly over the years, transitioning from basic text-based interfaces to graphical user interfaces (GUIs) controlled by devices such as mice and keyboards. While these methods have become the standard, they present challenges for individuals with mobility impairments and professionals engaged in multitasking. For over 1 billion people globally living with disabilities, traditional input devices often act as barriers to effective computer usage. Similarly, professionals in fast-paced environments face ergonomic strain and inefficiencies when relying solely on manual input devices.

CursorPilot aims to address these challenges by introducing a hands-free interaction system that uses eye and head movements for controlling a computer cursor and performing other tasks. The project leverages advances in computer vision and facial landmark detection to implement features such as cursor movement through eye gestures and page scrolling or video control via head tilts. These innovations aim to enhance accessibility for users with physical limitations while providing an ergonomic alternative for multitasking professionals.

The core motivation for this project stems from the need to make technology more inclusive and efficient. By utilizing accessible hardware such as webcams and leveraging open-source technologies like OpenCV and DLib, *CursorPilot* strives to democratize hands-free computing.

In addition to accessibility, this project emphasizes ease of use and adaptability. A simple graphical user interface (GUI) was developed to allow users to switch seamlessly between different modes of operation. This ensures that the system is not only functional but also intuitive for a wide range of users. By reducing reliance on traditional input devices, *CursorPilot* provides a gateway to inclusive computing and represents a step toward innovative interaction systems tailored for diverse user needs.

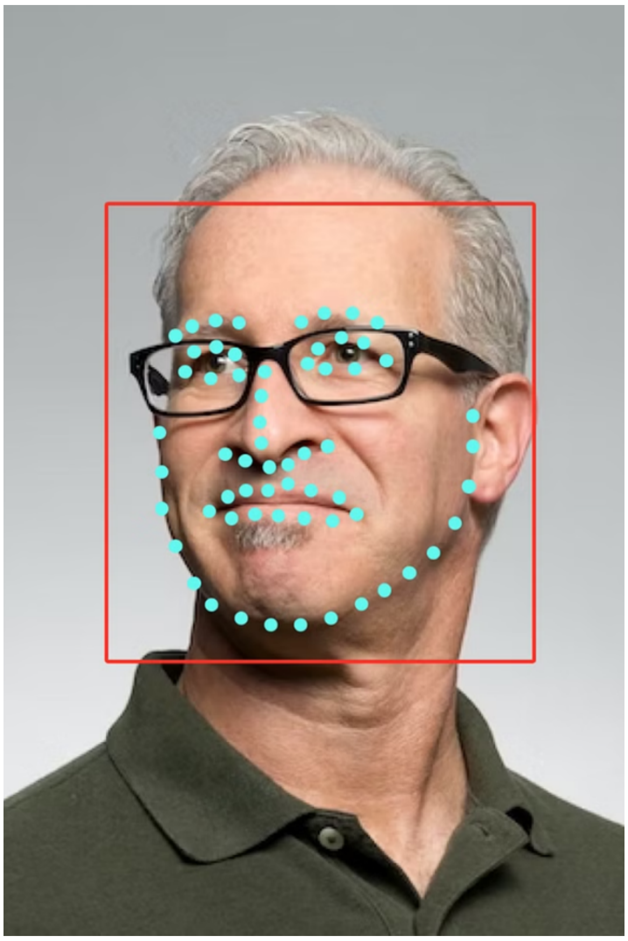

Figure 1: Facial Mapping example [10]

2 Related Work

The development of hands-free computer interaction systems has seen significant advancements, leveraging technologies like eye-tracking, head movement detection, and combined gesture-based control. These systems aim to enhance accessibility and improve user experience for individuals with physical limitations and multitasking professionals.

Eye-tracking technologies, such as the PRC's Look Eye Tracking System [1] and Tobii Dynavox PCEye [2], provide users with the ability to control devices using their gaze. These systems offer features like cursor movement, calibration options, and compatibility with accessories like glasses or contact lenses, making them highly accessible to individuals with disabilities. They enable tasks such as web browsing and document creation, proving instrumental in improving digital inclusivity.

Head movement detection solutions, like NuPoint by PRC [3] and the TrackerPro 2 Head Mouse [4], utilize optical sensors or reflective dots to translate head movements into cursor control. These technologies offer hands-free operation, providing ergonomic alternatives for users with limited hand functionality.

Innovative systems combining eye and head movements, such as the AsTeRICS Platform [5], integrate multiple input modalities to deliver versatile control. This platform employs low-cost webcams and infrared sensors, making it accessible for users with motor disabilities. Similarly, approaches like "Operating Computer Cursor Using Eye and Face Movements" [6] utilize facial landmarks detected by tools like OpenCV and DLib to interpret gestures, offering an efficient, open-source alternative.

AI-enhanced systems represent a cutting-edge approach in this domain. AI-based eye gesture control [7] and the use of Graph Convolutional Networks (GCNs) for gaze recognition [8] demonstrate high accuracy and reliability, pushing the boundaries of human-computer interaction. These advancements pave the way for more robust and adaptive solutions.

While these systems exhibit impressive functionality, challenges such as the "Midas Touch Problem" in eye tracking [9] and the cost of specialized hardware remains. Recent studies propose multimodal approaches and software-based solutions to overcome these limitations, offering promising directions for further innovation.

The *CursorPilot* project builds upon these advancements by combining the strengths of eye and head movement detection with an intuitive graphical interface. By utilizing open-source tools and accessible hardware, it addresses existing challenges while contributing to the broader goal of inclusive and ergonomic computing.

3 Technical Implementation

The head-based cursor control system and the head-tilt video and scrolling control system are both technically implemented in detail in this section. Through the combination of head movement tracking and facial landmark detection, the system enables users to interact with video content and control a cursor without the need for physical input devices.

3.1 Head-Based Cursor Control

The user's head movements are converted into real-time cursor movements by the head-based cursor control system. It tracks the user's face and determines the proper cursor location on the screen using facial landmark detection techniques.

3.1.1 Face Detection and Landmark Extraction

The core of the face detection and landmark extraction process is based on `dlib`, a toolkit that provides robust face detection and facial landmark prediction.

1. **Detection of Face:** Finding the user's face in the webcam feed is the system's first step. To swiftly detect a face in a picture, `dlib` use either a *Haar Cascade Classifier* or a *Histogram of Oriented Gradients (HOG)* based detector. The face detector can precisely locate the face in a variety of orientations and lighting situations because it has been trained on a sizable dataset of human faces. The extraction of facial landmarks: Key spots or *landmarks* on the user's face are extracted after the face has been identified. A pre-trained shape predictor model that recognizes 68 facial landmarks is provided by `dlib`. These landmarks include positions around the eyes, nose, mouth, and jawline, which are critical for tracking head movements. The model is based on a regression tree and allows for high accuracy in real-time applications.

3.1.2 Cursor Movement Calculation

The cursor movement is driven by the position of the user's head. The face landmarks extracted from the previous step are used to calculate the movement direction.

1. **Calculation of head position:** By averaging the coordinates of important landmarks like the eyes, nose, and chin, the system determines the user's face's center. The face's location in the 2D plane is accurately depicted by these points.
2. **Coordinate Mapping:** Screen coordinates are linked to the user's facial coordinates. Scaling the face's position (from the webcam frame) to the screen resolution is what this entails. For example, a scaling factor that accounts for the screen resolution is used to convert the detected face position, which is at `(x, y)` in the frame, to a screen position, `(scaled_x, scaled_y)`.
3. **Cursor Movement:** Using the `pyautogui.moveTo()` function, the system moves the cursor to the calculated screen position. To avoid jitter, the system uses a low-pass filter to smooth out small, erratic movements, only updating the cursor if the detected movement exceeds a certain threshold.

3.1.3 Smooth Cursor Control

The system employs a *low-pass filter* to prevent jitter or sudden movements of the cursor. By averaging the coordinates over time, this method smoothes out quick, slight variations in the user's facial position. By doing this, it makes sure that the cursor is only controlled by large, deliberate head movements and ignores smaller ones, like blinking.

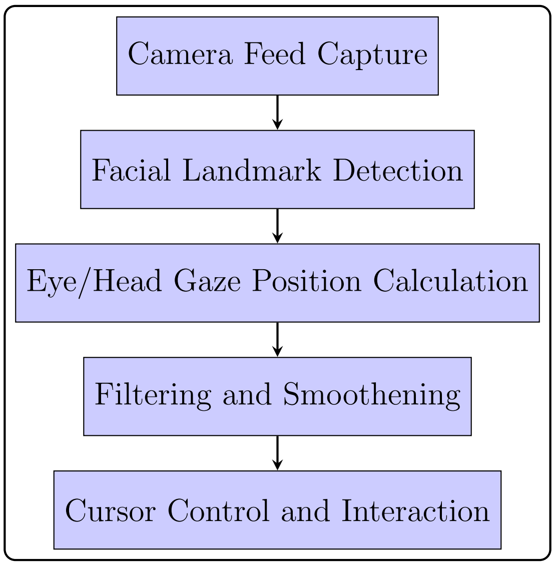
Camera Feed Capture
Facial Landmark Detection
Eye/Head Gaze Position Calculation
Filtering and Smoothening
Cursor Control and Interaction

Figure 2: Architecture of the System

3.2 Head-Tilt Based Scrolling and Video Control

This component of the system uses the head-tilt to control video playback and scrolling, providing a more natural way for users to interact with content.

3.2.1 Tilt Angle Calculation

Head-tilt is detected by monitoring the relative positions of the nose and chin landmarks. The system calculates the angle of tilt by comparing the vertical displacement between these two key points.

1. **Tilt Detection:** The tilt angle is derived by calculating the difference in the vertical positions of the nose and chin landmarks. If the chin moves to the LHS relative to the nose, the system interprets this as a left-ward tilt, and vice versa for an right-ward tilt.
2. **Angle Calculation:** The tilt angle is calculated using basic trigonometry. By computing the arctangent of the vertical displacement (difference in y-coordinates) over the horizontal distance between the nose and chin, the system calculates the tilt angle.
3. **Thresholds for Action:** The tilt angle is continuously monitored and, if it exceeds a predefined threshold (either positive or negative), a corresponding action is triggered, such as scrolling or controlling video playback.

3.2.2 Scrolling and Video Control Actions

Once the tilt angle is detected, the system interprets the movement to trigger either scrolling or video control actions.

- **Scrolling:** The system can detect the tilt direction (up or down) and trigger scrolling actions using `pyautogui.scroll()`. A downward head tilt triggers a downward scroll, while an upward tilt scrolls the page up.
- **Video Control:** If the system detects that a video player (such as a browser window with a video) is active, the tilt can control playback actions. For example, a downward tilt could pause or play the video, while an upward tilt might fast-forward or rewind the video.

The `pygetwindow` library is used to detect whether a video player is currently focused, enabling the system to trigger appropriate actions only when a video is present.

3.2.3 Thresholds and Sensitivity

It is possible to modify the tilt detection system's sensitivity. Users can set specific thresholds that define when a tilt is significant enough to trigger an action. By doing this, undesired behavior is prevented from being triggered by inadvertent or minor motions.

Furthermore, the cursor and video control actions' speed and accuracy are adjusted to provide a balance between user comfort and responsiveness. It is possible to adjust the sensitivity settings to suit various user tastes and situations.

3.3 Libraries and Tools Used

The implementation relies on several powerful Python libraries that work together to enable facial landmark tracking, cursor control, and video interaction.

- **OpenCV:** OpenCV is used for capturing real-time video from the webcam. It provides efficient image processing capabilities, which are crucial for face detection and landmark extraction.
- **dlib:** dlib is a key library for face detection and landmark extraction. Its `shape_predictor()` function is used to detect 68 facial landmarks that are critical for tracking head movements.
- **pyautogui:** This library is used to simulate mouse and keyboard events. It moves the cursor to the calculated screen coordinates and handles scrolling actions based on head movements.
- **pygetwindow:** Used to detect the currently active window on the screen, helping to determine if a video player is in focus so that it can respond to tilt-based controls.

3.4 Visual Feedback and Testing

Giving the user visual feedback is a crucial part of the system. The user can view how their head motions are being tracked by the system by superimposing the identified face landmarks on the camera video.

Users can better interact with the system by adjusting their head movements with the help of this real-time input. To enhance the user experience, visual cues can also be shown in response to particular activities (such scrolling or video control).

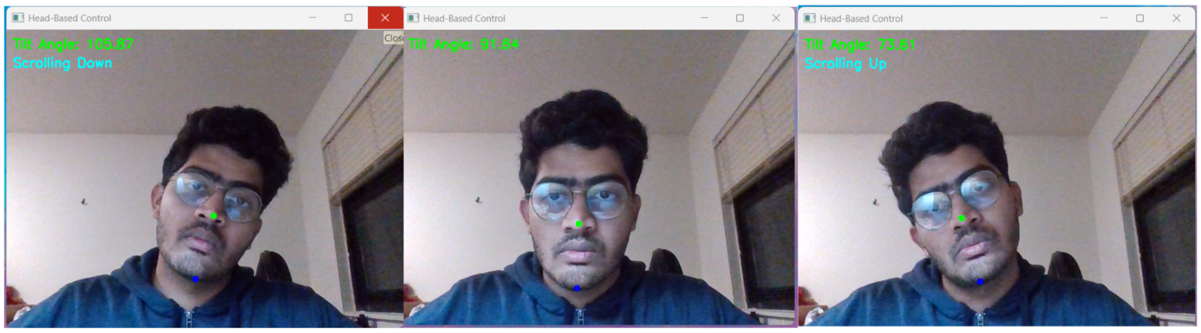

Figure 3: Real-time facial landmark tracking in action. The landmarks are overlaid on the user's face for feedback.

3.5 Performance Considerations

The system is made to function with the least amount of latency possible in real-time. Since tilt detection and facial landmark extraction require a lot of work, multi-threading strategies and effective code optimizations are employed to manage video processing and user interaction at the same time.

- **Real-time Processing:** Video frames are processed at a rate of approximately 30 frames per second (FPS), ensuring that user interactions feel natural and responsive.
- **Low Latency:** To reduce lag, the system can perform operations in separate threads, allowing for face detection and cursor control to run independently of the video feed capture.

4 Validation

The validation of *CursorPilot* involved conducting a simple user test to evaluate its core functionalities and identify areas for improvement. Additionally, a plan for a comprehensive user study is proposed to further validate the system's performance and usability.

4.1 User Test

A single-user test was conducted to validate the core functionalities of the system. The test was designed to evaluate two key features: cursor movement and scroll/video playback control. The testing methodology, parameters, and results are outlined below.

Testing Methodology: The test consisted of two tasks:
1. Moving the cursor to specific points on the screen using head gestures.
2. Controlling a web page scroll and video playback through head tilts.

Each task was performed 20 times, and the system's performance was assessed based on precision, response time, and user-reported ease of use.

Parameters and Results:
1. **Cursor Movement:** - Average precision: 75% (measured as successful alignment with target points on the screen).
- Average response time: 0.4 seconds per movement.
- Observation: Cursor movement was slightly jittery during fine adjustments, indicating the need for enhanced filtering.
- User feedback: Movement felt intuitive but required more stability for precision tasks.

2. **Scroll and Video Playback Control:**
- Gesture recognition accuracy: 95% (correct detection of intended head tilts).
- Average response time: 0.25 seconds per action.
- Observation: Both scrolling and video playback control were smooth and reliable, with no significant errors observed.
- User feedback: Gestures were easy to perform, and system responses felt seamless.

The test demonstrated the system's overall reliability while highlighting areas for improvement, particularly in stabilizing cursor movement.

Parameters and Results:

Feature	**Metric**	**Average Value**
Cursor Movement	Precision (%)	75
	Response Time (s)	0.4
Scroll/Video Control	Gesture Accuracy (%)	95
	Response Time (s)	0.25

Table 1: Summary of User Test Results

The test demonstrated the system's overall reliability while highlighting areas for improvement, particularly in stabilizing cursor movement.

4.2 Proposed Comprehensive User Study

To validate the system further, a multi-user study is proposed. This study will involve participants from diverse backgrounds, including individuals with mobility impairments. The proposed methodology includes:
- **Tasks:** Similar to the initial test, participants will perform cursor movement, scrolling, and video playback tasks.
- **Metrics:** Precision, gesture recognition accuracy, response time, and qualitative feedback on ease of use.
- **Sample Size:** A group of 15-20 participants to ensure statistically meaningful results.

This study aims to identify user-specific challenges and refine the system to cater to a broader audience effectively.

Based on quantitative results from initial testing, *CursorPilot* demonstrates its potential as a reliable and inclusive interaction system. Future iterations should focus on addressing the observed limitations and enhancing user experience.

5 Discussion and Future Work

5.1 Discussion

The development of *CursorPilot* highlighted several strengths and areas for improvement, providing valuable insights into hands-free interaction systems.

Strengths:

- The device effectively illustrates that eye and head movements can be used for natural computer interaction.
- The control features for the scroll and video playback are incredibly precise and dependable, offering seamless user experiences.
- Cost-effectiveness and wide use are guaranteed by the use of open-source technologies like OpenCV and DLib, as well as accessible hardware like common webcams.
- By making mode switching easier, the graphical user interface (GUI) improves the system's usability and accessibility in general.

Weaknesses:

- The cursor movement is a little jerky/jittery, especially when making small adjustments, which makes it less useful for activities requiring precision.
- The system is currently less usable in a variety of settings because it needs a controlled atmosphere with sufficient lighting to function at its best.
- Only a small number of users(1 user) participated in the validation procedure, which limits how broadly the findings may be applied.

5.2 Future Work

Although *CursorPilot* demonstrates the possibilities of hands-free interface systems, there are a number of ways to improve its usability and functionality:

1. **Enhanced Cursor Stability:** to decrease jitter and increase accuracy, sophisticated filtering methods like Kalman or complementing filters might be used.
2. **Calibration:** Adding a calibration module to modify system sensitivity according to user choices and environmental variables is known as adaptive calibration.
3. **Expanded Library of Gestures:** Adding more motions to improve functionality and control choices, including head nods or double blinks.

CursorPilot can develop into a more robust, adaptable, and widely available hands-free interaction system with these improvements.

6 Conclusion

CursorPilot demonstrates the potential of hands-free computer interaction using eye and head movements. It offers reliable features like cursor control and scroll/video playback, supported by accessible technologies such as OpenCV and DLib. The system's intuitive graphical user interface ensures ease of use, making it accessible to a wide range of users, including those with mobility limitations.

CursorPilot provides a strong foundation for inclusive and innovative digital interaction systems, bridging the gap between technology and accessibility.

References

[1] https://www.prentrom.com/prc_advantage/eye-tracking-and-access-technologies

[2] https://www.northwestergo.com/blog/eye-trackers-head-pointers-and-hands-free-mous

[3] https://www.prentrom.com/prc_advantage/eye-tracking-and-access-technologies

[4] https://www.boia.org/blog/understanding-assistive-technologies-what-is-a-head-mou

[5] https://project.asterics.eu/fileadmin/user_upload/Thesis_Yat-sing%20Yeung_final.pdf

[6] https://oaji.net/articles/2021/2698-1636359993.pdf

[7] https://f1000research.com/articles/13-109

[8] https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2021.709952/full

[9] https://tangemicioglu.com/files/papers/Tongue_Gestures_CHI_2023_Interactivity.pdf

[10] https://miro.medium.com/v2/resize:fit:828/format:webp/0*zoIjENJF2QpvmDQN.png