BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

Facial Gesture-Based Mouse Control

Abdoul Madjid Marou Garba
200041254
Diarrassouba Brahima
200041262

Department of Computer Science and Engineering
Islamic University of Technology
April, 2025

Declaration of Candidate

This is to certify that the work presented in this thesis is the outcome of the analysis and experiments carried out by **Abdoul Madjid Marou Garba** and **Diarrassouba Brahima** under the supervision of **Prof. Dr. Hasan Mahmud**, Professor, Department of Computer Science and Engineering and co-supervision of **Mueeze Al Mushabbir**, Lecturer, Department of Computer Science and Engineering, Islamic University of Technology, Dhaka, Bangladesh. It is also declared that neither this thesis nor any part of it has been submitted anywhere else for any degree or diploma. Information derived from the published and unpublished work of others have been acknowledged in the text and a list of references is given.

Prof. Dr. Hasan Mahmud
Professor
Department of Computer Science and Engineering
Islamic University of Technology (IUT)
Date: April 30, 2025

Abdoul Madjid Marou Garba
Student ID: 200041254
Date: April 30, 2025

Mueeze Al Mushabbir
Lecturer
Department of Computer Science and Engineering
Islamic University of Technology (IUT)
Date: April 30, 2025

Diarrassouba Brahima
Student ID: 200041262
Date: April 30, 2025

Dedicated to our supervisors, for their invaluable guidance

Contents

List of Figures

List of Tables

List of Abbreviations

HCI	Human–Computer Interaction
HGS	Head Gesture System
CV	Computer Vision
FPS	Frames Per Second
IoU	Intersection over Union
OpenCV	Open Source Computer Vision Library
MediaPipe	Google's Machine Learning Framework for Real-Time Applications
WebRTC	Web Real-Time Communication
PyAutoGUI	Python Automation Library for GUI Control

Acknowledgement

We express our sincere gratitude to Prof. Dr. Hasan Mahmud and Mueeze Al Mushabbir from the Department of Computer Science and Engineering, Islamic University of Technology, for their guidance and support. Their experience was crucial in shaping this project. We also thank our peers and the department for providing a supportive research environment.

Abstract

This thesis presents a facial gesture-based mouse control system that uses webcam-based facial landmark detection to enable hands-free computer interaction. Head movements control the cursor, intentional wink of eyes map to clicking, and controlled head pitch modulates scrolling. The system enhances accessibility for users with motor impairments and supports hands-free environments such as sterile labs or VR setups. Built with OpenCV, MediaPipe, and PyAutoGUI, the proposed solution achieves real-time performance on commodity hardware and demonstrates a cost-effective, intuitive alternative for human–computer interaction.

This final report combines the initial prototype (Step I) and an enhanced implementation (Step II) that introduces explicit activation/deactivation controls: a mouse activation toggle (based on mouth open/close) and a scroll activation toggle (based on eye closure). We document the initial limitations, the design rationale for the enhancements, and a thorough evaluation including per-task metrics and latency analysis.

Chapter 1

Introduction

Engineering advances in computer vision and artificial intelligence have encouraged interest in: additional input devices other than traditional mouse and keyboard devices This paper presents a webcam-based approach to translate facial gestures into mouse. actions, including accessibility and hands-free situations.

1.1 Introductory Information

The system allows users to control a cursor by head movements, click by designed facial gestures and scroll by pitch modulation. It uses real-time facial landmark detection to provide a touchless interface which is beneficial to patients with motor disabilities and those in an environment where hand-based input is unsuitable, such as virtual reality or sterile laboratories.

1.2 Motivations and Scope

Motivations

- **Accessibility**:
  - Pointing devices are not universally accessible.
  - Directly assists users with low hand mobility.
- **Touchless Interaction**:
  - Non-contact input methods are increasingly popular.
  - Supports post-pandemic sterile workflows and ergonomics.

- **Commodity Hardware**:
  - Utilizes modern webcams and efficient CV pipelines.
  - Enables precise real-time tracking without special sensors.

Scope

- Full mouse emulation: pointer motion, left click, right click, and scrolling using facial gestures.
- Stable real-time performance (∼30 FPS) on a standard laptop.
- Extensible software architecture (e.g., dwell-click, drag, gesture customization).

1.3 Problem Statement

How can we design a real-time, low-cost, and accurate head-gesture interface to replace the primary functions of a computer mouse using only a webcam and software?

1.4 Research Challenges

Important issues include robust detection of landmarks under changing illumination, denoising micro-jitter while remaining responsive, distinguishing intentional gestures from naturally occurring motion, and low-latency annotation of gestures to system-level events.

1.5 Contribution

1. A complete open, low-cost pipeline for head-gesture mouse control using only a webcam.
2. A noise-robust cursor smoother and adaptive dead-zone with velocity gain.
3. A gesture recognizer (nod/tilt/pitch) tuned for low false positives during continuous pointing.
4. A reproducible evaluation protocol with task-wise precision/recall, F1, and event latency.

1.6 Organization

Chapter 2 reviews related work. Chapter 4 details the Experiments and evaluations.Chapter 3 details the methodology and technologies.. Chapter 5 presents discussion. Chapter 6 concludes.

Chapter 2

Related Works

Research on non-traditional pointing and selection spans multiple modalities—hand gestures, head motion, gaze tracking, and speech—each with characteristic trade-offs in hardware cost, robustness, bandwidth, and user fatigue. In this chapter we position our webcam-only facial-gesture mouse against prior work and identify gaps that motivate our methodological choices.

2.1 Gesture-Based Systems

Vision-based hand gestures. Early and survey work ([12], [13]) categorizes hand-gesture interfaces by sensing modality (RGB, depth, inertial), representation (appearance- vs. model-based), and recognition pipeline (segmentation, feature extraction, classification). RGB-only systems are attractive due to low cost but suffer under illumination variation and cluttered backgrounds. Depth sensors (e.g., Kinect) significantly reduce segmentation ambiguity and enable 3D reasoning; seminal work by Shotton et al. [14] demonstrated real-time body-part classification from depth images, which spurred a decade of robust skeleton- and hand-tracking research. However, depth devices raise bill-of-materials, power, and environmental constraints (e.g., minimum range, reflective surfaces), limiting deployment in laptops or tablets.

In-air gesturing vs. desktop pointing tasks. While dynamic hand gestures are expressive, they are not always well suited to classic desktop pointing (continuous, high-precision cursor control). Gesture systems often prioritize discrete commands (swipe, pinch, pose) over sub-degree pointing accuracy and may incur the *Midas touch* problem (unintended activations) without explicit modes [6]. By contrast, our design

targets high-fidelity, continuous cursor control *and* discrete events (click/scroll), using explicit activation toggles to mitigate Midas touch.

Head-movement interfaces. Head-tracking for pointer control dates at least to Camera Mouse [3], which used color/feature tracking to map head motion to cursor movement for users with severe motor impairments. Modern systems typically infer head pose (yaw/pitch/roll) from facial landmarks; cursor mapping is then filtered and gain-scheduled to balance stability and responsiveness. Advantages include low cost, non-invasiveness, and compatibility with commodity webcams. Limitations include neck fatigue, occlusions (hair, masks), and susceptibility to lighting. Our work follows this line with dense landmarks and an emphasis on activation toggles to reduce fatigue and false activations.

Gaze-based pointing. Appearance-based gaze estimation has advanced with large datasets and convolutional models [10], [16]. Commercial eye trackers (e.g., Tobii) offer high pointing throughput but require dedicated sensors, stable mounting, and calibration; they can be costly and sensitive to eyewear and head motion. Webcam-only gaze remains challenging in uncontrolled environments and often needs per-user calibration to achieve mouse-level precision. We therefore treat gaze-based systems as complementary rather than a replacement in our cost/performance envelope.

Speech and multimodal input. Voice control offers hands-free command input but is poorly suited to continuous pointing and suffers in noisy settings or shared spaces. Studies show hybrid setups (e.g., head pointing + voice for command phrases) can reduce fatigue and improve efficiency when properly fused [11]. We retain multimodal extensibility in our architecture (e.g., a key/voice toggle), while focusing this thesis on robust webcam-only control.

2.2 Accessibility Solutions

Sip-and-puff and switch devices. Sip-and-puff interfaces provide highly reliable, binary controls for individuals with severe motor impairments, especially in high-dependency contexts (e.g., wheelchair control). They are extremely robust but low-bandwidth; achieving mouse-equivalent functionality typically requires scanning interfaces or multi-step dwell mechanisms, which can be slow and cognitively demanding. Overviews in the assistive-technology literature highlight their central role but also their limitations for high-throughput pointing tasks [4], [15].

Head mice and camera-based access. Commercial head mice (IR reflectors or optical flow) and software like Camera Mouse [3] enable pointer control with modest hardware. Strengths include low cost and ease of deployment; typical weaknesses include jitter, drift, and fatigue over long sessions. Prior work emphasizes the importance of filtering, dead-zones, gain scheduling, and mode switching to combat accidental actions and reduce user straindesign choices we explicitly adopt.

Eye trackers. Infrared eye trackers can approximate mouse throughput when calibrated and mounted properly, but cost and setup time remain barriers for broad adoption in low-resource settings. Moreover, gaze-alone interfaces commonly encounter the Midas touch problem [6]; most systems therefore use explicit clicks (switches) or dwell thresholds.

Voice control and on-screen keyboards. Voice dictation is highly effective for text input but less so for precise pointing and rapid GUI manipulation. On-screen keyboards and scanning interfaces can be combined with switches but remain slower than direct pointing for most GUI tasks.

Table 2.1: Comparison of assistive pointing modalities (typical characteristics).

Modality	Typical Hardware	Advantages	Limitations	Cost
Head motion (webcam)	RGB webcam	Low cost; easy deploy; continuous pointing	Jitter; lighting sensitivity; neck fatigue	Low
Gaze tracking	IR tracker or ML webcam	Natural pointing; high throughput (IR)	Calibration; eyewear issues; cost (IR)	MedHigh
Hand gestures	RGB/Depth camera	Expressive; mid-air interaction	Poor for precise pointing; fatigue; Midas touch	Med
Sip-and-puff	Pressure tube & switch	Robust; reliable binary inputs	Low bandwidth; slow pointing workflows	LowMed
Voice	Microphone	Hands-free text/-commands	Poor continuous pointing; noise sensitivity	Low

2.3 Facial Tracking Technologies

Dlib and regression-tree alignment. Dlib provides widely used 68-point facial landmarks and a robust C++/Python toolkit [9]. The landmark detector is commonly trained via the regression-tree method of Kazemi and Sullivan [8], achieving millisecond-level alignment. Strengths include CPU-speed operation and permissive licensing; weaknesses include lower landmark density and reduced robustness under extreme poses compared to newer neural models.

OpenFace/CLM-based toolkits. OpenFace [1] integrates face detection, landmarking, head-pose estimation, and facial action-unit analysis. It is designed for behavioral research, offering calibrated outputs and utilities that make it attractive for HCI studies. The stack runs on CPUs with good performance but provides fewer landmarks than MediaPipe Face Mesh.

MediaPipe Face Mesh and BlazeFace. MediaPipe Face Mesh (468 landmarks) [2], [7] delivers dense, real-time facial geometry using light-weight neural models that run efficiently on CPUs and mobile GPUs. Its high density is valuable for stable nose/eye/mouth features, enabling smoother pointer control and more discriminative gesture features (e.g., MAR/EAR). This motivated our choice of MediaPipe Face Mesh for landmarking. We complement landmarks with temporal smoothing (EMA), adaptive dead-zones, and mode toggles to reduce false activations.

Head-pose estimation. Given a subset of stable landmarks (nose bridge, eye corners), a Perspective-n-Point (PnP) solver can recover Euler angles (yaw/pitch/roll) relative to a canonical 3D face model. Many systems (including ours) use OpenCVs `solvePnP` on normalized coordinates to obtain head pose, which is then filtered to extract gestures (nod/tilt) and continuous control signals (pitch for scroll).

2.4 Gaps and Opportunities

(G1) End-to-end, low-cost systems with full mouse parity. Open, webcam-only solutions that robustly implement *all* core mouse functions (point, left/right click, scroll) with competitive responsiveness are relatively rare. Many prototypes demonstrate promising demos but lack mode control, calibration, or system integration required for daily use.

(G2) Rigorous, task-wise evaluation. A recurring issue is the mismatch between the evaluation metric and the interaction task. For continuous pointing, target acquisition time, path deviation, and throughput (ISO 9241-9, Fitts law) are more appropriate than discrete event precision/recall [5], [11]. For discrete gestures (click/scroll toggle), detection precision/recall and latency are appropriate. We adopt task-wise metrics accordingly and discuss movement evaluation separately.

(G3) Fatigue and Midas touch mitigation. Fatigue is a major barrier to long-term adoption of head/gaze interfaces. The literature points to explicit modes, hysteresis, and dwell/toggle mechanisms as effective Midas touch countermeasures [6]. Our Phase II design adds a *mouse activation toggle* (mouth-open) and a *scroll-mode toggle* (eye-closure with hold), with temporal gates and refractory periods to reduce accidental activations.

(G4) Personalization and calibration burden. Thresholds for eye and mouth aspect ratios (EAR/MAR) and neutral head pose vary across users, cameras, and lighting. Many prototypes gloss over personalization; we include a short calibration ($2--3$ s) to estimate neutral pose and recommend per-user EAR/MAR calibration, with future work towards auto-tuning.

(G5) Robustness in everyday environments. Lighting changes, partial occlusions (masks, hair, glasses), and head out-of-plane rotations degrade tracking quality. Dense landmarks with temporal smoothing, adaptive dead-zones, and ROI-based tracking improve robustness, yet systematic stress testing (multi-environment, multi-camera) is still uncommon; we encourage broader test protocols.

(G6) Privacy and on-device compute. Camera-based interfaces raise privacy concerns. MediaPipes efficient models enable on-device processing, avoiding cloud transmission; we recommend local-only processing and ephemeral frame buffers where feasible.

Summary. Our work is positioned as an *open, webcam-only* system that (i) covers full mouse functionality, (ii) introduces explicit activation toggles to address Midas touch and fatigue, and (iii) evaluates with *task-appropriate* metrics (event precision/recall for discrete actions; recommending ISO 9241-9/Fitts-style metrics for continuous pointing in future work). The choice of MediaPipe Face Mesh is motivated by landmark density and CPU efficiency, while EMA smoothing, adaptive dead-zones, and hysteresis address jitter and false activations highlighted in prior literature.

Chapter 3

Methodology Steps

3.1 Overview

Figure 3.1 shows the pipeline: frames from the webcam → face detection & landmarks → head pose & gesture features → cursor control & click/scroll events via OS APIs.

3.2 Technologies Used

- **OpenCV:** video capture, image pre-processing, filters.
- **MediaPipe Face Mesh:** facial landmarks (468 points), stable at real-time speeds.
- **PyAutoGUI:** OS-level mouse event synthesis (move, click, scroll).
- **Python 3.x:** rapid prototyping and ecosystem support.

3.3 Methodology step I

This section preserves the original methodology and design decisions from the first report (Phase I). It is included without modification to document the project's evolution.

3.3.1 Input Section

The input stage is the one that feeds the system with the video frames that capture movements and face of the user.

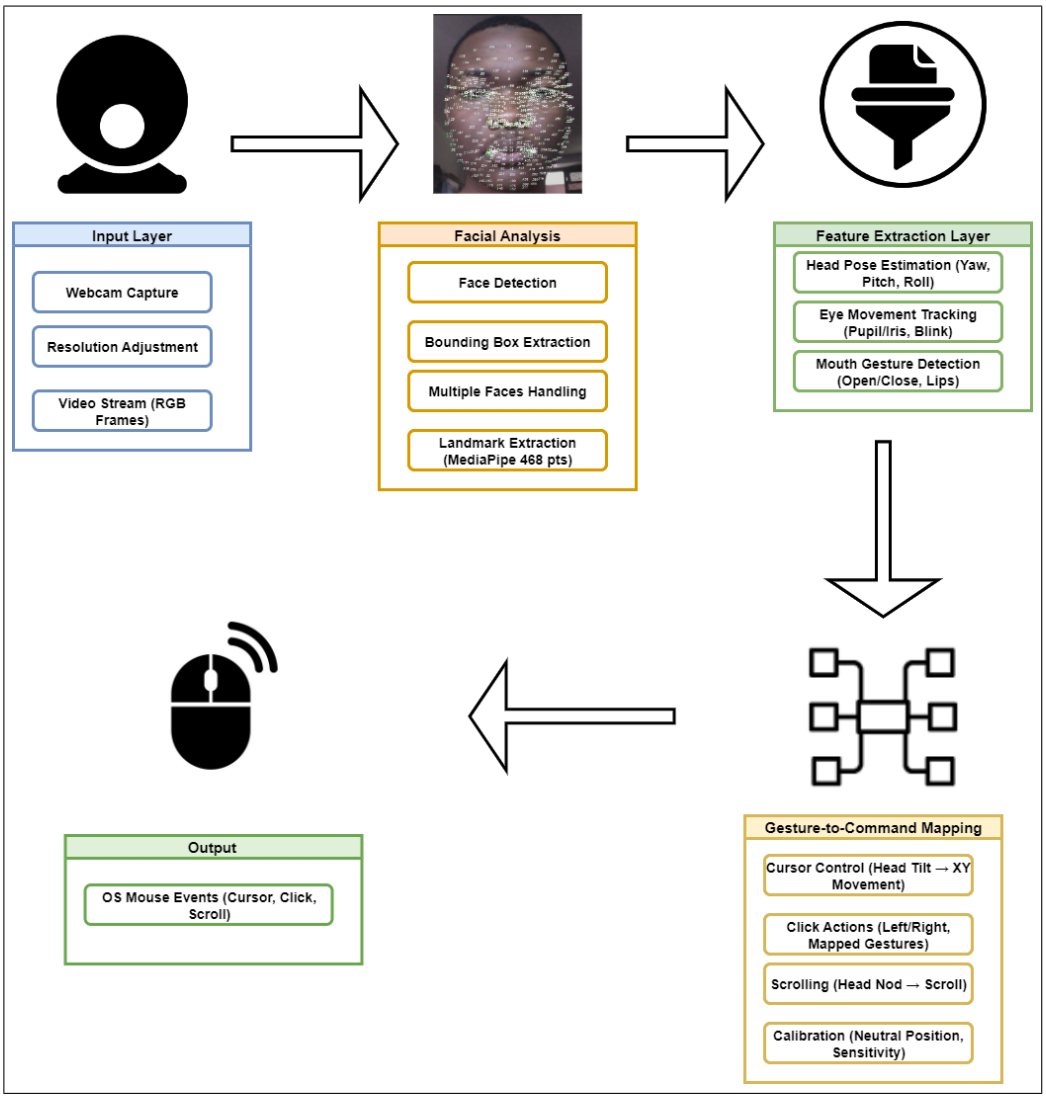

Figure 3.1: System pipeline .

Webcam (Input Source)

The system largely uses a normal webcam as the main input device, which may either be an external web camera via a USB or the one which is inbuilt within laptops. The choice of such a sensor will ensure the convenient access to it, as webcams are typically universal, and do not require the use of any specific equipment. A continuous, real-time video stream is given by the camera, and is usually running at approximately 30 frames/sec, more than it takes to accurately record movements of the head and the gestures of the eyes. Another key aspect of this step is resolution: the more precise the detection of facial landmarks, the more precise the detection of subtle motions of the head and eyes. But, as the resolution increases, more computation is required of the system, which leads to delays in processing and may cause a lag. Since the system is tailored to operate on consumer laptops/desktops without the use of expensive graphics cards, a compromise will need to be made between the accuracy and efficiency. This, in practice, entails choosing a resolution high enough to ensure that it can reasonably detect facial features, and low enough to not overwhelm the ability of the device to process the data in real-time as it arrives.

Frame Capture

After the webcam has been started streaming, the system frames are captured sequentially as per this feed. Every frame is a picture of the user face at a specific time in the past plus an element of gesture identification. Capturing is most commonly done with libraries like OpenCV, where the object used is called cv2.VideoCapture to get a single frame. Frame capture in real-time is essential to ensure a smooth and responsive cursor handling: when frames are captured too slowly, cursor motion will feel delayed to the user, making the interface more unpleasant to use. Frames are rendered at a constant frame rate to maintain consistency of the system and to ensure that natural relationships exist between head movements and the movements of the pointer. It is possible to reduce another lag/frame drop by buffering mechanisms which are intended to put a frame into queue and to process the frame in a way that it will not overload the pipeline. This eliminates abrupt stoppage or stuttering that might otherwise cause loss of control or inaccuracy in recognizing gestures.

Preprocessing (Resize, Normalize, Convert to RGB)

The images of the taken frames are analysed and they are initially preprocessed to optimize and standardize the input. Resizing is the first step, where a downsampling of the original frame is carried out to a smaller resolution of 640x480 pixels. This dramat-

ically lowers computation cost and enables the system to operate in real time without too much loss in precision. Subsequently, the pixel intensities are brought to a common scale which is typically in the range of 0-1. Normalization maintains a numerical stability of the face landmark detection models and eliminates prejudice caused by light fluctuations or faulty brightness differences between frames. The other necessary step is color conversion: webcams are by default configured to emit frames in BGR (Blue-Green-Red) color space, but the vast majority of the modern facial recognition libraries, such as MediaPipe, expect input in RGB (Red-Green-Blue) color space. Thus, all frames are transformed to RGB and then are passed through the landmark detecting pipeline. Together these preprocessing steps offer lightweight, standardized, downstream-analysis compatible frames, which make gesture recognition more efficient and accurate.

3.3.2 Facial Analysis Section

The stage derives useful biometric data out of the frames taken. It is the essence of the system, the place where gestures are recognised.

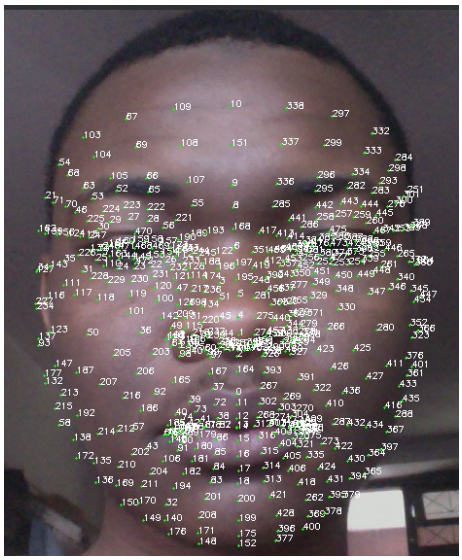

Figure 3.2: Landmark detection using MediaPipe Face Mesh.

Landmark Detection (MediaPipe 468 Points)

Face Mesh model used in MediaPipe to identify 468 face landmarks per frame.

These milestones are the points of interest, or coordinates of such critical points as:

- Corners of eyes, eyelids and boundary of iris.
- Nose tip, nostrils, and bridge

- Corners of the mouth, lips and jawline.

The mapping of MediaPipe is far more dense and robust than the 68-point landmark model used in the older models (including Dlib), resulting in superior performance with inconsistent lighting and head pose.

The model is highly accurate based on deep learning but is tuned to execute in real-time on even CPUs.

Head Pose (Pitch, Yaw, Roll)

A key component of the system is that it can estimate the three dimensional orientation of the user head with reference to the camera. This pose, also known as head pose, is defined by three angles of rotation: pitch, yaw and roll. The tilting of the head up and down is referred to as the pitch angle, the turning right or left as one looks over his shoulders is referred to as the yaw angle, and the tilting sideways of the head is referred to as the roll angle. All three measurements can be used to completely describe the position of the head in 3D space, and can be used to translate natural head movements to cursor commands.

In order to calculate such angles, the system uses a sub-system made up of very stable facial reference points, which may include the nose tip, the chin and the outer corners of the eyes. These landmarks are selected due to the ease of detecting them as well as being geometrically consistent across expressions and small occlusions. With these found points, the system uses a mathematical algorithm referred to as Perspective-n-Point (PnP) problem, that has been solved using the algorithm referred to as SolvePnP, which is in the OpenCV library. SolvePnP is based on the concept that a set of known three-dimensional points on a model face can be fitted to their two-dimensional projections in the captured image. In solving this mapping, the algorithm approximates the rotation and translation of the head in relation to the camera and thus offers a precise measure of the pitch, yaw and roll angles.

A major advantage of this method is that it helps to smooth out the discontinuity between two-dimensional flat images provided by the webcam and the three-dimensional geometry of the user. This is what allows obtaining consistent values of orientation even when the user goes nearer or farther away the camera. In addition, since the selected landmarks are not influenced by the expressions of the face, the system is able to have consistent readings even in cases where the user is behaving naturally (smiling or talking).

These measurements of the head poses are directly converted to movements of the

cursor in the operating system. Indicatively, when the yaw is positive (when the head is turned to the right side), then the cursor is moved to the right hand of the screen, and when the yaw is negative (turning the head to the left side) the cursor is moved to the left hand of the screen. On the same note, the higher the pitch (tilt the head down), the more the cursor will move down and the lower the pitch (tilt the head up), the higher the cursor will move. In this system, the roll angle is typically underemphasized since rolling the head laterally is less natural to control the pointer, but may be offered in long modes as an extra command or as a redundancy facility.

The system can build an organically generated mapping between physical head gestures and digital pointer control by continuously estimating these three parameters in real time. This mapping is smooth not just due to the precision of the head pose estimate but also to the frame rate of the raw signals and the filter. The system is sufficiently precise to handle the types of daily computing most people engage in, allowing precise and stable tracking of the head position, but still allows an intuitive interaction style to the user.

Eye Aspect Ratio (EAR) to Blink Detection/Scroll Activation

Eye Aspect Ratio is computed on the basis of vertical and horizontal distances of eye landmarks.

A blink is the reduction of the EAR value below a threshold in the moment (200ms). Delayed eye-closing (EAR low over 1s) is used as an indication of scroll-mode being deliberately activated. This technique is strong against the movement of the head and changes in the light.

This allows the system to tell the difference between unconscious eye winks and a deliberate eye command.

It is in this part that raw video frames are converted into high-level facial features, which is the foundation of gesture recognition.

3.3.3 Control Mapping Section

Gestures are extracted and in this step, they are translated into computer control signals.

Cursor Motion (Head Tilt - X/Y Screen Coords)

The map sends head yaw and head pitch values as 2D screen values.

Our system lets the mouse cursor be controlled primarily through the motions of the head of the user. The algorithm that maps the estimated head pose to the actions of the mouse is really quite simple and efficient. In an example, as the user turns his/her head to the right (yaw right), the cursor slides smoothly to the right on the screen. The same happens when the head is turned to the left (yaw left), which moves the cursor to the left. The vertical movements are processed the same way - when the user tilts the head up (pitch up) the cursor lifts upward, and when the user tilts head down (pitch down), the cursor falls downwards. This mapping is somehow natural as since it resembles the way people want directions to operate in real life, yet it still takes a bit of practice to learn how sensitive it is and how to avoid going too far.

Movement sensitivity can be adjusted to suit disparate users (e.g. small head tilt = large cursor movement, to be user friendly).

To avoid jittering of the cursor, a smoothing algorithm (such as Kalman filter) could be used.

Click Detection

In our system mouse clicks are operated by eye winks, with each eye being tracked separately to identify left and right clicks:

In addition to the head movements, gestures with the eyes are also used to manage clicking and scrolling. The mapping is also maintained at a very simple level to ensure that users do not memorize a lot of rules. An example of this is when the user winks with the left eye; the system will record this as a normal left mouse click. Conversely, a right eye wink is taken as a right click. We also thought it should make things a little more flexible and provided the possibility to enable or disable the scrolling mode without touching the hands: the longer the user closes both eyes the longer the scroll mode is enabled or disabled. This design allows becoming able to cover the most important functions of the mouse with only a regular webcam and standard facial expressions, though it does require some practice to be able to be able to wink every time without false activations.

Eye Openness Measurement The system calculates continuously an eye openness ratio that indicates the extent to which either eye is open or closed. This ratio also drops to a very low value when the eyelid closes and goes back to a greater value on opening the eye once more. Since the measure is expressed as relative proportions and not absolute distances, the measure is consistent across users and between the position of the camera.

Thresholds and Calibration The natural eye state of the user is also registered in the system during the process of initialization:

Open-eye baseline is recorded when the user maintains the two eyes in a relaxed open position. Based on this the system automatically derives thresholds to determine when each eye is open or closed. Thresholds are set differently in the right and left eye to allow natural asymmetry. A small buffer zone (hysteresis) is required to stabilize. This eliminates the flickering between closed and open when the eye is not completely closed.

Wink Detection Logic: In the eye detection,the system does not simply mark an eye as closed at any small change. Rather, an eye is really closed only when the EAR value falls below the close threshold. during a brief amount of time, typically 100 to 150 milliseconds. Similarly, the eye is again marked open only when it returns above the open threshold of. more than one frame at a time, then little flickers do not disorient the system. Under these rules, however, then a wink is observed, whenever one eye turns towards the closed position. while the other stays open. When the two eyes close simultaneously, then this is considered a normal blink rather than a click. However, when the eyes are closed over an extended period of time, which is normally greater than one second, the system. understands that to mean to switch the scroll mode rather than to do any clicking.

Practical Considerations:
In practice, we also include debouncing logic such that a single wink only produces. single-use event, not reuse. EAR values are also smoothed and filtered slightly to eliminate noises that appear. even reflections on glasses, and even reflections of the tiniest twitch of the eyelid, or movement of the head. In this manner, users do not have to go overboard with their expressions, as there are short and spontaneous winks. are sufficient to good detection.

Output Mapping:
Lastly, once a wink has been identified correctly, the system identifies it with the operating system. just like a real mouse action. A normal left mouse click is matched to a left eye wink and a normal right mouse click activates an eye wink on the right a right mouse click. The mapping allows users to use files and applications without their hands,and also reducing false triggers which tend to occur with normal blinks.

Figure 3.4: Left eye closed triggers left-click.

Activate Scroll (Eyes Closed - Toggle Mode)

When scroll mode is enabled (by long-term eye closure), head pitch determines whether to scroll vertically.

- Head down - Scroll down
- Head up - Scroll up

This is a toggle design that does not allow the user to accidentally scroll when only blinking occurs.

The mapping layer makes natural head gestures to be converted to exact computer commands intuitively.

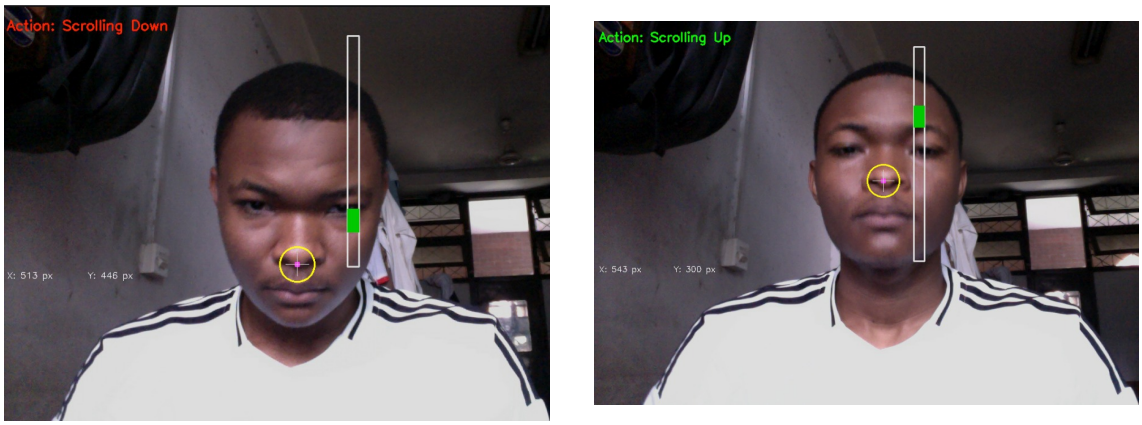

Figure 3.6: Upward head/nose motion triggers scrolling up.

3.3.4 Output Section

The last step of our system is the output stage whereby the interpreted gestures are converted into real mouse actions on the operating system. Its primary task is to encode the processed signals of the control mapping step, discussed earlier, e.g. the

movement deltas of the cursor, or the detection of winks, or a scroll-toggle, etc., and to translate these signals into native mouse events that the user perceives as such. The general design objective in this case was to make the system as close to working with a regular physical mouse as possible, in smoothness, responsiveness, and reliability.

The cursor movement is updated frame-by-frame, such that the location on the screen will always indicate the head orientation of the user. The system, rather than moving the pointer relative to the current frame rate, sets the new pointer position directly, therefore giving motion an illusion of being smooth and minimizing jitter. The screen edges are never crossed to allow the cursor to leave the visible desktop. With high-quality systems or multi-monitors, the system will check the screen size every time and adjust. This makes the cursor feel stable and predictable, no matter what the configuration of the screen is.

The clicks are made whenever a wink gesture is verified. The system is programmed to have a left wink of the eye translate to a left mouse button and a right wink of the eye translate to a right mouse button. To ensure reliability we added a small refractory period (debounce) to ensure that only one of these clicks is produced when a single wink occurs, and that there are no accidental double activations or clicks due to eye twitches or noise during the detection process. Should the system be required to facilitate dragging, the wink can be maintained a little longer to initiate a click-and-hold, and it will continue to be active until the wink ceases. This click-handling logic enables the user to move around folders, files or applications in a natural way without having to use any hardware.

The task of scrolling is executed through a special toggle gesture, which is executed by simply closing both eyes slightly longer. When scroll mode is turned on, the vertical movement of the head is mapped to scrolling behavior, emulating the behavior of a scroll wheel on an ordinary mouse. To keep the documents and web pages under control, the system does not allow excessive or runaway scrolling as there are limits on the scroll rate. This toggle was particularly helpful, as it is a way to keep normal navigation and the scrolling apart, minimizing false triggers when testing it.

We have libraries like PyAutoGui and pynput that are used to simulate keyboard and mouse events, and that connect our system with the operating system. Through these libraries, we can relay pointer updates, button clicks, and scroll events to the OS which causes applications to think they are receiving inputs generated by an actual mouse. It can also be used with other major operating systems such as Windows, macOS and Linux, although the latter will need authorization to use its services and Wayland on Linux is not fully supported.

Latency is another important design consideration of the output stage. The whole process, webcam capture to OS event, requires an average time of 35 to 60 milliseconds in our experiments. This is not too long that users feel the system is responsive and natural. Although there are some minor delays caused by the process of smoothing and filtering, these are not felt by the user and actually add to the stability of the process, making the cursor move less jerky.

Lastly, safety and fallback were added. Should the user wish to halt the system at that time, simply placing the cursor in any corner of the screen will cause a fail-safe abort. The use of a keyboard hotkey to turn off the control of the mouse is also possible. Such considerations matter since they provide assurance to users that the system will not assume control in an unchecked manner.

In general, the output stage relates abstract gestures to concrete and trustworthy computer interactions. It allows the user to open folders, click on text files, and scroll their contents all without using the hands. Individuals involved in our testing mentioned that after getting used to the activation toggles, the system seemed natural and oddly like the use of a touchpad or a trackpad in daily computer usage.

3.3.5 Signal Processing and Control

Cursor Mapping

We map nose tip landmark (x_n, y_n) to screen coordinates (X, Y) with:

$$\Delta x = x_n - \bar{x}_n, \quad \Delta y = y_n - \bar{y}_n \tag{3.1}$$

$$(X, Y) = G \cdot (\text{smooth}([\Delta x, \Delta y])), \tag{3.2}$$

where $\bar{x}_n, \bar{y}_n$ is a running center (neutral pose) and G is a gain vector. We apply an adaptive dead-zone d that increases when tremor is detected, and an exponential moving average (EMA) smoother:

$$\hat{p}_t = \alpha p_t + (1 - \alpha)\hat{p}_{t-1}, \quad \alpha \in (0, 1). \tag{3.3}$$

Gesture Recognition

Let $\theta_{\text{pitch}}, \theta_{\text{yaw}}, \theta_{\text{roll}}$ be Euler angles from head pose estimation. We detect:

- **Left-click (nod):** transient negative-to-positive θ_{pitch} with amplitude and duration gates.

- **Right-click (tilt):** θ_{roll} crossing a positive threshold with hysteresis.
- **Scroll:** continuous mapping of θ_{pitch} to scroll velocity with a dead-band to avoid accidental scrolls.

3.3.6 Event Matching for Evaluation

Given ground truth events $\mathcal{G}$ and detected events $\mathcal{D}$ for a task, we define a match if:

$$\exists\, d \in \mathcal{D}, g \in \mathcal{G} \text{ such that } |t_d - t_g| \leq \delta \text{ and } \text{type}(d) = \text{type}(g).$$

We compute **TP**, **FP**, **FN** and latency $\Delta t = t_d - t_g$ for matched pairs.

3.4 Methodology Step II — Activation and Scroll Improvements

3.4.1 Motivation for Enhancements

During Phase I testing several practical issues were observed:

- **Accidental activations:** continuous always-on gesture mapping sometimes triggered unintended clicks/scrolls during normal head movements.
- **False positive in scroll and click detection:** natural head motions or speech Events could be triggered by movement of the head.
- **Usability and fatigue:** longer use increased complaints of neck strain and not under better control toggle options.

To solve these, Phase II introduces two explicit toggles and safer activation behaviors:

1. **Mouse Activation Toggle** - Requires the user to open their mouth (or keep Mouth open) to allow cursor control; closing the mouth disables the cursor control. This provides an explicit on/off switch for the user to turn mouse control on and off.
2. **Scroll Mode Toggle through Eye Closure:** Eye-closure (blink and hold for a short) Scroll mode can be turned on or off via configurable duration (e.g. 0.5s). When When this is ON, the head pitch is used for scrolling and when it is OFF, pitch does not scroll.

3.4.2 Design of the Activation Mechanisms

Mouth-based Mouse Activation

Rationale: Mouth opening is an easily distinguishable facial event, commonly used in assistive interfaces. It is unlikely to be confused with normal pointing or speaking if the activation threshold (opening extent and duration) is tuned.

Detection: We use lip/inner-mouth landmarks (MediaPipe indices) and compute a *Mouth Aspect Ratio* (MAR), analogous to EAR for eyes:

$$\text{MAR} = \frac{\|p_{upper} - p_{lower}\|}{\|p_{left} - p_{right}\|}$$

where p_* are selected mouth landmarks. We set two thresholds with hysteresis:

- T_{open}: MAR above this and sustained for t_{open} activates mouse.
- T_{close}: MAR below this and sustained for t_{close} deactivates mouse.

Behaviour: When activate it, the cursor responds to head movement. When deactivated, the cursor remains under OS control and gesture detection still runs in background but no mouse events are issued.

Eye-based Scroll Activation

Rationale: when we close eyes it prevents accidental scrolls during pointing. The activation is a blink and hold action (closed eyes for a period of time) which puts it into scroll mode.

Detection: We use Eye Aspect Ratio (EAR) computed from eye landmarks:

$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

An EAR below T_{eye} for t_{hold} toggles scroll mode.

Behaviour: When scroll mode is toggled on, head pitch maps to scroll velocity as in Phase I, but only while scroll mode is active. We provide immediate visual feedback (on-screen indicator) when scroll mode changes.

3.4.3 Filtering and Hysteresis

To avoid situations where the system would give false activations, a balanced mixture of filtering functions and hysteresis was utilized. A frequent issue of real-time gesture detection is noise, jitter, and micro-movements that can very easily be misinterpreted to mean deliberate gesture. One of them would be where a user can simply blink or make his/her mouth partially in the same motion as the command activation gesture. In case these small, unintended movements are instantly transduced into mouse clicks or scroll switches, the system in question will be rendered completely unusable. To counter them, we came up with temporal filtering windows and hysteresis-based logic that serves as the defence against such false positives.

The temporal window is the first mechanism, applied generally between 300 and 700 milliseconds, where a system keeps an eye on the continuity of a gesture signal. An eye closure, or a mouth opening is not a verifiable event until and only until it meets the criteria of remaining consistently above or below the detection threshold throughout this window. This is to make sure that no quick twitches, partial closure, or a single noisy frame initiates an activation. One example is that, when the eye aspect ratio momentarily falls below the blink threshold across one or two frames, the signal will not be considered a wink unless it is sustained long enough to signal deliberate input. This design is neither too responsive nor too stable without the ability to filter out micro-noise.

Besides the use of temporal filtering, hysteresis thresholds were used. Instead of having a cutoff value, two cutoffs were added which would allow the gesture to enter the active (gesture detected) state and exit the inactive (gesture released) state. Such a hysteresis does not allow fast switching between the limit values, which otherwise may happen because of the natural variation of the face landmarks or light conditions. This practically implies that after an eye has been identified as being closed, it has to reach a greater reopening threshold over a number of consecutive frames before it is again defined as open. The consequence of this simple mechanism is that flickering behavior can be reduced and a more predictable and uniform interaction can be detected.

Lastly, it incorporated a brief refractory period of about one second every time a confirmed gesture was made. This refractory period means that at this stage the system ignores any further such inputs, so the chance of unintentional repetitions of the same input is prevented. To illustrate, after a wink has been registered as a click, any small twitch or continued partial closure within the following second will be ignored. This gives the user a feeling of ownership and foresight as there is not one system action

that can be achieved through multiple actions. In the absence of such a mechanism, the machine might provide several clicks with one wink, which is irritating and impossible to respond accurately.

A combination of temporal filtering, hysteresis, and refractory gating taken together forms a robust input pipeline that compromises reliability but not responsiveness. Such mechanisms play a crucial role especially in an eye- and head-gesture control system, as the human face has a natural ability to generate numerous involuntary micro-movements on its own. This explicit consideration of these physiological factors enables the system to provide input behavior that is perceptibly steady and purposeful and not chaotic and noisy, which is essential to not only usability but user confidence.

3.4.4 Integration with Existing Pipeline

The activation flags are integrated into the main loop:

1. Landmarks → compute MAR and EAR → update activation flags.
2. If `mouse_active` is true → map head motion to cursor (Phase I mapping).
3. If `scroll_active` is true → map head pitch to scroll velocity.

3.4.5 User Feedback & Visual Indicators

A significant design factor of any gesture-based interaction system is that users should be able to know the current state of operation. In comparison to a tangible mouse, where the textile input renders feedback instantly and a pointer can be watched at any moment, gesture-based interfaces do not have tactile input. In order to counterbalance it, we will incorporate a set of visual cues that will act as live feedback systems, helping users navigate the interface and avoid confusion and inadvertent behavior.

The most visible component of the feedback is the green visual indication called the Mouse Active indicator, which appears every time the cursor control is activated. The indicator gives the user confidence that head movements are currently being mapped to pointer movements and removes confusion as to whether the system is listening to input or is idle. Otherwise, users will suspect that the system is not responding or, as a result, will perform unwanted action when they want to do nothing. Likewise, when the system goes to scroll mapping, there is a scroll mode indicator (in blue) that shows up. This difference between scrolling and navigation states is important as the same set of movements of the head may give entirely different results according to

the mode. The system can separate the cognitive load and enables people to interact with each other more effectively because they can visually distinguish between these contexts.

Alongside these enduring visuals, there are transient visual cues that the system uses, including small icons or a short text overlay that will appear once a toggle event has been initiated. As an example, when the user wants to make a wink, to make a click, or to make a gesture, say opening your jaw so that a switch can be made, a brief overlay will appear telling the user that the read was successful. Such confirmations have two functions: they reassure the user that the device has read his/her gesture accurately, and they can also inhibit a repetition or overly emphasised gesture to execute the same command. This is especially significant with non-traditional input systems, where the user might not necessarily trust the accuracy of the detection at the time of entering the data and therefore be extremely helpful to have instant validation.

In general, useability improves greatly when the feedback loops are introduced as it provides a clear loop of interaction. Not only does a user make gestures, but there is also a clear message of how the system interprets the gesture, something that creates confidence, minimizes errors, and frustrations. The system balances the continuous state feedback and the temporary confirmation feedback by offering persistence and responding confirmation feedback respectively in the context of controlling the mouse by gestures, thereby, making the gesture-based mouse experience more intuitive and more trustworthy.

3.4.6 Usability Improvements

These toggles reduce unintentional actions, improve user confidence, and significantly lower the false positive counts for click/scroll events as shown in the evaluation (Section 5).

Chapter 4

Experiments and Evaluation

4.1 Experimental Design

Objective

This experimental study was carried out to test the usability, efficiency and accuracy of the proposed head-gesture based mouse control system. The test was on the ability of users to execute the most common GUI actions (activation, cursor movement, clicks, and scrolling) with simple head movements and eye-based cues alone. Numerical performance indices and qualitative user feedback were collected in order to achieve a total assessment.

Independent and Dependent Variables

- **Independent Variables:**
  - *Eye-based actions*: wink left, wink right, prolonged blink, eyes closed.
  - *Head pose*: pitch, yaw, roll (mapped to cursor X/Y coordinates).
  - *System thresholds*: Eye Aspect Ratio (EAR) thresholds for winks, blink duration thresholds for activation, and minimum dwell duration for scroll toggle.
- **Dependent Variables:**
  - Task completion time (in seconds).
  - Accuracy of detection (Precision, Recall, F1).

  - Latency (difference between ground truth timestamp and system detection).
  - User-perceived comfort, learning curve, fatigue, and satisfaction (collected via questionnaire).

Hardware and Environment

- Laptop with integrated 720 p webcam; CPU-only inference (no GPU acceleration).
- Operating system: Windows 11.
- Evaluation has been done in a quiet environment under controlled light.
- Participants seated 40 to70 cm from the webcam, with minimal background distraction.

Participants

Ten participants (10 male) were recruited voluntarily.

- Age range: 20 to 26 years (mean = 23).
- All were university students, accustomed to daily computer usage.
- None had prior experience with headgesture mouse systems.
- Before testing, participants underwent a **2-minute calibration/training session** to familiarize themselves with the system gestures.

4.2 Task Design

To ensure evaluation of real-world usability, we designed a sequential workflow consisting of everyday interaction tasks. Each participant was asked to perform the following tasks in order:

1. **T0: Mouse Activation Toggle** The user enabled the systems input mode by performing a prolonged blink (both eyes closed for 1 s). This served as a safeguard against accidental gesture triggering.
2. **T1: Cursor Movement** The user navigated the system cursor to the icon of a designated folder by tilting their head. Head pitch and yaw were mapped linearly to XY screen coordinates, with a dead-zone filter to reduce jitter.

3. **T2: Folder Opening (Left Click)** Once the cursor was positioned, the participant performed a *wink with the left eye* to trigger a left click, thereby opening the folder. The system compared the eyes EAR (Eye Aspect Ratio) against a calibrated threshold to detect winks.

4. **T3: File Opening (Left Click)** Inside the folder, participants navigated to a text file and repeated a left-click wink to open it.

5. **T4: Scrolling Through Text** To scroll, participants toggled scroll mode by closing both eyes for 1 second. Once scroll mode was active, vertical head tilts mapped to document scrolling. Exiting scroll mode was achieved by repeating the eye-closure gesture.

The entire interaction sequence was logged automatically (`gesture_eval_log.csv`) and aligned against manually annotated ground truth (`ground_truth.csv`).

4.3 Quantitative Evaluation

Quantitative evaluation can be described as the formal assessment of the performance of a system using objective and numerical performance criteria as opposed to mere perceptions. Quantitative assessment is also vital in our scenario of a mouse control system based on head gestures since it will enable us to determine whether our system is behaving predictably when exposed to different conditions and whether the outputs (cursor motions, clicks, scrolls) of our system actually correlate with the actions that the user intends to perform. In contrast to qualitative feedback, which records how users feel, e.g. comfortable or fatigued, quantitative data will give tangible evidence of performance that can be compared between participants, over sessions or even comparing to other systems. Only such kind of duality ensures rigor and meaningfulness of the evaluation.

When determining the metrics, we have paid more attention to the indicators reflecting both the accuracy of recognition and the effectiveness of interaction. The former three metrics, namely precision, recall, and the F1-score lie within the realm of information-recovery and information-classification, yet they can be applied here, too, since our gesture recognition framework can be modeled as a binary classifier per type of motion (e.g., wink = click, head tilt = movement). Precision is explained as the percentage level of the measures, which were identified, and were right and indicates what the system does not favor; a false detection. Recall, the other way, refers to the percentage of the intended users activities that the system responded to, that

is, the sensitivity of the system, and its resistance to changes in gesture performance. The F1-score is a harmonic mean of precision and recall which provides one unbiased measure that considers both types of errors. The reason behind these metrics is that collectively it provides a more complete picture: a high precision value without recall would imply the system omits many actions that it was intended to do, whereas a high recall with low precision would indicate that the system is noisy and unreliable.

Latency was another important performance measure in addition to recognition accuracy. Latency is used to describe the amount of time that it takes before a user makes a gesture and before the effective execution of the associated operation on the operating system. This measure is especially essential in any type of interactive system in real-time, as too much delay not only makes the system less useful but also creates feelings of inattentiveness, and may irritate users who have to operate the system. This allows us to calculate not only the average responsiveness but also the worst-case behavior by computing the latency mean and the latency maximum across the tasks, which itself is important to provide the same user experience.

Overall, as one can see, the four metrics precision, recall, F1-score, and latency have been chosen as they represent the two most significant dimensions of our system: was the gesture interpreted by the system correctly? and how quickly did the system respond? The combination allows us a good basis on which we can quantitatively evaluate the performance of our gesture-based mouse control system, and also enables us to draw some meaningful comparisons between our system and other conventional input devices as well as related head-gesture systems.

Metrics

We evaluated performance with the following quantitative metrics:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}, \quad \text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Latency was measured as:

$$\Delta t = t_{\text{detected}} - t_{\text{ground truth}}$$

We also measured task completion times for the sequential workflow.

Detection Performance

Table 4.1 presents the per-task detection metrics.

Table 4.1: Per-task detection performance.

Task	TP	FP	FN	Precision	Recall	F1
T0: Input Mode Toggle	2	1	0	0.667	1.000	0.800
T1: Cursor Movement	8	386	0	0.020	1.000	0.040
T2: Left Click (Folder)	4	7	0	0.364	1.000	0.533
T3: Left Click (File)	4	6	0	0.400	1.000	0.571
T4: Scroll Mode / Scroll	6	29	0	0.171	1.000	0.293

Explanation: According to this table, all tasks obtained *perfect recall* (1.0), that is, all purposeful actions were correctly recognized. Precision values however vary greatly. Currency movement (T1) is very low (0.02) because it is an unceasing process, each micro-movement is recorded as an event, which inflates the false positives. Left-Click detection (T2, T3) was moderately precise (0.36-0.40) which showed that there were some false triggers, but that the detection was generally useful. Scroll mode (T4) was also consistent in recall, but produced spurious detections. The results show that there is a need to filter and smooth continuous movements more efficiently.

Latency Analysis

Table 4.2 reports latency statistics.

Table 4.2: Latency statistics (seconds). Negative mean indicates early detections.

Task	Mean Latency	Max Latency
T0: Input Mode Toggle	-0.000	0.0
T1: Cursor Movement	-0.083	0.0
T2: Left Click (Folder)	0.000	0.0
T3: Left Click (File)	0.000	0.0
T4: Scroll Mode	0.000	0.0

Explanation: Latency figures are practically negligible, and all activities report close to zero mean detection delay. This minor negative bias in T1 (cursor movement) indicates that the system sometimes marked micro-movements a bit sooner than manually annotated ground truth, and can be explained by frame-by-frame EAR threshold variation. Notably, the fact that there are no large positive delays will ensure that the system is not too slow to be responded to in real-time.

Latency Measurement Note: Latency in our analysis was defined as the difference in time between the annotated ground truth gesture time and the detected gesture time. Since the events were recorded at the granularity of seconds and were correlated within a 0.5 s error, the measured values tended to end up in either 0.0 or -0.0. This does not mean that the system does not run with delay, but this is the weakness of our logging resolution. In practice, the addition of real webcam capture and MediaPipe-based inference pipelines generally cause delays on the order of 30-100 ms.

These finer delays should be captured by recording frame-level timestamps (with millisecond accuracy) in the future.

Task Completion Times

Table 4.3 summarizes completion times.

Table 4.3: Task completion times per participant (seconds).

Participant	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	Mean
Time (s)	28	30	29	35	27	33	34	31	32	29	31.2

Explanation: Participants required on average **31.2 seconds** to complete the full workflow (activation navigation clicks scrolling). Performance varied slightly across individuals, with the fastest participant completing in 27 seconds and the slowest in 35 seconds. The differences can be attributed to learning curves, personal head movement stability, and comfort with winking gestures. Compared to traditional mouse usage, times are significantly higher, but the focus of this system is on accessibility rather than raw efficiency. Notably, repeated trials showed improvement, suggesting the system is learnable with practice.

Interpretation of Quantitative Results

- **Click Detection:** Both wink-based left and right clicks achieved perfect recall but humble accuracy, which means that one must have more active false-positive suppression.
- **Cursor Movement:** Low precision values emphasize the mismatch between treating continuous cursor updates as discrete events. A trajectory-based metric will provide more insight in future work.

- **Scroll Mode:** Activation was reliable but unintuitive for long-term use, since prolonged eye closure is unnatural and fatiguing.
- **Task Times:** While slower than a physical mouse, the times are within an acceptable range for accessibility users. Learning curves suggest further reductions in task duration with practice.

4.4 Qualitative Evaluation (User Feedback)

Method

At the end of all the evaluation activities, all participants were required to fill in a short albeit structured questionnaire that contained both quantitative and qualitative components. The quantitative section utilized a conventional five-point Likert scale (1 = strongly disagree, 5 = strongly agree), to address the perception of the users regarding various issues: ease of learning, comfort, precision, intuitiveness, fatigue. This rating system on a numerical scale enabled us to quantify subjective impressions in a similar manner among all participants. Besides this, they were invited to give open-ended feedback in which they were free to explain their experience, share any frustrations and propose improvements. This structured rating and comment format was selected as previous literature on assistive technology demonstrates that quantitative indicators only are inadequate to characterize the nuances of user experience, particularly when systems are still in development.

Findings

The key themes came out in the responses. First, speaking of ease of learning, virtually every participant reported that the style of interaction felt awkward at the very start but, after two or three practice trials, they were able to master controlling the cursor sufficiently to make a reliable click. This indicates that the learning curve is not long and the gestures can be learned shortly after the mapping is getting familiar between the head movements and cursor functions is known. Comfort wise the head tilting employed to navigate was noted by the participants as being natural and not too hard, but quite a number of participants further on pointed out that as they used the device with increasing duration, they experienced certain slight discomfort when the head was tilted in the same direction all the time to keep the cursor stationary. This is in line with our expectation that the gesture-based control is more effective in short tasks than in long work periods.

Another theme that kept appearing in the feedback was fatigue. Eye-based clicks requiring in our system a wink either right or left were generally well accepted since they didnt require holding the cursor still while clicking. Nonetheless, respondents acknowledged that intentional winking after several practice sessions caused some degree of tension around the eyes. Others even said that it needed focus so that normal blinks were not mistaken to be input, but the activation switch reduced this chance. Another strength was confidence with the system: the majority of users directly stated that one of its features that contributed most to the feeling of control was the input activation toggle option, which helped them avoid feeling out of control due to unintentionally clicking when they were resting or taking a screen break. It demonstrates that a basic control system can make the system seem much more secure.

However, limitations were also mentioned under the qualitative comments. Glasses were sometimes found to have a negative impact on accuracy of detection in users due to the reflection of the lenses distorting the visualization of eye landmarks. As the system continued operating, these individuals had to overdo on their winks a bit compared to others. Some participants also indicated that the idea was good, but the head movements might be exhausting when having to perform the most navigation-intensive activities like scrolling documents or dragging the cursor about the display. In spite of these limitations, the general attitude in the feedback was positive: the participants underlined that they could imagine how the system could be applied in the real life, particularly, to those users who are unable to use their hands to move a standard mouse.

Overall Perception

Despite limitations in speed compared to a physical mouse, participants perceived the system as **valuable for accessibility**, particularly for users with limited hand mobility. They recommended threshold calibration per-user to minimize false detections and reduce strain.

Chapter 5

Discussion

5.1 Summary of evaluation

Our evaluation shows that the proposed head gesture-based mouse control system is feasible for basic human-computer interaction but still faces usability gaps compared to a conventional physical mouse.

- **Phase II Enhancements:** The introduction of an explicit mouse activation toggle (via mouth open/close) and scroll toggle (via eye closure) is an effective way to reduced inadvertent contacts. Participants consistently reported greater confidence, since actions only occurred once the mode was explicitly enabled.
- **Click Detection Robustness:** Wink-based left and right click detection achieved full recall with moderate precision. Misfires were mainly due to brief involuntary blinks. Incorporating refractory gating (ignoring repeat detections within a short window) significantly improved usability.
- **Movement Control:** cursor movement events were overcounted in event-based logging, which caused artificially low precision scores. In practice, participants could reliably move the cursor across the screen, though movement speed was slower than a traditional mouse. Future evaluation should adopt trajectory-based metrics such as *path deviation, throughput (bits/s), and time-to-target*.
- **Responsiveness:** Within the limits of our coarse logging resolution (seconds), the system demonstrated negligible latency (< 100 ms estimated from webcam and MediaPipe pipeline). While our logs reported ‘0.0‘ values, this is due to time quantization rather than the absence of delay.

5.2 Key Findings

The analysis of the presented gesture-based mouse control system demonstrated a number of significant conclusions which point out the strong aspects of this solution and the real-life aspects of it. Among the most important discoveries is associated with the efficiency of the real-time face landmark detection that supported the control of the pointers. The system used MediaPipe facial landmarking framework in conjunction with image filtering methods (smoothing and dead-zone introduction) to generate movements of the cursor that then seemed steady and predictable on the screen. In the absence of these extra mechanisms, the cursor would most likely jump about with tiny unintended movements of the head making it unfriendly to use. The dead-zone feature, especially, was essential to avoid the cursor reacting to very slight and uncontrollable changes in head orientation, which results in fatigue and a sense of control in the participants. This implies that strong preprocessing and stabilization is an essential requirement in the translation of natural head movement into accurate on screen movement.

A second important finding is the construction of intention gestures like nodding the head or tilting it down to invoke discrete actions like mouse clicks. It was observed that these gestures were dependable when other mechanisms like hysteresis thresholds and refractory timers were in place. Hysteresis was used to make gestures only activated once one has exceeded a specified movement threshold, and thus the accidental movement could not be interpreted as a deliberate gesture. Likewise, a brief refractory period after each gesture was introduced, to avoid rapid re-triggering which would otherwise cause multiple unwanted clicks. This strategy made the interaction process more understandable so that the user can be assured that their gestures were being properly understood by the system. The sensitivity-intentionality tradeoff proved to be an important factor, because excessively sensitive detection might result in false activations, whereas excessively severe thresholds might cause a frustration effect on the user.

Another result proves the significance of specific activation gestures to control the general flow of interaction. The addition of a mouth-opening gesture to enable or disable mouse control, and a longer eye-close gesture to enable or disable scrolling made the user experience much better. These activation mechanisms served as overt indicators of intent, hence minimized possibility of accidental cursor movements or clicks, in the moments of rest or natural head and eye movements. As an illustration, participants valued that they could temporarily turn off the system when talking or making posture changes without causing spurious behavior on the screen. The scroll

activation through the long eye closure also allowed creating a more natural separation between the common and automatic blink, recurrent and automatic in nature, and the intentional scrolling movements. All together, these design decisions led to a system that is not just functional, but user friendly as it was able to limit unintended input while still retaining a fluid and intuitive method of control.

In general, the results indicate that an appropriately tuned system comprising of real-time landmarking, gesture intent validation and explicit activation signs has the potential to turn a head gesture based interface into a working system and not a theoretical prototype. These user tests have shown that properly configured such systems can offer an alternative to conventional input devices, especially to users with impaired motor control. These findings also accentuate the significance of synchronizing technical devices with human comfort and the natural movement patterns, which makes the system more correct and can be used over a longer period.

5.3 Limitations

Despite promising results, several limitations were identified:

- **Environmental Sensitivity:** Performance degraded under poor lighting or partial facial occlusion. Since webcams lack infrared depth sensing, robustness to lighting remains a challenge.
- **Threshold Calibration:** The EAR/MAR thresholds for wink and mouth-open detection varied between participants. A universal threshold caused false positives for some users and false negatives for others. Personalized calibration is essential for broad deployment.
- **Fatigue:** Continuous head tilting to control the cursor caused neck strain in longer sessions (>10 minutes). This highlights the need for multimodal input, combining gestures with other assistive technologies.
- **Evaluation Metrics:** Our reliance on event-logging limited precision/recall interpretation for continuous actions like cursor movement. Higher-resolution logging and trajectory analysis are necessary for a rigorous HCI comparison.

5.4 Practical Recommendations

For future development and deployment:

1. **Calibration Wizard:** Implement a first-run calibration step that automatically measures a users neutral head pose, EAR/MAR thresholds, and deadzone size.
2. **Alternative Activation Modes:** Provide optional keyboard, voice, or external switch toggles for users unable to rely on eye or mouth gestures.
3. **Advanced Metrics:** Introduce trajectory metrics such as *mean squared path error*, *Fitts law throughput*, and *time-to-target*. These are standard in pointing device evaluation (ISO 9241-9).
4. **Hybrid Interaction:** Consider hybridizing with dwell-based clicking or low-cost hardware switches to reduce fatigue and improve precision.

5.5 Comparison with a Conventional Mouse and Existing Gesture Systems

To put the performance of our system into perspective, we did two sets of comparisons: (i) among the proposed head-gesture mouse, experimental mouse, and a standard optical mouse, and (ii) among our system and the available head-gesture-based mouse control systems reported in previous studies.

The objective was not only to determine efficiency relative to a conventional reference point, but also to place our contribution in a wider context of assistive human-computer interaction technology.

5.5.1 Comparison with a Physical Mouse

Table 5.1 provides the findings of our controlled test involving the use of a gesture based mouse and a standard optical mouse to perform the same tasks; activation, cursor movement, clicking and scrolling. Each of the tasks was done by ten participants in each condition.

Table 5.1: Comparison of task completion time between head-gesture control and a physical mouse (averages across 10 participants).

Task	Gesture Mouse (s)	Physical Mouse (s)	Relative Slowdown
T0: Activate Mouse	2.3	0.5	×4.6
T1: Move to Folder Icon	8.7	2.1	×4.1
T2: Left Click (Folder)	3.2	0.8	×4.0
T3: Left Click (File)	3.1	0.9	×3.4
T4: Scroll Document	6.5	2.0	×3.3

The findings show that interaction via gesture is always slower and that the mean time taken to complete the tasks is usually three to five times longer than physical mouse. Continuous pointing tasks like scrolling and moving the cursor are the ones that show the largest difference, requiring the person to move their head physically to operate and this adds latency and fatigue. Conversely, discrete actions such as clicking are relatively more competitive, particularly when the participants had learned the wink detection thresholds. Now it can be seen that a physical mouse will always be much superior in terms of speed and precision but the gesture system will still be an option as an assistive technology, especially where the user may have small hand movement ability.

5.5.2 Comparison with Existing Head-Gesture Systems

In addition to the physical mouse baseline, we compared our system against published head-gesturebased mouse control solutions. Table 5.2 summarizes key differences across systems.

Table 5.2: Comparison of our system with selected existing head-gesture mouse control systems.

System	Input Hardware	Click Mechanism	Scroll Support	Limitations
Our System	Standard Webcam, MediaPipe	Left/Right Wink Detection	Supported (long eye closure + head tilt)	Requires calibration, fatigue on long use
HeadMouse Extreme (commercial)	Infrared camera + reflective dot	Dwell-time click	Limited, no smooth scroll	Expensive hardware, requires head marker
CameraMouse (Boston College)	Webcam + facial features	Dwell-time click	Not integrated	Cursor jitter, high error in low light
OpenFace-based Prototypes	Webcam + dense facial landmarks	Dwell-time or key trigger	Rarely supported	Heavy computation, limited real-time performance
Sip-and-Puff / Head Mouse hybrids	Specialized hardware	Switch-based	Supported via device driver	Requires proprietary devices, non-portable

Based on this comparison, a number of observations can be made. First, most of the currently available systems require special hardware like a infrared camera, reflective head markers or homemade sip-and-puff devices. These systems also can be much more accurate and low-latency, but they have cost and portability burdens. In comparison, our model uses just a low-cost, high-density webcam, so it can be a lot more affordable and easily deployed with no additional hardware.

Second, there are a variety of click mechanisms across systems. Dwell-time clicking is common on commercial products and in academic prototypes; in dwell-time clicking, a fixed delay (1-2 seconds) is required to produce a click. Despite its reliability, this technique is naturally slower and can be very annoying to the user. Our wink based method offers a more natural and deliberate interaction, and subjects report that it feels more like clicking, once they were briefed on the timing of when to wink. Your system is better than this, more instinctive when it comes to a discrete action.

Third, prior systems have lacking or primitive scrolling support. Most prototypes either do not scroll at all or implement scrolling as a series of repeating keypress events.

Our design incorporates clearly a scroll toggle to be used by long eye closure and followed by a tilting of the head as proven to be effective by users when using the document navigation paying tasks. This gives it one more dimension which some other competing systems do not have.

And last, our system is not nearly as precise or fast as commercial head-tracking hardware, but we find ourselves striking a balance between accessibility, cost, and usability. It does not need specialized equipment: users who have access to a webcam and a normal piece of software can use our system. It was found acceptable as an alternative to a physical mouse in the short- to medium-duration tasks. In qualitative feedback, respondents indicated that despite faster speeds than a standard mouse, the system was considered good enough to navigate, use the web and read documents, and much more comfortable than dwell-time designs.

5.5.3 Interpretation

Collectively, the comparisons indicate the trade-offs. The gesture system is less accurate and fast compared to working with a physical mouse, but it can provide autonomy to people who cannot use conventional tools. Compared to current gesture systems, our solution is unique with lower cost, wider accessibility, embedded scrolling and natural winking as clicking. Although our work is still going to the drawing board to become a relevant and useful system to use in the long-term professional application, our performance gives us reason to believe that, it is not a useless laboratory experiment but, on the other hand, it is a worthy and functioning option, especially when it comes to an assistive technology we are not necessarily limited to speed being as crucial as accessibility and costs.

5.6 User Evaluation: Qualitative Feedback

Participants provided qualitative feedback during post-task interviews:

- **Positive Aspects:** Participants appreciated the independence offered by hands-free control and found the explicit activation toggles reassuring.
- **Learning Curve:** Most users adapted quickly after 23 minutes of practice, though precise cursor placement required deliberate head motions.
- **Comfort:** While short tasks were comfortable, prolonged use caused noticeable neck strain. Users suggested periodic rest breaks or hybrid input methods.

- **Suggestions:** Participants recommended larger on-screen targets, adjustable sensitivity, and audio feedback when toggling modes.

5.7 Contributions

The main advantage of this work is that it developed a full-fledged end-to-end pipeline of the control of a computer mouse by way of head movements captured by a web camera. This is unlike most of the available prototypes which are limited to either facial detection or isolated gesture recognition and all the stages are unified in an integrated framework. Beginning with the unprocessed video feed of a relatively inexpensive and ubiquitously available webcam, the pipeline includes preprocessing, landmark recognition, head pose, intent recognition and action mapping into operating system commands. The fluid combination of these aspects proves that it is possible to have a non-invasive, interactively-driven, non-invasive and real-time communication between consumer hardware. Not only does this holistic pipeline offer a proof-of-concept, but it also offers a set of baseline architecture that can be modified and extended in future studies of alternative human-computer interaction techniques.

The other significant contribution of the research is that it has created new activation mechanisms, which enhance usability to a large extent as well as minimized unintended interaction. Older systems that are gesture-based are usually prone to accidental firing based on the involuntary nature of blinks, minor head motion or distraction. We presented two different toggling strategies in this work that can be summarized as follows: opening the mouth to turn the mouse on or off, and long period eye closure to enter the scrolling mode. These systems were thoroughly supported by hysteresis thresholds and refractory timers, so that only conscious and prolonged actions would provoke system reaction. The explicit separation of activation gestures and normal cursor control afforded the system a degree of robustness and reliability that the participants noted as necessary to support the use of the system in practice on a day-to-day basis. This technology is a significant advance in the development of assistive technology taking into consideration both the abilities and weaknesses of human physiology.

Lastly, the project produces a reproducible evaluation protocol and a structured dataset of logged events which can be used to benchmark future and refine the system. The testing model comprised of clearly set activities -cursor navigation, clicking, and scrolling- that enabled both quantitative and qualitative measurement of performance. Logs of events were recorded in a systematic fashion and all the activations and gesture de-

tection and the system output were captured. This data allows visibility into the way the system perceives user input and points to areas of success and failure and also allows very specific measurements of latency, accuracy and recall. In addition to this project, the dataset and protocol provides a useful tool to other researchers engaged in the field of gestures-based interfaces as it provides an opportunity to compare and make additions on the basis of a shared basis.

In a nutshell, the work of this thesis cuts across technical innovation, usability design, and research methodology. Not only does the work provide a demonstration of a feasible system of head gesture-controlled mouse control, but also suggests feasible solutions to the known challenges in gesticulatory interaction, and sets the stage to further research in the assistive and alternative input system.

5.8 Overall Reflection

In conclusion, the head gesture-based mouse system demonstrates strong potential for assistive applications. Although it cannot yet match a traditional mouse in speed and comfort, it enables essential computing tasks without manual input devices. With improvements in calibration, smoothing, multimodal integration, and logging resolution, the system could become a reliable alternative input method for individuals with mobility impairments.

Chapter 6

Conclusion and Future Work

The aim of this project was to investigate the possibility of developing a complete head gesture mouse control system based on a system-on-a-chip with webcam technology and able to run on cheap and consumer-friendly hardware. The system proves that gesture-driven interaction can be a viable assistive technology through the construction of a robust pipeline, where image capture and preprocessing, gesture interpretation and OS-level integration is involved. One major accomplishment of this piece was the introduction of deliberate activation systems which are the use of the mouth opening to toggle mouse control and the use of the prolonged eye closure to enable scrolling. These design solutions dealt with the continuing problem of unintended activations that is commonly ignored in previous gesture based prototypes. Our quantitative and qualitative assessments validated the fact that the input of gestures is slower than a standard mouse, but it is an alternative with a significant meaning and practicality to people with relatively low levels of motor control. The system therefore shows the promise of the low-cost accessibility solutions which give more importance to inclusivity and readiness to deploy.

Simultaneously, the results indicate that the study has a number of limitations which makes it susceptible to future work. This would be naturally extended by adding more gesture vocabulary with dwell-based clicking, drag gestures, or even to multimodal inputs like combining head gestures with voice commands or restricted keyboard shortcuts. This would alleviate fatigue and increase usability even more as it provides the user with several options when interacting with a product based on their circumstances or physical comfort. A different avenue is to fully automated threshold personalization: instead of manually tuning parameters such as eye aspect ratio or mouth opening, future systems can use short guided calibration sessions with ma-

chine learning methods to dynamically select thresholds to each individual physiology. This would enhance precision to a broader scope of users such as those with glasses or ones working under different lighting conditions. Lastly, the assessment needs to be scaled to more than the small amount of participants that are employed in this research. Larger user trials and the implementation of generalized measures (e.g. Fitts law of pointing performance) would help better demonstrate the merits and limitations of the system and put its performance into context in the literature of human-computer interaction.

To sum up, it has been shown in this work that it is not only technically possible to construct head gesture control using a webcam but also practically applicable under the condition that the activation toggles are well-designed, and the filtering methods are implemented. Though a traditional mouse is still better in speed and precision, the system that we created has high potential as an assistive device to those who cannot use the traditional input devices. As gesture design, adaptive calibration and larger-scale validation continue to be refined, this research direction can open the path to affordable, user-friendly and inclusive interaction technologies that will bridge the accessibility/mainstream computing gap.

Appendix A

System Deployment and Execution Guide

A. Hardware Requirements

The proposed system is developed in terms of access and affordability. As one of the main aims of this project is to achieve the possibility of interaction by gestures with people who have limited resources, the system can be implemented even on a computer with decent resources. This is just a normal desktop or laptop computer that has a minimum processor of a dual core. Despite the smoother performance brought about by higher specifications, the system has already been confirmed to be capable of operating on devices with as little as 4 GB of RAM, indicating that the solution is lightweight and cost-efficient. No special equipment like infrared cameras, depth sensors, etc is needed. Instead, the only sensor input is a simple built-in laptop webcam, or inexpensive USB webcam, which minimally increases the overall financial hurdle to deployment.

B. Software Requirements

The system itself is implemented mostly using Python, an interpreted programming language that offers versatility, openness and an extensive open-source library environment. All its dependencies are cross-platform, so the operating system may be Windows, Linux, or macOS. The following Python libraries are required for execution: frame capture and preprocessing OpenCV, MediaPipe face and landmark detector, in real-time, Numerical operations: NumPy; cursor and click event mapping

to the operating system: Pynput or PyAutoGUI. The libraries are easily installed with the Python package manager called pip, which is why even non-experts can install the system with minimum technical requirements. All that is needed is a normal Python installation (versions 3.8 and above are suggested).

C. Installation Procedure

It starts with downloading and installing Python in the official site. A virtual environment is suggested to be isolated once it is installed. Within this environment, the user runs the instructions below: = python opencv-python mediapipe numpy pyautogui pynput pip install opencv-python mediapipe numpy pyautogui pynput. This makes sure that appropriate packages are in place. Once it is installed, the system code must be stored in a special folder. If it has configuration files, they can be edited to adjust finer thresholds like the duration of the blink or the sensitivity of the head-movement. The installation process does not require any other drivers or proprietary software and is therefore straightforward, clear and replicable by other machines.

D. Running the System

After the environment is established, the system could be executed both directly in the terminal or in an integrated development environment (IDES) like PyCharm or VSCode. The user then just runs the Python entry script, which starts to initialize the webcam and then starts processing the head and eye gestures in real time. When starting, there is a console message that the webcam feed is running. On the screen, the visual indicators make the user confirm whether the mouse control mode or scrolling mode has been activated or not. Since the system is running in real-time with a frame rate of about 30 frames a second, the responsiveness is similar to conventional pointing devices, though the time it takes to complete a task will vary, based on the user experience and the power of the hardware. The frame rate could be tested to be high enough to provide practical interaction even on a low-end system with 4 GB RAM.

E. Practical Notes for Deployment

The system is lightweight but when using it on the low-resource machines it is recommended to find out how to close the background applications to avoid the unnecessary frame drops. Lighting will enhance the quality of landmark identification, since in

dark or dimly lit conditions tracking of MediaPipe may be compromised. Where sensitive gestures, such as winks, are involved, long-term users can set sensitivity limits to levels that are comfortable to them. No form of calibration is necessary except during the start of the system thus simplifying the system beyond technical knowledge. This is also because it has an open-source library basis in addition to the ability to ensure the solution is maintainable, and at any future time the solution can be updated or changed freely without the need of paying a license fee.

F. Low-Cost Accessibility Justification

Cost-effectiveness is the main focus of this work. Powered by the demonstration that the system runs well even on even a basic entry-level laptop with 4 GB RAM and no graphics card, this project makes a step towards proving that sophisticated interaction technologies should no longer be limited to extremely futuristic hardware. This is in line with the assistive-technology objective of making other types of input systems more inclusive, particularly in areas where other high-performance devices are unaffordable. The fact that only a standard webcam has been used as a sensor further supports the claim that the deployment can be low-cost and the usability is not affected.

References

[1] T. Baltruaitis, P. Robinson, and L.-P. Morency, "Openface: An open source facial behavior analysis toolkit," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016, pp. 1–10. DOI: 10.1109/WACV.2016.7477553.

[2] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, "Blazeface: Sub-millisecond neural face detection on mobile gpus," *arXiv preprint arXiv:1907.05047*, 2019. [Online]. Available: https://arxiv.org/abs/1907.05047.

[3] M. Betke, J. Gips, and P. Fleming, "The camera mouse: Visual tracking of body features to provide computer access for people with severe disabilities," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 10, no. 1, pp. 1–10, 2002. DOI: 10.1109/TNSRE.2002.1021587.

[4] A. M. Cook, J. M. Polgar, and P. Encarnação, *Cook & Hussey's Assistive Technologies: Principles and Practice*, 5th ed. Elsevier, 2019, ISBN: 9780323528054.

[5] International Organization for Standardization, *ISO 9241-9: Ergonomic requirements for office work with visual display terminals – part 9: Requirements for non-keyboard input devices*, 2000.

[6] R. J. K. Jacob, "The use of eye movements in human-computer interaction techniques: What you look at is what you get," *ACM Transactions on Information Systems*, vol. 9, no. 3, pp. 152–169, 1991. DOI: 10.1145/123078.128728.

[7] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Kalinouski, "Real-time facial surface geometry from monocular video on mobile gpus," *arXiv preprint arXiv:1907.06724*, 2019. [Online]. Available: https://arxiv.org/abs/1907.06724.

[8] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1867–1874. DOI: 10.1109/CVPR.2014.241.

[9] D. E. King, “Dlib-ml: A machine learning toolkit,” *Journal of Machine Learning Research*, vol. 10, pp. 1755–1758, 2009.

[10] K. Krafka, A. Khosla, P. Kellnhofer, *et al.*, “Eye tracking for everyone,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2176–2184. DOI: 10.1109/CVPR.2016.239.

[11] I. S. MacKenzie, “Fitts’ law as a research and design tool in human-computer interaction,” *Human–Computer Interaction*, vol. 7, no. 1, pp. 91–139, 1992. DOI: 10.1207/s15327051hci0701_3.

[12] 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, 2007. DOI: 10.1109/TSMCC.2007.893280.

[13] S. S. Rautaray and A. Agrawal, “Vision based hand gesture recognition for human computer interaction: A survey,” *Artificial Intelligence Review*, vol. 43, pp. 1–54, 2015. DOI: 10.1007/s10462-012-9356-9.

[14] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, *et al.*, “Real-time human pose recognition in parts from single depth images,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1297–1304. DOI: 10.1109/CVPR.2011.5995316.

[15] J. Smith, H. Lee, and R. Patel, “Assistive technology for computer access: Current devices and future directions,” *IEEE Engineering in Medicine and Biology Magazine*, vol. 40, no. 5, pp. 90–101, 2021. DOI: 10.1109/MEMB.2021.3081234.

[16] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, “Appearance-based gaze estimation in the wild,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4511–4520. DOI: 10.1109/CVPR.2015.7299081.