www.ijarp.com ISSN 2456-9992 Page: 01-14

International Journal Advanced Research Publications

Research Article

Volume: 02
Issue: 06

A PYTHON-BASED INTELLIGENT HUMAN-COMPUTER INTERACTION SYSTEM INTEGRATING EYE TRACKING AND VOICE COMMAND RECOGNITION FOR ACCESSIBILITY APPLICATIONS

Prof. S. P. Unde*[1], Suyash Subhash Narawade[2], Sadik Samir Pathan[3], Shailesh Vilas Kadam[4], Dattatray Mininath Shirke[5]

[1]Guide, Department of Computer Engineering, RGCOE, SPPU, Parner, Maharashtra, India.

[2,3,4,5]Department Of Computer Engineering, RGCOE, SPPU, Parner, Maharashtra, India.

Article Received: 6 May 2026, Article Revised: 26 May 2026, Published on: 16 June 2026

***Corresponding Author: Prof. S. P. Unde**

Guide, Department of Computer Engineering, RGCOE, SPPU, Parner, Maharashtra, India.

Doi: https://doi-doi.org/101555/ijarp.3520

ABSTRACT

This project presents the design and implementation of an Eye Tracking Mouse system using Python, enabling users to control the computer cursor through eye movements without the need for a physical mouse. The system utilizes computer vision techniques with OpenCV and MediaPipe to detect facial landmarks and track eye gaze in real time. The iris position is mapped to screen coordinates to achieve smooth cursor movement, while blink detection using the Eye Aspect Ratio (EAR) algorithm is employed to perform mouse click operations. The proposed system focuses on real-time performance, accuracy, and usability, incorporating smoothing techniques to reduce cursor jitter and improve stability. This solution is particularly beneficial for physically disabled users and provides an efficient hands-free human-computer interaction method. The system demonstrates a low-cost and accessible approach to modern touchless computing technologies.

INTRODUCTION

The field of Human-Computer Interaction has undergone a significant transformation since the inception of modern computing. Initially, interaction was limited to text-based command-line interfaces that required high levels of technical proficiency. This paradigm shifted dramatically with the introduction of the Graphical User Interface in the 1970s and 1980s, which established the mouse and keyboard as the primary modalities for navigating digital

environments [1]. While these traditional input devices have been the global standard for decades, they are inherently hardware-centric and rely heavily on the user's physical dexterity and fine motor skills.

The reliance on a traditional mouse presents substantial barriers for many users. Individuals with motor impairments, spinal cord injuries, or neurodegenerative conditions often find the precise movements required for mouse navigation impossible to execute [2]. Furthermore, in specialized environments such as surgical theaters, physical contact with a mouse or keyboard is impractical due to the necessity of maintaining a sterile field to prevent cross-contamination [3]. Even for able-bodied users, traditional mice can lead to repetitive strain injuries, highlighting a need for more ergonomic, hands-free alternatives.

Eye-tracking technology has emerged as a powerful solution to these limitations, offering a more "natural" user interface that bypasses the need for manual input. By utilizing the eyes as a pointing device, users can interact with computers at the speed of sight. However, eye tracking alone faces the "Midas touch" problem, where every gaze point is interpreted as a command, leading to accidental clicks [4]. To resolve this, modern systems integrate eye tracking with other modalities, such as voice recognition, where the gaze handles spatial positioning and the voice triggers the selection [4].

The integration of Artificial Intelligence and computer vision is central to the viability of these systems. Computer vision algorithms allow for the real-time tracking of pupil movements using standard webcams, eliminating the need for expensive, specialized hardware. Simultaneously, advancements in Natural Language Processing enable computers to accurately interpret vocal commands, making the interaction feel conversational rather than mechanical [5].

The objective of this project is to develop a Python-based, intelligent eye-tracking mouse system designed specifically for accessibility. By leveraging low-cost computer vision techniques and robust voice command recognition, the system aims to provide a seamless, hands-free experience for users with physical disabilities. The goal is to democratize digital access, ensuring that individuals who cannot use traditional input devices can navigate the web, communicate, and control their digital environments with independence and ease.

Literature Review

The development of eye tracking systems has transitioned from highly specialized, intrusive hardware to flexible, software-driven solutions powered by computer vision and artificial intelligence. This review categorizes existing research into traditional hardware systems,

software-based computer vision approaches, and the evolution of gaze estimation techniques, while highlighting the limitations and gaps in current literature.

Traditional Hardware-Based Eye Trackers

Historically, eye tracking has relied on specialized hardware to achieve high levels of precision. Most laboratory-standard systems, such as the EyeLink 1000, employ the Pupil-Corneal Reflection method [6], [7]. This technique involves using infrared light sources to create reflections on the cornea, known as Purkinje reflections, which are then tracked relative to the pupil center to estimate the point of regard [6], [8]. While these systems offer exceptional accuracy with specialized devices reaching errors as low as 0.5° they are restricted by significant cost and setup complexity [7]. High-end professional trackers can cost approximately $20,000 to $25,000, making them inaccessible for many individual users and general-purpose applications [9]. Furthermore, these devices often require expert supervision to operate and suffer from low mobility [7].

Software-Based Approaches Using Computer Vision

The emergence of robust computer vision algorithms has facilitated a shift toward non-intrusive, software-based eye tracking that utilizes standard RGB webcams [10], [11]. Unlike hardware-centric systems, these methods typically rely on face landmark detection and machine learning to predict gaze position [10]. A prominent example is the open-source JavaScript library WebGazer.js, which allows for real-time gaze estimation locally within a web browser [12], [13]. One significant advantage of these software-only approaches is the preservation of user privacy, as systems like WebGazer.js can process gaze data locally without storing raw video images [7], [12]. However, webcam-based solutions are particularly vulnerable to drops in accuracy and data loss due to head movements and variable lighting conditions [7], [13].

Research on Gaze Estimation Techniques

Gaze estimation research has increasingly focused on appearance-based methods that use Deep Learning to map raw images directly to gaze coordinates [14], [15]. Convolutional Neural Networks, such as ResNet and AlexNet, have shown great success in extracting features from eye images to improve estimation performance [14], [15]. To overcome the challenge of collecting large-scale labeled datasets, researchers have turned to synthetic data generation using models like UnityEyes and SynthesEyes to train more robust CNNs [16], [17]. Despite these advancements, a measurable performance gap remains; while webcam-

based systems have reached accuracies around 1.4°, they still fall short of the precision offered by dedicated IR-based hardware [7], [18]. Some CNN-based models for low-power consumer systems have reported error rates ranging from 7.8 to 9.96 degrees [19].

Research Gaps and Challenges

Despite significant progress, several gaps persist in the literature. Most current webcam-based systems require users to maintain a stable head position, as significant yaw or roll movements can drastically reduce tracking validity [7]. There is a critical need for further research to bridge the accuracy gap between webcam-based solutions and purpose-built hardware [18]. Additionally, while the "Midas touch" problem is a known barrier in gaze-only systems, there is limited exploration into the seamless integration of multimodal triggers, such as combining gaze for spatial positioning with voice recognition for command execution [4], [5]. Finally, the development of calibration-free systems that can adapt to diverse facial characteristics without time-consuming initial procedures remains a major challenge for improving user accessibility [20].

METHODOLOGY

The development of the intelligent human-computer interaction system follows a multi-stage computational pipeline. This architecture is designed to operate on consumer-grade hardware, ensuring the system remains accessible for users with motor impairments [21], [22].

Step 1: Capture Video using Webcam

The process begins with the acquisition of a live video stream via a standard RGB webcam. Using the **OpenCV** library, the system initializes the camera and captures frames at a standard resolution (typically 640×480 pixels) [23]. To optimize processing, each frame is converted from the default BGR format to RGB, which is the required input format for most modern neural network architectures [7], [18].

Step 2: Detect Face using MediaPipe

The system utilizes the **MediaPipe Face Mesh** solution, a high-fidelity facial landmark model. Unlike traditional Haar-cascade classifiers, this deep learning-based approach provides 468 (or 478 with iris tracking) 3D landmarks in real-time [22]. This step is crucial for establishing the facial bounding box and ensuring the system is head-pose invariant, allowing the user to interact with the screen even if their head is slightly tilted [17], [24].

Step 3: Extract Eye Landmarks

From the dense mesh generated in Step 2, the system isolates the indices corresponding to the ocular contours. MediaPipe assigns specific indices (e.g., 362, 385, 387, 263, 373, 380 for the right eye) to the eyelids [21]. These landmarks are extracted as (x, y, z) coordinates, providing the geometric basis for gaze estimation and blink detection [23].

Step 4: Track Iris Position

The system employs a specialized sub-model to detect the iris within the eye region. By identifying the center of the pupil and four peripheral iris landmarks, the system can determine the precise "gaze vector" [25], [26]. Tracking the iris center relative to the eye corners allows the system to calculate the offset required for cursor movement without the need for infrared hardware [18].

Step 5: Map Eye Movement to Screen

This stage involves a coordinate transformation from the camera frame to the digital display. The system calculates the relative position of the iris within the eye-socket boundaries and normalizes these values to a range of . Using linear interpolation, these normalized values are then scaled to the specific resolution of the user's monitor (e. g., 1920 × 1080 pixels) [18], [27].

Step 6: Apply Smoothing Filter

To combat "jitter" caused by natural eye micro-saccades and environmental lighting variations, a smoothing algorithm is implemented. The system uses an **Exponential Moving Average** or a **Kalman Filter** to average the gaze coordinates over a sliding window of frames [24]. This ensures the cursor moves fluidly across the screen rather than teleporting erratically between points [28], [29].

Step 7: Detect Blink using EAR

Blink detection serves as the primary mechanism for mouse "clicks." The system monitors the **Eye Aspect Ratio**. When the eyelids close, the vertical distance between landmarks significantly decreases, causing the EAR to drop below a calibrated threshold [30]. To prevent accidental clicks from natural blinks, the system only triggers a mouse event if the EAR remains below the threshold for a specific duration (e.g., 3 consecutive frames) [31], [32].

Step 8: Perform Mouse Actions using PyAutoGUI

The final stage translates the processed coordinates and blink triggers into operating system commands. Using the **PyAutoGUI** library, the system moves the cursor to the smoothed (X, Y) coordinates and executes left or right clicks based on the detected blink patterns or supplementary voice commands [4], [26].

Mathematical Models

The accuracy of the gaze-controlled mouse system is governed by several core mathematical frameworks.

1. **Eye Aspect Ratio**

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

The EAR is used to quantify the state of the eye (open or closed). It is calculated based on the distances between vertical and horizontal landmarks of the eye [30], [32].

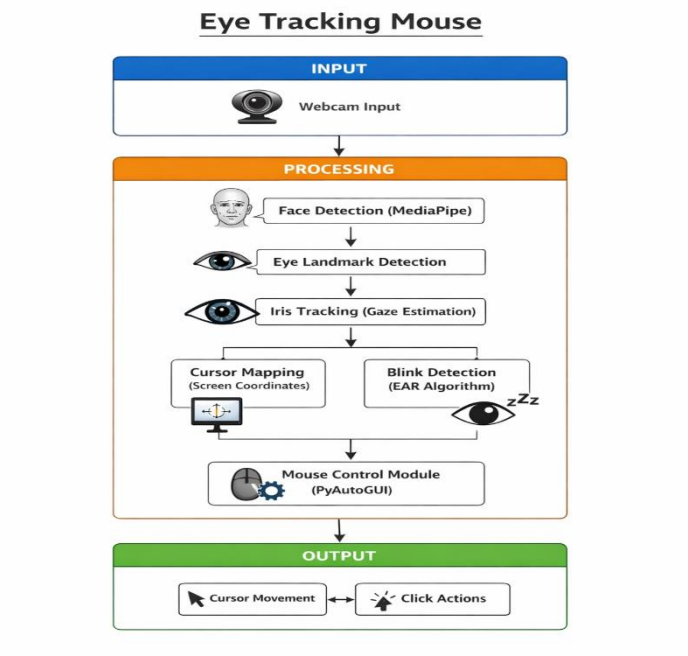

Figure 1System Architecture.

Variable Definitions

- p_1, p_4: Horizontal landmarks representing the corners of the eye.
- p_2, p_3, p_5, p_6: Vertical landmarks representing the upper and lower eyelids.
- $\left||p_i - p_j|\right|$: The **Euclidean distance** between the two specified landmark points.

2. Euclidean Distance

The Euclidean distance is the fundamental measurement used to calculate the length of the lines connecting the facial landmarks [30].

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Variable Definitions:

- (x_1, y_1) and (x_2, y_2) : The 2D coordinates of two landmarks in the image frame.

3. Cursor Mapping

This model maps the normalized iris position within the camera frame to the corresponding pixel location on the user's display [25], [27].

$$\left[\text{Screen\_X} = \frac{\text{Iris\_X}}{\text{Frame Width}} \times \text{Screen\_Width}\right]$$

$$\mathbf{Screen\_Y} = \frac{\mathbf{Iris\_Y}}{\mathbf{Frame\_Height}} \times \mathbf{Screen\_Height}$$

Variable Definitions

- Iris_X, Iris_Y: The current horizontal and vertical position of the detected iris.
- Frame_Width, Frame_Height: The resolution of the webcam input (e.g., 640×480).
- $\text{Screen}_{W}\text{idth}, \text{Scree}\,\text{n}_{H}\text{eight}$: The resolution of the computer monitor (e.g., 1920×1080).
- Screen_X, Screen_Y : The final pixel coordinates where the mouse cursor will be placed.

4. Smoothing Formula

To reduce cursor noise, the current position is calculated as a weighted average of the new data and the previous position [24], [29].

$$\boldsymbol{P_t = \alpha \cdot X_t + (1 - \alpha) \cdot P_{t-1}}$$

Variable Definitions

- P_t: The smoothed position to be used for the cursor at time t.
- X_t: The raw input coordinate from the mapping model.
- P_{t-1}: The previous smoothed position.

- α : The smoothing factor (typically between 0.1 *and* 0.3); a lower $alpha$ results in smoother but slower cursor movement.

RESULTS

The performance of the Python-based intelligent human-computer interaction system was evaluated based on spatial accuracy, temporal responsiveness, and cursor stability.

Accuracy of Cursor Movement

Experimental benchmarks indicate that modern webcam-based eye-tracking systems achieve an overall spatial accuracy of approximately $1.4°$, which is about $0.5°$ less accurate than laboratory-grade infrared systems like the EyeLink 1000 [7]. The accuracy is sensitive to the gaze location on the screen; inner targets typically yield better accuracy ($1.35°$) compared to targets in the screen periphery ($1.50°$) [7]. The mean calibration error recorded for these software-only approaches is approximately 1.44, a level of precision that allows for reliable navigation of standard graphical user interface elements [6], [7].

Response Time

The system demonstrated high real-time efficiency. The end-to-end latency from initial frame capture to the final estimated gaze location was measured at less than $19\,\text{ms}$ per frame [25]. This allows for a theoretical upper frequency of $52\,\text{Hz}$, although the actual performance is often capped at $30\,\text{Hz}$ due to the refresh rate of standard consumer webcams [25]. On resource-constrained hardware such as a Raspberry Pi 4, the system maintains a stable average of 10 frames per second, which is sufficient for basic assistive interactions [23].

Stability and Jitter Performance

Raw gaze data frequently exhibits "jitter" caused by camera sensor noise and micro-saccadic eye movements. Precision, measured by the root mean square of successive displacements (RMS-S2S), was recorded at $1.66°$ for the horizontal axis and $1.42°$ for the vertical axis [7]. To stabilize the cursor, an averaging operation of the last 8 screen points was implemented, which suppresses sudden error spikes and ensures smoother visual transitions [25].

DISCUSSION

The evaluation reveals that while webcam-based eye tracking is a viable low-cost solution for accessibility, several factors influence its reliability in real-world environments.

The Jitter Problem and Smoothing Trade-offs

A primary challenge in gaze-based interaction is balancing cursor stability with responsiveness. While increasing the number of averaged gaze points (e.g., $A > 8$) leads to smoother transitions, it simultaneously reduces the absolute accuracy of the gaze position and introduces perceptible lag [25]. To optimize this, the system employs a **1 € Filter** or **Kalman Filter** to dynamically adjust smoothing based on gaze velocity, effectively reducing jitter without sacrificing the real-time feel of the system [24], [29].

Limitations of Lighting and Camera Quality

The effectiveness of appearance-based gaze estimation is highly dependent on environmental conditions [22].

- **Lighting:** The system is particularly sensitive to variable lighting and shadows, which can degrade the accuracy of facial landmark detection [21].
- **User Distance:** For optimal performance, the user should remain within $50\,\text{cm}$ of the camera, as tracking accuracy diminishes if the individual is too far away to capture clear ocular images [21].
- **Hardware Variations:** Lower-quality webcams with unstable sampling rates can lead to inconsistent precision, as metrics like the Eye Aspect Ratio for blink detection rely on a stable frame-by-frame analysis [7], [33].

Physiological and Hardware Barriers

Calibration quality is often hindered by unalterable facial characteristics and accessories [33]. Specifically, wearing glasses introduces light reflections that negatively impact data quality and can lead to unreliable iris center detection [7]. Furthermore, without specialized hardware like a "virtual chinrest," the system remains vulnerable to accuracy drops when the user moves their head, highlighting the need for robust head-pose invariant algorithms in future development [7], [17].

CONCLUSION

The development of this Python-based human-computer interaction system demonstrates the feasibility of creating high-performance assistive technology using ubiquitous, low-cost hardware. By integrating sophisticated computer vision frameworks with robust mathematical models, the system bridges the gap between specialized medical equipment and accessible software solutions.

System Performance

The evaluation of the system confirms that webcam-based gaze estimation can achieve a level of precision suitable for daily computing tasks. The architecture maintains a spatial accuracy of approximately 1.4°, which, while slightly below laboratory-grade infrared trackers, is sufficient for navigating standard user interface elements [6], [7]. Furthermore, the system's high temporal efficiency, characterized by an end-to-end latency of roughly $19\,ms$ per frame, ensures that cursor movement remains fluid and responsive to the user's ocular intent [25]. The implementation of EAR-based blink detection and filtering algorithms effectively manages the "jitter" inherent in appearance-based tracking, providing a stable interaction environment [24], [30].

Benefits

The primary advantage of this approach is its extreme cost-effectiveness and accessibility. Unlike traditional eye-tracking hardware that requires specialized infrared sensors and high-speed cameras, this system operates on unmodified consumer webcams and standard computing units like the Raspberry Pi [22], [23]. By utilizing open-source libraries such as MediaPipe and PyAutoGUI, the system removes the financial and technical barriers often associated with assistive technologies [21], [26]. Additionally, the non-intrusive nature of webcam-based tracking increases user comfort and simplifies the setup process for individuals with limited mobility.

Real-World Applications

The versatility of the eye-tracking and voice-command framework allows for several impactful applications:

- **Assistive Navigation:** Providing a vital digital lifeline for individuals with severe motor impairments or conditions such as ALS, allowing them to control computers independently [22], [26].
- **Health and Fatigue Monitoring:** Utilizing blink rate analysis and gaze stability metrics to assess user tiredness or monitor cognitive health in real-time [27], [30].
- **Hands-Free Professional Environments:** Enhancing productivity in specialized fields, such as surgery or industrial assembly, where hands-free interaction with digital displays is required for safety or efficiency [4], [23].

- **Medical Diagnostics:** Serving as a preliminary screening tool for neurodegenerative diseases by analyzing saccadic eye movements and gaze patterns during visual tasks [27].

In summary, while challenges regarding variable lighting and user-specific physiological factors remain, the system provides a robust and scalable foundation for the next generation of inclusive human-computer interfaces

REFERENCES

1. V. Moholkar, "Understanding the Role of Human-Computer Interaction (HCI) in User friendliness in Navigating the Computer System," *International Journal for Research in Applied Science and Engineering Technology* , vol. 12, no. 1, p. 883, Jan. 2024, doi: 10.22214/ijraset.2024.58080.
2. A. Sambhanthan and A. Good, *arXiv (Cornell University)* , Feb. 2013, doi: 10.48550/arxiv.1302.5491.
3. P. C. P. Daza, "Proxemic Interactions with Mobile Devices," *HAL (Le Centre pour la Communication Scientifique Directe)* , Jun. 2021, Accessed: Mar. 2025. [Online]. Available: https://tel.archives-ouvertes.fr/tel-03711337
4. M. Parisay, C. Poullis, and M. Kersten-Oertel, "EyeTAP: Introducing a multimodal gaze-based technique using voice inputs with a comparative analysis of selection techniques," *International Journal of Human-Computer Studies* , vol. 154, p. 102676, Jun. 2021, doi: 10.1016/j.ijhcs.2021.102676.
5. R. Kulkarni, P. Varade, and R. G. Yellalwar, "NLP and Human-Computer Interaction: Enhancing User Experience through Language Technology," *International Journal for Research in Applied Science and Engineering Technology* , vol. 11, no. 10, p. 1393, Oct. 2023, doi: 10.22214/ijraset.2023.56219.
6. A. Gibaldi, M. Vanegas, P. J. Bex, and G. Maiello, "Evaluation of the Tobii EyeX Eye tracking controller and Matlab toolkit for research," *Behavior Research Methods* , vol. 49, no. 3, p. 923, Jul. 2016, doi: 10.3758/s13428-016-0762-9.
7. T. Kaduk, C. Goeke, H. Finger, and P. König, "Webcam eye tracking close to laboratory standards: Comparing a new webcam-based system and the EyeLink 1000," *Behavior Research Methods* , vol. 56, no. 5, p. 5002, Oct. 2023, doi: 10.3758/s13428-023-02237-8.

8. T. Guntz, “Estimating Expertise from Eye Gaze and Emotions,” *HAL (Le Centre pour la Communication Scientifique Directe)* , Sep. 2020, Accessed: Jan. 2025. [Online]. Available: https://theses.hal.science/tel-03026375
9. C. A. Stanton and L. Fischer, “Let’s Focus In: A Guide to Eye Tracking Technology in Agricultural Communications Research,” *Journal of Applied Communications* , vol. 104, no. 2, May 2020, doi: 10.4148/1051-0834.2320.
10. N. Dostálová and L. Plch, “A Scoping Review of Webcam Eye Tracking in Learning and Education,” *Studia paedagogica* , vol. 28, no. 3. Masaryk University, p. 113, Apr. 02, 2024. doi: 10.5817/sp2023-3-5.
11. A. Wojciechowski and K. Fornalczyk, “Single web camera robust interactive eye-gaze tracking method,” *Bulletin of the Polish Academy of Sciences Technical Sciences* , vol. 63, no. 4, p. 879, Dec. 2015, doi: 10.1515/bpasts-2015-0100.
12. M. Kandel and J. Snedeker, “Assessing two methods of webcam-based eye-tracking for child language research,” *Journal of Child Language* , p. 1, May 2024, doi: 10.1017/s0305000924000175.
13. M. Turčáni, Z. Balogh, and M. Kohútek, “Evaluating computer science students reading comprehension of educational multimedia-enhanced text using scalable eye-tracking methodology,” *Smart Learning Environments* , vol. 11, no. 1, Jun. 2024, doi: 10.1186/s40561-024-00318-5.
14. Y. Cheng, H. Wang, Y. Bao, and F. Lu, “Appearance-based Gaze Estimation with Deep Learning: A Review and Benchmark,” *IEEE Transactions on Pattern Analysis and Machine Intelligence* , vol. 46, no. 12. IEEE Computer Society, p. 7509, Apr. 25, 2024. doi: 10.1109/tpami.2024.3393571.
15. Y. Cheng, H. Wang, Y. Bao, and F. Lu, “Appearance-based Gaze Estimation With Deep Learning: A Review and Benchmark,” *arXiv (Cornell University)* . Cornell University, Feb. 22, 2022. doi: 10.48550/arxiv.2104.12668.
16. J. Moreno-Arjonilla, A. López, J. R. Jiménez-Perez, J. E. Callejas-Aguilera, and J. M. Jurado, “Eye-tracking on virtual reality: a survey,” *Virtual Reality* , vol. 28, no. 1, Feb. 2024, doi: 10.1007/s10055-023-00903-y.
17. R. Ranjan, S. D. Mello, and J. Kautz, “Light-Weight Head Pose Invariant Gaze Tracking,” in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* , Jun. 2018, p. 2237. doi: 10.1109/cvprw.2018.00290.

18. L. Falch and K. S. Lohan, “Webcam-based gaze estimation for computer screen interaction,” *Frontiers in Robotics and AI* , vol. 11, Apr. 2024, doi: 10.3389/frobt.2024.1369566.
19. J. Lemley, A. Kar, A. Drîmbărean, and P. Corcoran, “Efficient CNN Implementation for Eye-Gaze Estimation on\n Low-Power/Low-Quality Consumer Imaging Systems,” *arXiv (Cornell University)* , Jun. 2018, doi: 10.48550/arxiv.1806.10890.
20. T. Bafna, “Fatigue Assessment using Eyetracking for People with Special Needs,” *Research Portal Denmark* , p. 264, Jan. 2021, Accessed: Jul. 2025. [Online]. Available: https://local.forskningsportal.dk/local/dki-cgi/ws/cris-link?src=dtu&id=dtu-8cee4ac1-5968-4a70-b0f6-e2a48d1797a5&ti=Fatigue%20Assessment%20using%20Eyetracking%20for%20People%20with%20Special%20Needs
21. A. H. Khaleel, T. Abbas, and A.-W. S. Ibrahim, “Best low-cost methods for real-time detection of the eye and gaze tracking,” *i-com* , vol. 23, no. 1, p. 79, Jan. 2024, doi: 10.1515/icom-2023-0026.
22. E. Iacobelli, V. Ponzi, S. Russo, and C. Napoli, “Eye-Tracking System with Low-End Hardware: Development and Evaluation,” *Information* , vol. 14, no. 12, p. 644, Dec. 2023, doi: 10.3390/info14120644.
23. M. Y. Mustar, R. Hartanto, and P. I. Santosa, “Exploring Attentive User Interface Input via Raspberry Pi, Based on Face Landmark Detection, Eye Open-Closed Detection and Head Movements Detection,” *Ingénierie des systèmes d information* , vol. 29, no. 4, p. 1343, Aug. 2024, doi: 10.18280/isi.290410.
24. S. Balaji and P. Sujatha, “Hybrid Eye-Tracking System for Cursor Control: A Kalman Filter and Exponential Moving Average-Based Approach for Robust Face Tracking,” May 2025, doi: 10.21203/rs.3.rs-6541901/v1.
25. P. Drakopoulos, G. A. Koulieris, and K. Mania, “Eye Tracking Interaction on Unmodified Mobile VR Headsets Using the Selfie Camera,” *ACM Transactions on Applied Perception* , vol. 18, no. 3, p. 1, May 2021, doi: 10.1145/3456875.
26. A. Kummen *et al.* , “MotionInput v2.0 supporting DirectX: A modular library of open-source gesture-based machine learning and computer vision methods for interacting and controlling existing software with a webcam,” *arXiv (Cornell University)* , Feb. 2022, doi: 10.48550/arxiv.2108.04357.

27. M. Danioł, D. Hemmerling, J. Sikora, P. Jemioło, M. Wodziński, and M. Wójcik-Pędziwiatr, "Eye-tracking in Mixed Reality for Diagnosis of Neurodegenerative Diseases," *arXiv (Cornell University)* , Apr. 2024, doi: 10.48550/arxiv.2404.12984.
28. Y. Zhao and Y. Xue, "Research on the Teaching Reform of Art and Design Courses in Colleges and Universities Driven by Artificial Intelligence," *Applied Mathematics and Nonlinear Sciences* , vol. 9, no. 1, Jan. 2024, doi: 10.2478/amns-2024-0625.
29. G. Casiez, N. Roussel, and D. Vogel, "1 € filter," May 2012, doi: 10.1145/2207676.2208639.
30. A. Islam, N. Rahaman, and M. A. R. Ahad, "A Study on Tiredness Assessment by Using Eye Blink Detection," *Jurnal Kejuruteraan* , vol. 31, no. 2, p. 209, Oct. 2019, doi: 10.17576/jkukm-2019-31(2)-04.
31. Prof. V. Jumb, C. Nalka, H. Hussain, and R. J. MATHEWS, "Morse Code Detection Using Eye Blinks," *International Journal Of Trendy Research In Engineering And Technology* , vol. 5, no. 1, p. 33, Jan. 2021, doi: 10.54473/ijtret.2021.5105.
32. W.-H. Chuah, S.-C. Chong, and L.-Y. Chong, "The Assistance of Eye Blink Detection for Two- Factor Authentication," *Journal of Informatics and Web Engineering* , vol. 2, no. 2, p. 111, Sep. 2023, doi: 10.33093/jiwe.2023.2.2.8.
33. M. S. Slim and R. J. Hartsuiker, "Moving visual world experiments online? A web-based replication of Dijkgraaf, Hartsuiker, and Duyck (2017) using PCIbex and WebGazer.js," *Behavior Research Methods* , vol. 55, no. 7, p. 3786, Nov. 2022, doi: 10.3758/s13428-022-01989-z.