

Eye-Controlled Virtual Mouse with Independent Click, Text Selection, Drag-and-Drop and Scroll Operations for Physically Challenged Users

Janasri S¹, Kiruthika D², Nandhitha J³, Yashwanthini K A⁴

Sengunthar Engineering College (Autonomous), Tiruchengode – 637205, Tamil Nadu, India (Affiliated to Anna University, Chennai)

Abstract—Modern law enforcement agencies face the dual challenge of managing complex public safety threats while maintaining strict institutional accountability. This paper explores the integration of artificial intelligence (AI), automation, and advanced digital tools within modern policing frameworks. By analyzing current implementations—including predictive policing algorithms, automated facial recognition, automated license plate readers (ALPRs), and AI-driven evidence management systems—this study assesses how these technologies enhance operational efficiency, accelerate response times, and optimize resource allocation. Furthermore, the paper addresses critical systemic challenges, specifically targeting algorithmic bias, citizen privacy concerns, data security vulnerabilities, and the regulatory frameworks required to govern automated decision-making. Ultimately, this research provides a balanced framework for law enforcement executives and policymakers, demonstrating how agencies can leverage cutting-edge technical innovation while upholding the core principles of ethical, transparent, and community-oriented justice.

Keywords—Artificial Intelligence (AI), Automated Policing, Law Enforcement Technology, Public Safety, Predictive Policing, Ethical Automation.

I. INTRODUCTION

Computers are central to education, employment and social participation, yet the mouse and keyboard assume that the user can grasp, point and press with the hands. For amputees and people with motor impairments, this assumption is a barrier to digital independence. Commercial eye trackers such as the Tobii Dynavox family address the problem, but their cost places them out of reach for many users. A webcam-based alternative, built from open-source computer-vision tools, offers a far more affordable route to hands-free interaction.

In our earlier work (Phase-1) we built a webcam virtual mouse in which gaze, estimated from the iris landmark of MediaPipe Face Mesh, moved the cursor and a hybrid CNN-SVM classifier turned blink events into clicks. Although the classification pipeline was accurate, the interaction vocabulary was narrow. A single eyelid threshold served both click types, so a left wink could be misread as a right wink; there was no way to select text; and click-and-hold was never extended into a complete, confirmed drag-and-drop interaction. Real assistive use requires reading, correcting and organising content, not just clicking once. This paper reports the Phase-2

enhancement, which closes that gap through software changes only. The main contributions are:

- Independent left-click and right-click recognition based on which eye closes and for how long, with per-click cooldown.
- Eye-driven text selection using a long left-eye closure that holds and later releases the left mouse button while the gaze traces the text.
- Eye-driven drag-and-drop using a long right-eye closure with explicit start and drop confirmation.
- Hands-free vertical scrolling using a double blink followed by head-pitch movement.
- A state guard with mutual exclusion, safety release and an on-screen feedback overlay that keeps these gestures from interfering with each other.

The remainder of the paper is organised as follows. Section II surveys related work, Section III describes the system and gaze mapping, Section IV details the gesture algorithms, Section V lists the implementation parameters, Section VI discusses results, Section VII presents use cases, Section VIII covers limitations and future work, and Section IX concludes.

II. RELATED WORK

Webcam-based pointing has been explored through both hand and eye channels. Multi-modal systems combine hand-gesture and eye-gesture control in one OpenCV pipeline, layering gaze on top of hand tracking for users with limited hand mobility [1]. Low-cost eye-controlled cursors with homography-based calibration and adaptive graphical interfaces have been proposed as alternatives to commercial trackers, and their nine-point calibration idea informs our calibration roadmap [2]. Dual-modality virtual mice that map either hand or eye gestures to the same operations show that one gesture vocabulary can serve several actions [3], while open-source pupil-tracking systems confirm that MediaPipe and OpenCV are sufficient for real-time performance on modest hardware [4]. Gaze-only systems are, however, prone to drift and to misclassifying involuntary blinks [5], which motivates our duration-based blink filtering.

Selection techniques in gaze interaction have a longer history. EyePoint introduced a look-and-shoot scheme in which a rough fixation is magnified and then confirmed, greatly reducing pointing error [6]. EyeSwipe replaced fixed dwell with continuous gaze paths [7], a concept reflected in our gaze-traced text selection. Eye-S built complete commands from a small alphabet of directional gaze gestures and dwell events [8]. Analyses of dwell-time thresholds highlight the Midas-touch problem, where natural looking triggers unintended selections [9], and PinchCatcher pairs gaze with a second confirming gesture to disambiguate nearby targets [10]. Our system follows the same gaze-plus-confirmation principle, using eyelid closure rather than a pinch as the confirming action.

Three themes follow from this literature

- (a) calibration-based, webcam-only gaze tracking is a viable low-cost alternative to dedicated hardware;
- (b) timing parameters must be tunable per user to balance speed against accidental triggers;
- (c) a shared gesture channel serving several actions needs an explicit disambiguation layer.

The Phase-2 design addresses

- (c) through duration thresholds and a mutual-exclusion state guard, and (b) through configurable constants.

III. SYSTEM OVERVIEW AND METHODOLOGY

1. Phase-1 Baseline

The Phase-1 pipeline captured frames with OpenCV, extracted 468 face landmarks with MediaPipe Face Mesh, mapped the iris landmark to screen coordinates, smoothed the result, and issued PyAutoGUI mouse events. Blinks were detected from the eye aspect ratio (EAR) [15] and classified by a hybrid CNN-SVM into left, right, double and click-and-hold actions. Its limitations—one shared eyelid threshold, no text selection and an incomplete drag pipeline—define the Phase-2 requirements

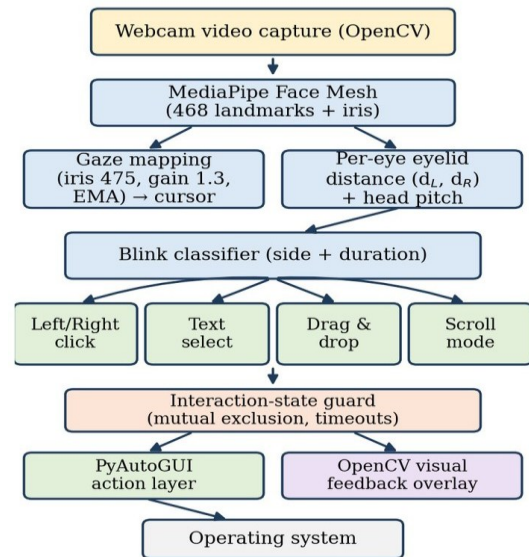


Fig. 1. Architecture of the enhanced eye-controlled virtual mouse.

3. Gaze-to-Cursor Mapping

The iris landmark (index 475) of the refined face mesh provides normalised coordinates (x_i, y_i) in $[0, 1]$. These are scaled to the screen of width W and height H , expanded about the screen centre by a gain $g = 1.3$ so that small eye movements reach the screen edges, and clamped:

$$s_x = \text{clamp}((W \cdot x_i - W/2) \cdot g + W/2, 0, W-1)$$

(1)

The same expression is used for the vertical axis. To suppress jitter, an exponential moving average with smoothing factor $\alpha = 0.25$ is applied before the cursor is moved:

where ct is the smoothed cursor position. The result is passed to PyAutoGUI `moveTo()` with a 10 ms

duration. The PyAutoGUI fail-safe is disabled so that corner-gazing cannot interrupt control; this is compensated by an Esc key exit and safe release logic described in Section IV.

4. Per-Eye Blink Measurement

Because left and right actions must be independent, each eye is measured separately. For the left eye the vertical distance between the upper-lid landmark 159 and the lower-lid landmark 145 is used, and for the right eye landmarks 386 and 374:

$$dL = |y_{159} - y_{145}|, \quad dR = |y_{386} - y_{374}|$$

(3)

An eye is considered closed when its distance falls below a threshold $\theta = 0.012$ (in normalised image units). Using one distance per eye, rather than a single shared value, removes the Phase-1 ambiguity in which one wink could be read as the other. The duration of each closure, measured from the first closed frame to the first open frame, then separates intentional winks from natural blinks and from long, deliberate closures.

Head pitch for scrolling is measured as the vertical offset between the nose-tip landmark (index 1) and the mean eyelid-centre height, smoothed with a factor of 0.25:

$$pt = y_{nose} - (y_{L,c} + y_{R,c}) / 2$$

(4)

5. Threshold Selection and Personalisation

The closure threshold θ and the timing constants were chosen empirically from the eyelid distances observed with the live preview overlay: the open-eye distance sits well above θ , a deliberate wink falls well below it, and natural blinks are short enough to fall under the 0.05 s minimum or to be excluded by the cooldown.

The click window (0.05–0.40 s) therefore captures deliberate winks, while the 0.65 s long-closure threshold is deliberately longer than any spontaneous blink, whose typical duration is a few hundred milliseconds [15]. Because eyelid geometry, webcam placement and blink habits differ between users, all constants are exposed as named parameters so that they can be re-tuned, and the planned calibration module will estimate each user's resting and closed eyelid distances to set θ automatically. The design places three requirements on the pipeline: (R1) every command must be reachable with a gesture that cannot be produced by natural blinking alone; (R2) no gesture may leave the operating system in an unrecoverable

state; and (R3) the user must always be able to see the active state. R1 is met by duration thresholds and single-eye closure, R2 by mutual exclusion and release-on-exit, and R3 by the feedback overlay.

IV. GESTURE DESIGN AND ALGORITHMS

Table I summarises the gesture vocabulary. Closure duration τ is measured for one eye while the other stays open; both-eye closures are used only for the scroll trigger.

Table I. Gesture-To-Action Mapping

Gesture	Condition	Action
Left-eye blink	$0.05 \text{ s} \leq \tau \leq 0.40 \text{ s}$	Left click
Right-eye blink	$0.05 \text{ s} \leq \tau \leq 0.40 \text{ s}$	Right click
Long left close	$\tau \geq 0.65 \text{ s}$	Start / end text selection
Long right close	$\tau \geq 0.65 \text{ s}$	Start / drop drag
Two both-eye blinks	Within 1.2 s	Enter scroll mode
Head up / down	$ \Delta p > 0.012$	Scroll up / down
Esc key	—	Exit program

• Independent Left and Right Click

When exactly one eye is closed, its closure start time is recorded. On re-opening, the closure duration τ is computed. Closures shorter than the minimum (0.05 s) are treated as noise, and those longer than 0.40 s but shorter than 0.65 s are ignored as ambiguous. A closure in the valid range issues a left click if it was the left eye and a right click if it was the right eye. A cooldown of 0.60 s after each click suppresses duplicate triggers. A click is further blocked if text selection, dragging or scroll mode is active, or if a selection has just ended (0.30 s guard), so that releasing a selection cannot produce a stray click.

• Text Selection

Selecting text with a mouse requires pressing the button, moving, and releasing. The system reproduces this with two long left-eye closures. The first closure with $\tau \geq 0.65 \text{ s}$ calls `mouseDown(left)` at the current gaze position and sets a selecting flag. While the flag is set, the cursor continues to follow the gaze, extending the highlighted span. A second long left-eye closure calls `mouseUp(left)`, ending the selection and recording the finish time for the 0.30 s guard. The toggle design

lets the user keep both eyes open and look freely while selecting, which avoids the fatigue of holding an eye closed for a long period.

Drag-and-Drop

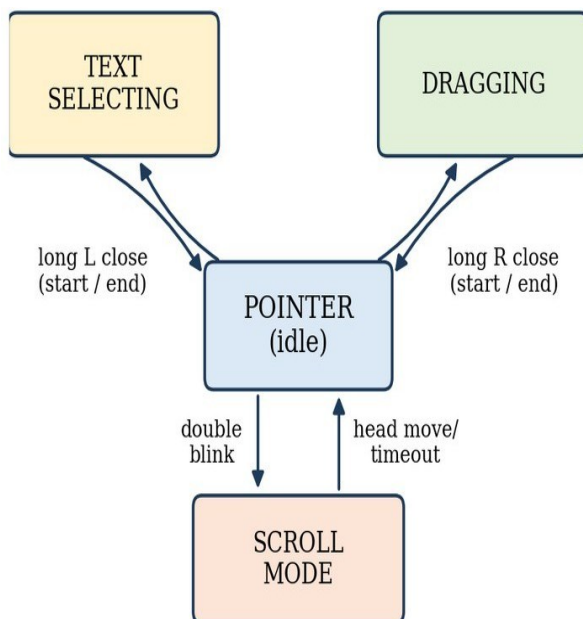
Drag-and-drop follows the same toggle principle on the right eye. The first long right-eye closure issues mouseDown(left) over the source object and sets a dragging flag; the object then follows the gaze. A second long right-eye closure issues mouseUp(left) at the destination, completing the drop. Using separate eyes for selection and drag keeps the two interactions mutually distinguishable and cognitively simple for the user. Both are blocked while the other is active or while in scroll mode.

• Head-Assisted Scrolling

Scrolling needs a continuous one-dimensional control that is awkward to produce with blinks alone. Two successive both-eye blinks, each of valid duration and separated by at most 1.2 s, enter scroll mode and record the current head pitch as a reference. If the pitch then differs from the reference by more than 0.012 upward, the page scrolls up by five units; downward, it scrolls down by five units. Scroll mode ends after one scroll action or after a 5 s timeout. Entering scroll mode first releases any active selection or drag, so the mouse button is never left pressed.

• State Guard and Safety

Selection, drag and scroll are three mutually exclusive states in addition to the idle pointer state (Fig. 2). Gestures that do not belong to the active state are ignored, which prevents cross-mode false triggers. Pressed buttons are always released when the program exits, so the operating system cannot be left in a held-button state.



• Visual Feedback

Because blink-based control has no tactile feedback, the OpenCV preview window overlays the live left and right eyelid distances, the double-blink counter, and a state label—"TEXT SELECTING...", "DRAGGING...", "SCROLL MODE", or "ACTION FINISHED"—so users can confirm what the system believes is active before they continue.

• Timing and Responsiveness

All processing is performed once per captured frame in a single loop, so the end-to-end delay of a gesture is bounded by the frame period plus the closure duration needed to disambiguate it. A click is issued at eye re-opening, so its latency is the wink duration itself, typically between 0.05 s and 0.40 s. The cursor path adds only the 10 ms moveTo duration and the smoothing lag of the EMA filter, which at $\alpha = 0.25$ settles within a few frames. The cost of this lag is the trade-off for jitter suppression: a larger α would track the gaze faster but amplify landmark noise.

Algorithm 1. Blink-event handling

- 1: compute dL, dR ; $L \leftarrow dL < 0$; $R \leftarrow dR < 0$
- 2: if L and R : track both-eye blink; two valid blinks in 1.2 s $\rightarrow$ enter scroll 3: else if exactly one eye closed: record start time
- 4: on re-open, $\tau \leftarrow \text{now} - \text{start}$
- 5: if $\tau \geq 0.65$: toggle selection (left) or drag (right)
- 6: else if $0.05 \leq \tau \leq 0.40$ and idle and cooldown elapsed: click (left/right)

V. IMPLEMENTATION

The system is written in Python and runs on Windows. OpenCV captures and displays frames, MediaPipe Face Mesh (one face, refined landmarks, detection and tracking confidence 0.5) extracts landmarks, NumPy supports coordinate processing, and PyAutoGUI executes click, mouseDown, mouseUp, scroll and moveTo operations. The minimum hardware is a laptop or desktop with an Intel Core i3 processor or equivalent, 4 GB RAM, 500 MB free storage and a built-in or external webcam. No additional hardware is needed, so the enhancement is purely a software upgrade of Phase-1. Table II lists the tunable constants used in the experiments.

Table II. Tunable Parameters

Parameter	Value	Purpose
Closure threshold θ	0.012	Eye closed
Min / max blink	0.05 / 0.40 s	Click window
Long closure	0.65 s	Select / drag toggle
Click cooldown	0.60 s	Duplicate guard
Double-blink window	1.2 s	Scroll trigger
Head-move threshold	0.012	Scroll direction
Scroll amount / timeout	5 / 5.0 s	Scroll step / exit
Cursor / head smoothing α	0.25 / 0.25	EMA filter
Gaze gain g	1.3	Edge reach

Table III. Accuracy Comparison Over 300 Trials

Method	Accuracy (%)	Gain (pp)
Hybrid CNN-SVM (proposed)	95.0	—
MTCNN	89.0	6.0
Dlib facial landmark	86.0	9.0
Haar cascades	83.0	12.0

3. Functional Comparison

Table IV compares the two phases. Phase-1 offered cursor movement and one undifferentiated click. Phase-2 adds separate left and right clicks, text selection, drag-and-drop and both scroll directions, covering the routine desktop operations that a physical mouse provides

VI. RESULTS AND DISCUSSION

1. Experimental Setup

The enhanced system was implemented and tested on a laptop with a standard webcam under normal indoor lighting. The operations tested were cursor movement, left click, right click, text selection, drag-and-drop, scroll up and scroll down. Screenshots of the selection, click and drag tests appear in the project report appendix; in the click test a right-eye blink opened the context menu of an editor window, in the selection test a highlighted code region was produced by gaze alone, and in the drag test a desktop file icon was relocated to a new position using only the long right-eye closures.

2. Blink-Classification Performance (Phase-1)

The underlying blink classifier was evaluated over 300 trials under varied lighting and head-pose conditions. The hybrid CNN-SVM achieved 95.0% accuracy, 94.0% precision, 93.0% recall and a 93.5% F1-score. Table III compares its accuracy with baseline approaches: it exceeds MTCNN by 6.0, Dlib facial landmarks by 9.0 and Haar cascades by 12.0 percentage points. Because Phase-2 reuses the same sensing layer, these figures characterise the reliability of the blink input on which the new gestures depend.

Table IV. Functional Comparison Of Phase-1 And Phase-2

Mouse function	Phase-1	Phase-2
Cursor movement	✓	✓
Basic click	✓	✓
Separate left click	—	✓
Separate right click	—	✓
Text selection	—	✓
Drag-and-drop	—	✓
Scroll up / down	—	✓

1. Discussion

Three design decisions explain most of the usability gains. First, measuring each eye separately and gating clicks by duration resolves the left/right ambiguity of Phase-1, and the 0.60 s cooldown prevents one wink from generating two clicks. Second, implementing selection and drag as toggles rather than hold-while-closed gestures avoids prolonged eyelid strain and lets the gaze move freely between press and release. Third, the state guard and the release-on-exit logic make the system fail safe: a stuck mouse button, the most disruptive error for a user who cannot easily reach a physical mouse to recover, cannot persist across mode changes or program termination.

The on-screen overlay was important in practice. Since the user cannot feel a click, seeing “TEXT SELECTING...” or “DRAGGING...” confirms that a long closure was recognised before the gaze is moved. Quantitative success rates for the new Phase-2 operations are a natural extension of the evaluation protocol used in Phase-1 and are left for a larger user study.

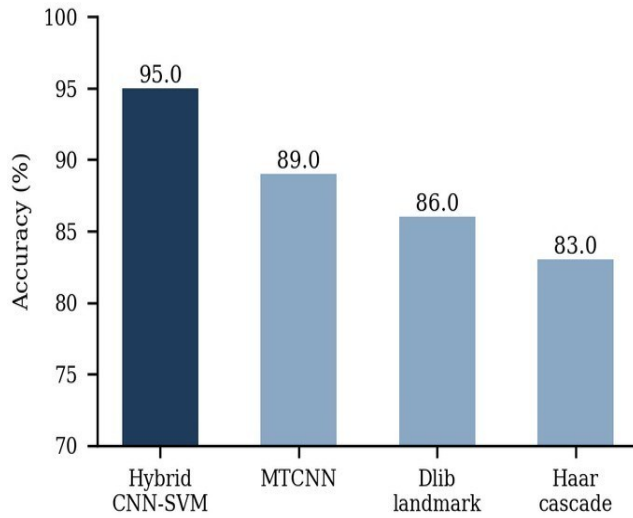


Fig. 3. Accuracy of the Phase-1 blink classifier versus baselines.

Test Scenarios

Each Phase-2 operation was exercised in a scripted scenario, summarised in Table V. For every scenario, the console log of the program printed the detected event (for example “LEFT CLICK”, “RIGHT CLICK”, “TEXT SELECTION START”, “DRAG STARTED”, “DROP COMPLETED”, “SCROLL UP/DOWN”), which was matched against the on-screen effect.

Table V. Test Scenarios And Observed Behaviour

Operation	Scenario	Observed behaviour
Left click	Left blink over a button	Button activated
Right click	Right blink in editor	Context menu opened
Text select	Long left close, trace, long left close	Code region highlighted
Drag-and-drop	Long right close on icon, look, long right close	Icon moved to gaze target
Scroll	Double blink, tilt head up/down	Page scrolled up/down

Comparison with Alternatives

Table VI places the enhanced system in context. Dedicated eye trackers offer higher gaze precision but need specialised, costly hardware. Phase-1 offered a low-cost webcam solution with limited commands. The enhanced system keeps the webcam-only cost while providing the complete set of routine pointing operations.

Table Vi. Qualitative Comparison Of Approaches

Feature	Dedicated tracker	Phase-1	Phase-2
Extra hardware	Required	None	None
Relative cost	High	Low	Low
Separate L/R click	Yes	No	Yes
Text selection	Yes	No	Yes
Drag-and-drop	Yes	No	Yes
Scrolling	Yes	No	Yes

1. Threats to Validity

The reported classification figures come from the Phase-1 evaluation and the Phase-2 operations were verified functionally under normal indoor lighting. The results therefore demonstrate feasibility rather than population-level performance. Accuracy may differ for users with ptosis, involuntary blinking or eyewear glare, and for cameras with different fields of view. The planned user study will address these factors with a larger and more diverse participant group.

VII. USE CASES

Internet browsing. An amputee can move the pointer by gaze, follow links with a left-eye blink, open context menus with a right-eye blink, select text to read or copy it, and scroll pages with a double blink and a head tilt. Desktop management. Files and folders can be opened, selected and dragged between locations without a mouse, which is the scenario demonstrated in the drag-and-drop test. Virtual meetings. Mute, camera and screen-sharing buttons can be activated by gaze and blink when the hands are occupied. Education and presentation. Teachers and presenters can point at, select and scroll digital whiteboard or slide content while remaining in front of the camera. The contactless operation also suits hygiene-sensitive settings such as hospitals and public kiosks.

VIII. LIMITATIONS AND FUTURE WORK

The system depends on adequate, stable lighting; shadows, glare or backlighting degrade landmark detection. Low-resolution or low-frame-rate webcams add latency and reduce cursor smoothness. Prolonged use, especially frequent intentional blinking, can cause eye strain, so the current design

is best suited to short and moderate sessions. The gesture vocabulary is deliberately small to avoid accidental commands, and fixed thresholds may not suit users with involuntary eye movements or difficulty producing deliberate blinks.

Future work will therefore include: (i) the nine-point per-user calibration and stored threshold profiles described in the Phase-2 design, with a Kalman filter for jitter reduction; (ii) dwell-based word and sentence selection using double-click and triple-click events, with user-adjustable dwell times to manage the Midas-touch problem [9]; (iii) customisable gesture mapping and adaptive timing; (iv) an eye-controlled virtual keyboard and voice-command integration; (v) improved robustness under varied illumination; and (vi) a structured user study reporting per-operation success rate, response time and fatigue measures with target users.

IX. CONCLUSION

This paper presented an enhanced eye-controlled virtual mouse for physically challenged users. Using only a webcam, MediaPipe Face Mesh, OpenCV and PyAutoGUI, the system extends a gaze-and-blink prototype into a multi-function pointing device with independent left and right clicks, text selection, drag-and-drop and head-assisted scrolling. Duration-based blink filtering, cooldown timers, a mutual-exclusion state guard and a visual feedback overlay keep the expanded gesture set reliable and safe. Together with the 95.0% accuracy of the underlying hybrid CNN-SVM blink classifier, the results indicate that low-cost, vision-based eye control can provide practical hands-free access to common computing tasks and promote digital independence for users with motor impairments.

Acknowledgment

The authors thank the Department of Artificial Intelligence and Data Science, Sengunthar Engineering College, and their project supervisor Ms. J. Priyadharshini, for guidance and support throughout this work.

REFERENCES

1. "Multi-modal human-computer interaction system with hand gesture and eye gesture control," IEEE, 2025.
2. "Eye-controlled mouse cursor and adaptive graphical user interface for disabled individuals," IEEE, 2025.
3. "Virtual mouse using eye gestures," IEEE, 2025.
4. "Smart AI based eye gesture control system," IEEE, 2025.
5. "Mouse cursor control with eye gestures," IEEE, 2024.
6. M. Kumar, A. Paepcke, and T. Winograd, "EyePoint: Practical pointing and selection using gaze and keyboard," in Proc. ACM CHI, 2007.
7. Kurauchi, W. Feng, A. Joshi, C. Morimoto, and M. Betke, "EyeSwipe: Dwell-free text entry using gaze paths," in Proc. ACM CHI, 2016.
8. Porta and M. Turina, "Eye-S: A full-screen input modality for pure eye-based communication," in Proc. ACM ETRA, 2008.
9. R. Schweigert, V. Schwind, and S. Mayer, "Eye pointing: A gaze-based selection technique," in Proc. Mensch und Computer, 2019.
10. "PinchCatcher: Enabling multi-selection for gaze + pinch," in Proc. ACM CHI, 2025.
11. S. J. Basha, L. S. L. Sowmya, S. Rumiya, and S. Afroz, "Virtual mouse with hand gestures using AI," in Proc. ICESC, Coimbatore, India, 2023, pp. 1432–1438.
12. K. Raja, C. Sugandhi, G. Nymish, N. S. Havish, and M. Rashmi, "Convolutional neural network based virtual mouse," in Proc. INCET, Belgaum, India, 2023, pp. 1–7.
13. G. Saini, R. Ottalwar, K. Upadhyay, and G. Kaur, "Gesture-driven interface redefinition: Enhancing accessibility with a virtual mouse system," in Proc. ICACCTech, Sonipat, India, 2024, pp. 132–136.
14. R. S. Kumar, A. V. Reddy, B. A. Kumar, B. P. K. Reddy, and A. S. K. Reddy, "Hand gesture-based virtual mouse with advanced controls using OpenCV and MediaPipe," in Proc. ICPECTS, Chennai, India, 2024, pp. 1–6.
15. T. Soukupová and J. Čech, "Real-time eye blink detection using facial landmarks," in Proc. 21st Computer Vision Winter Workshop, 2016.
16. M. Naidu, N. Muthukumaran, S. Chandralekha, K. T. Reddy, and K. S. Vaishnavi, "An analysis on virtual mouse control using human eye," in Proc. ICIPCN, Dhulikhel, Nepal, 2024, pp. 233–237.
17. N. Bhuvaneswary, M. Sathish, S. S. Abhayankar, K. S. V. Barath, and P.
18. N. Reddy, "Real time eye blink recognition for virtual mouse using Raspberry Pi," in Proc. IDCIoT, Bengaluru, India, 2025, pp. 972–975.
19. Lugaresi et al., "MediaPipe: A framework for building perception pipelines," arXiv:1906.08172, 2019.

