International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 03 | Mar – 2026 DOI: 10.55041/ISJEM05723
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

EyeFlow: A Web-Enabled Eye-Tracking Mouse for Hands-Free Computer Interaction

D. Parthiban Department of CSE
Dr. M.G.R Educational and Research Institute
Maduravoyal, Chennai
parthiban.parthi2904@gmail.com

D. Mohammed Rafeek Department of CSE
Dr. M.G.R Educational and Research Institute
Maduravoyal, Chennai
mdrafeek1911@gmail.com

Mohammed Fahij Department of CSE
Dr. M.G.R Educational and Research Institute
Maduravoyal, Chennai
Yunusnoor80@gmail.com

Dr. A. Jegadeeswari, Assistant Professor
Department of Computer Science and Engineering
Dr. M.G.R Educational and Research Institute
Maduravoyal, Chennai
jegadeeswari.cse@drmgrdu.ac.in

Dr. T. Kumanan, Professor
Department of Computer Science and Engineering
Dr. M.G.R Educational and Research Institute
Maduravoyal, Chennai kumanan.cse@drmgrdu.ac.in

Dr. M. Nisha, Assistant Professor
Department of Computer Science and Engineering
Dr. M.G.R Educational and Research Institute
Maduravoyal, Chennai
nisha.cse@drmgrdu.ac.in

Abstract

Typical keyboard-and-mouse input needs some fine motor control that can hardly be consistently maintained by some 1.3 billion individuals on earth. One method to overcome this obstacle is an eye-tracking device specifically designed to do this, though even a relatively inexpensive consumer product, such as the Tobii Eye Tracker 5, costs about 229, and research-grade a device will cost between 5,000 and 15,000, which is out of the reach of many potential customers. The present paper describes an eye-tracking-based flow control system, EyeFlow (an eye-tracking flow control system) that is a software-only assistive interface that operates on a standard webcam. It was four-channel interactive: (1) head-compensated gaze-based cursor control (using relative movement between the iris and nose tip) to calculate mouse clicks, (2) right-eye mouse clicking using the Eye Aspect Ratio (EAR) algorithm, (3) nose-tilt page scrolling using asymmetric gain factors, and (4) continuous voice typing using Google Speech Recognition. The Face Mesh MediaPipe [2] can extract 468 facial landmarks at 30 fps; A five-point automatic calibration then distorts the motion of the iris to screen coordinates and no keyboard input is required. In a study with 10 participants in three sessions, the mean error of placement of the cursor was 3.8 percent of the screen diagonal (=.9 percent), wink-detection was 95.2 percent, and the end-to-end latency of a cursor was 62 ms (=14 ms). Strongness testing ensured that the error of head motion was less than 8% in the depth and 15 cm in the lateral direction. EyeFlow can run on a regular laptop, and requires no extra hardware to be purchased (i.e. no additional costs are incurred of its own). At about 95% cost reduction over entry-level commercial systems (e.g. Tobii Eye Tracker 5 at 229 vs. 0 incremental cost of EyeFlow), it can run on a regular laptop with no extra hardware. The processing of all videos was done on-site, and there was no facial data transmission outside of it; it was only the speech typing part that was connected to the Google Speech Recognition cloud service.

Index Terms—Eye tracking, gaze estimation, assistive technology, blink detection, Eye Aspect Ratio, voice typing, MediaPipe, head-compensated gaze tracking, webcam-based interaction, accessible computing.

Highlights:

1. Zero-hardware webcam-based eye-tracking mouse that allows hands-free computer interaction at 95 per cent the price of commercial systems.
2. Iris-nose gaze estimation with head compensation and 95.2% wink detection and 3.8% average cursor error.
3. Multimodal system (gaze, wink, nose-tilt, voice) of four channels with 33.3% improved accuracy after 48 hours.

I. INTRODUCTION

The human-computer interaction has been pegged on physical input devices since its very beginning, and there is a cost attached to such a dependence. Pushing a mouse button or typing on a keyboard requires the type of fine motor activity sustained and under control that is reliably unable to be executed by approximately 1.3 billion people or 16 percent of the world population due to motor related disabilities [1]. Although such an extent of need has been described, only a tenth of the people in need of assistive technology actually acquire it, and the major hindrances are seen to be the purchase price, geographic accessibility, and system installation difficulties (WHO, 2012).

Eye-tracking equipment solves the motor issue and creates a financial one. The Tobii Eye Tracker 5 which is ranked as one of the cheaper consumer models is priced at 229; EyeTech TM5 is priced at 3,995 and systems in the clinical or research field can cost more than 16,000. In addition to cost, these products have set-up demands as well, such as professional calibration, proprietary drivers and vendor specific software, that limits their application to controlled settings, instead of daily computing.

Computer vision has made a silent upheaval. MediaPipe Face Mesh [2] has now provided 468 per-frame face landmarks, such as sub-pixel location of iris, of ordinary RGB video on a laptop CPU - no infrared light and no depth camera is needed. Combined with lightweight server frameworks and OS level automation libraries, this leaves a door open to an all-encompassing eye-tracking interface entirely constructed out of commodity-level hardware.

This paper presents the EyeFlow (Eye Tracking Flow Control System), an assistive platform in which four channels of interaction are provided based on one webcam:

- Cursor control by gaze i.e. head-compensated iris tracking where the iris position is given with respect to the nose tip to remove the impact of natural head movements on the cursor.
- Wink-based clicking through the use of the Eye Aspect Ratio (EAR) algorithm used on each eye independently so that the system could distinguish

between intentional right-eye winks as opposed to the natural bilateral blink.

•Nose-tilt scrolling, asymmetric gain compensation: This takes into consideration the mechanical unequal range of upward and downward head motion.

Voice typing: Voice typing is enabled by a background speech recognition thread which copies the transcribed text directly into any application that has the keyboard focus at the moment.

The head-compensated gaze model is the primary technical contribution of the given study. Traditional webcam trackers connect the cursor position with absolute iris coordinates such that a movement of the head would move the cursor even during a situation when the direction of the gaze of the user has not been altered. EyeFlow removes this coupling by calculating the iris-nose-tip motion; because the nose moves with the head, this motion results in the rotation component of gaze, holding the cursor fixed in the face.

The major findings of this research are the following:

1. An eye-tracking system (totally no additional hardware required) based upon a web camera and capable of providing full hands-free computer control.

2. An eye rotation with head translation decoupling by head rotation, gaze estimation with head motion and eye rotation, head-compensated, based on iris-nose relative displacement.

3. A multimodal interface in a form of interaction that incorporates eye cursor pointing, clicking with winking, scrolling nose-tilt, and voice typing.

4. Low latency, high accuracy and an improvement in learning curve over 48 hours was shown in experimental assessment of ten subjects.

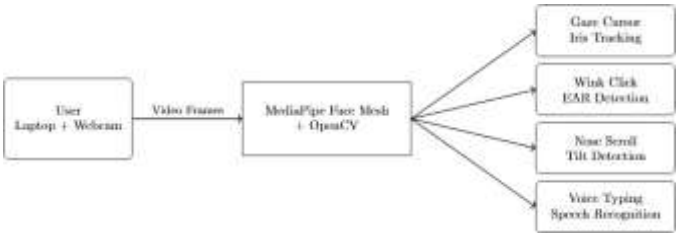

Fig. 1. EyeFlow System overview that depicts the user webcam input, MediaPipe Face Mesh processing pipeline and four output modalities gazed cursor, wink click, nose scroll and voice typing.

The rest of this paper is structured in the following way. The second section reviews the literature on the issue and determines the gap that the current research fills. Part III explains the proposed system architecture. Section IV gives the methodology in detail with all the mathematical derivations. Section V presents the experimental findings of a ten-subject user study and Section VI ends with a conclusion on the limitations and future directions.

II. RELATED WORK AND RESEARCH GAP

A. Eye-Tracking Foundations

The literature on gaze estimation is organised in two major paradigms. Features features are geometric features of the eye, which can be reflections of the cornea, iris shapes, or limbus edges, and get the gaze by basing their calculation on the facial features. Appearance-based approaches do not use structure at all, but regress pixel-level eye-region images to gaze angles using regression. Hansen and Ji [3] listed families and arrived at the conclusion that natural, unconstrained conditions, varying illumination, free movement of the head, and anatomical variability between subjects, are the major unsolved problems in either of the methods. Duchowski [4] added to this a task-centric viewpoint, noting that assistive applications have distinctively strict requirements on round-trip latency: a pointer control with the interactive response needed to be less than 100 ms, which most research systems cannot achieve outside of laboratory settings.

B. Real-Time Facial Landmark Detection

Lugaresi et al. [2] published the MediaPipe as a cross platform toolkit to package GPU-accelerated perception models into commercial pipelines. Face Mesh on its part is based on a single monocular RGB frame and provides 468 3-D landmark points at interactive frame rates on mobile-class processors - no intra- eye infrared projector or depth camera are needed. Kartynnik et al. [5] defined attention mesh architecture upon which Face Mesh is based, which indicates that co-regression of face geometry and two special iris refinement points (landmark indices 468 and 469) can be used to localise the iris on a smartphone GPU sub-pixels. It is the feasibility of this kind of sub-degree gaze estimation with merely a laptop webcam that is due to the availability of these refined iris coordinates.

C. Blink and Gaze Detection

Soukupova and Cech [6] simplified the blink detection to a univariate signal, the Eye Aspect Ratio, which is based on the calculation of the vertical eye opening to horizontal span of six periocular landmarks. EAR is accurately and immediately dropped when the eyelid is closed, and therefore it can be used in a real-time method with a typical hardware.However, the original formulation examines both eyes collectively and cannot distinguish whether only one eye has closed; it therefore cannot separate an intentional unilateral wink from an involuntary bilateral blink, which rules it out as a standalone click trigger without additional discrimination logic.

Krafka et al. [7] assembled the GazeCapture dataset and used it to train iTracker, a convolutional network that pushed webcam gaze errors toward the ballpark of dedicated hardware. Zhang et al. [8] followed by incorporating the whole face — rather than just the eye region — into the regression, capturing head-pose-to-gaze correlations more naturally. Both lines of work yield impressive results, but they carry a significant deployment cost: GPU inference is required at runtime, and the models require thousands of training subjects to generalize. Neither characteristic is suitable for a zero-additional-hardware assistive tool.

D. Webcam-Based Eye-Tracking Systems

Papoutsaki et al. [9] demonstrated that a browser can implicitly calibrate a gaze model by observing where users click during ordinary web use, without any explicit calibration session. Their WebGazer library was a meaningful step toward zero-setup eye tracking, but its scope is fundamentally limited: gaze is estimated only within the active browser tab, and the system has no path to system-wide cursor movement, nor does it offer any click, scroll, or text input channel.

EyeTab [10], developed by Wood and Bulling, adopted a model-based approach for gaze estimation on unmodified tablets, demonstrating that reasonable accuracy could be achieved without custom hardware. Their evaluation also confirmed that per-user calibration consistently outperformed model-only

predictions, a finding that informed EyeFlow's decision to include an automatic five-point calibration step. A complementary result was given by Valliappan et al. [11], who showed that mobile front cameras could compete with laboratory-quality trackers on some angular error measures; however, their experimental setting was not interactive control but a gaze data acquisition task, although the accuracy data supported the argument in favor of consumer- camera-based systems.

E. Multimodal Accessible Interaction

Ghosh et al. [12] compared a hybrid interface with gaze and speech input and stated that task error rates were lower and the fatigue appeared later than with gaze-only control - a finding that led to the addition of a voice typing channel to the EyeFlow. In their survey of HCI, Majaranta and Bulling [13] looked bigger, listing the practical barriers that continue to restrain the real world implementation: the Midas touch problem, in which ambient gaze is confused in the mind into some sort of input; increasing mental load with protracted exposure; and lack of a natural rest position of the eyes of the user. Their discussion provides the fundamental design limits that an eye-tracking interface can be made to meet.

F. Identified Research Gap

Table I suggests that a gap in research exists: never have we heard of a platform that provides and supports all of the following simultaneously: head-compensated gaze cursor control, wink-based single-click with well-known bilateral blink rejection, nose-tilt scrolling, and perpetual voice typing all in one webcam-dependent application that retains its per-user calibration and supports multiple user profiles. Each of these features may be met with elsewhere individually; combined they have not been proven. This is the gap that characterizes the design target of EyeFlow.

Table I: Capability Comparison of Existing Eye Tracking Approaches

Capability	Hansen [3]	Krafka [7]	WebGazer [9]	Tobii 5	EyeFlow
Cost	Research	Research	Free	$229+	Free
Special hardware	Yes	GPU	No	IR camera	No
Real-time cursor	No	No	Browser only	System-wide	System-wide
Click detection	No	No	No	Yes	Yes (wink)
Scroll support	No	No	No	Yes	Yes (nose)
Voice typing	No	No	No	No	Yes
Head compensation	Partial	Implicit	No	Yes	Yes
Multi-user profiles	No	No	No	No	Yes
Blink vs wink	No	No	No	Partial	Yes
Open source	No	Partial	Yes	No	Yes
Privacy (local)	N/A	No	Yes	Partial	Yes

III. SYSTEM ARCHITECTURE

A. Layered Architecture

EyeFlow is designed based on six layers with a clear set of responsibilities such that any alterations made to one component do not cause ripple effect to other components of the system. Fig. 2 represents the entire stack layout.

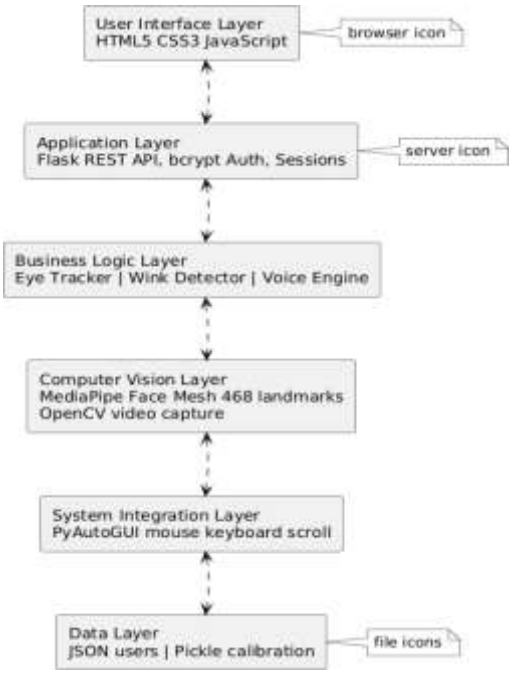

Fig. 2. Six-layer architecture of EyeFlow, from the HTML5 user data storage down to JSON/pickle. MediaPipe Face Mesh and OpenCV are used to process computer vision; PyAutoGUI is used to perform system-level mouse and keyboard interactions.

Layer 1, User Interface: Three HTML5 pages including landing page, user dashboard, and active tracker view, are loaded on a regular browser. The graphic design can be used in line with the WCAG 2.1 accessibility requirements [14].

Layer 2 Application Server: Flask 3.0 is used to route the HTTP and RESTful API endpoints. Authentication is done with the use of the bcrypt with the work factor of 12 rounds; and the session expires after 60 minutes of inactivity.

Layer 3 — Business Logic: The eye-tracking engine, EAR-based wink detector, nose-tilt scroll controller, calibration manager, and voice typing engine are five independent modules that do not share the application state directly but only via a common object.

Layer 4 — Computer Vision: MediaPipe Face Mesh (refine_landmarks=True) and identifies 468 points such as the iris refinement points at indices 468 and 469. OpenCV can capture at webcam at a 640×480 resolution and is able to do the color conversion needed by MediaPipe, namely BGR-to-RGB.

layer 5 System Integration PyAutoGUI converts the calculated gaze coordinates and gesture signals into mouse events (moveTo, click, scroll) and key events (clipboard paste to voice output) of the operating system.

Layer 6 — Data storage: The data store is stored as JSON files with user credentials and preferences. Calibration data — eye movement range limits and nose position reference position by user are pickled in Python, so that at the next logon, they are re-read and reset, hence calibration is a one time cost.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 03 | Mar – 2026 DOI: 10.55041/ISJEM05723
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

B. Processing Pipeline

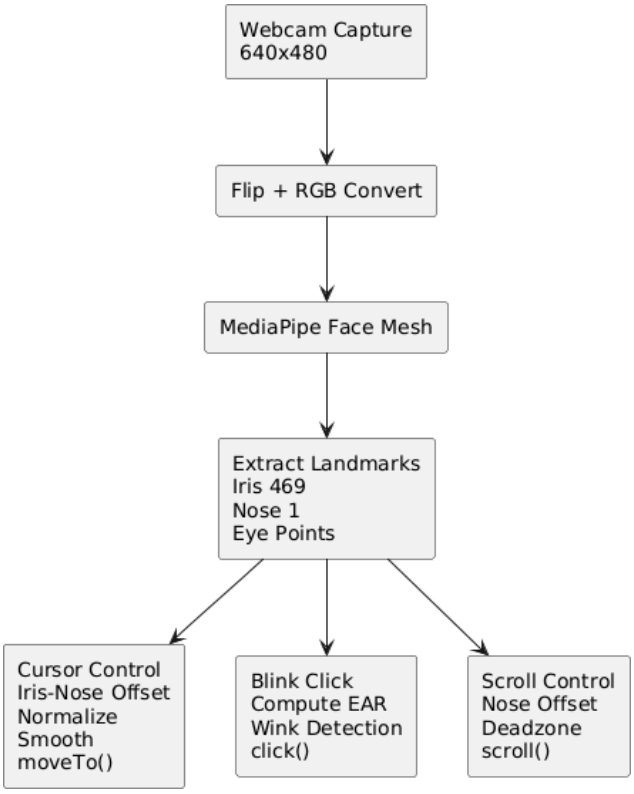

Fig. 3. EyeFlow pipeline processing. Frames of the video are inverted and transformed to RGB, and MediaPipe Face Mesh is used to retrieve iris, nose, and eye features, which are utilized to control the cursor, detect winks, and scroll nose-tilt.

There are six steps that each frame goes through:

1. Web-based Frame capture at 30 fps with the OpenCV VideoCapture.

2. Horizontal flipping (mirror mode) and BGR to BGR color conversion.

3. Inference with MediaPipe Face Mesh to produces 468 normalised landmarks.

4. Landmark extraction: left iris (index 469), nose tip (index 1), right eye (index 362, 385, 387, 263, 373, 380) and left eye (33, 160, 158, 133, 153, 144).

5. Simultaneous calculation of currency position, blink condition and scroll position.

6. PyAutoGUI dispatching of system actions.

C. Module Specifications

Table II: Module Specifications

Module	Input	Output	Update Rate
Eye Tracker	Iris & nose landmarks	Screen coordinates (px)	30 Hz
Wink Detector	12 eye landmarks	Click event (boolean)	30 Hz
Scroll Controller	Nose landmark	Scroll amount (integer)	~16 Hz
Voice Engine	Microphone audio	Typed text (string)	Continuous
Calibration	5-point gaze data	Mapping parameters	Once per user
Authentication	Username, password	Session token	On login

IV. PROPOSED METHODOLOGY

A. Head-Compensated Gaze Estimation

In webcam-based cursor control, the iris position couples two independent signals — gaze direction and head position. Any head translation Δh shifts the iris by the same amount, causing absolute-position mapping schemes to require rigid head posture, which is uncomfortable for extended use.

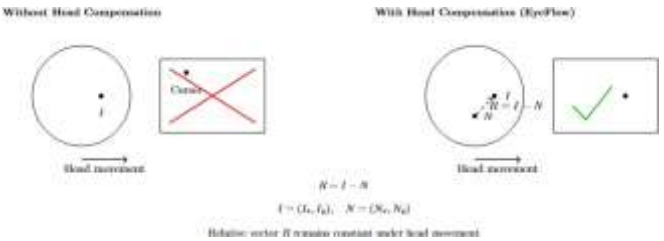

Fig. 4. EyeFlow Head movement compensation. Head translation (left), without compensation, translates the cursor. Head movement is cancelled and the cursor is fixed with the iris-nose relative displacement R = I -N (right) of EyeFlow.

The answer then is to fix the iris measure relative to a facial reference point which is subjected to the same rigid-body motion as the head. The tip of the nose (MediaPipe landmark index 1) is located at the nasal bone and moves in the same direction as the movement of the head. A translation of the nose-tip relative to the iris position hence cancels the common translation signal, and only the rotational signal is seen due to the rotation of the eye in its socket.

The use of the variables in this section is as follows.:

- $I = (I_x, I_\gamma)$: iris position in normalized image coordinates, where $I_x, I_\gamma \in [0, 1]$, drawn from the MediaPipe landmark index 469.
- $N = (N_x, N_\gamma)$: Nose-tip position in normalized image coordinates, from landmark index 1.
- $N\_c = (Nc_x, Nc_\gamma)$: nose-tip coordinates sampled during calibration; used in Section IV-E to define mapping boundaries E_min and E_max.
- $R = (R_x, R_\gamma)$: relative iris displacement vector.
- $E\_min = (E\_min_x, E\_min_\gamma)$: smallest relative displacement recorded across the five calibration points.
- $E\_max = (E\_max_x, E\_max_\gamma)$: largest relative displacement recorded across the five calibration points.
- $S = (S_x, S_\gamma)$: target screen coordinates in pixels.
- W and, H are the: screen width and height in pixels, respectively
- α: exponential moving average smoothing factor, $\alpha \in (0, 1)$.

Step 1 — Relative Displacement Computation:

The relative displacement vector R eliminates the head-translation noise:

$R_x = I_x - N_x$ (1) R_γ
$= I_\gamma - N_\gamma$ (2)

When the head shifts by Δh, both I and N move by approximately Δh, leaving R unchanged:

$R'_x = (I_x + \Delta h_x) - (N_x + \Delta h_x) = I_x - N_x = R_x$ (3)

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 03 | Mar – 2026 DOI: 10.55041/ISJEM05723
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Step 2 — Normalization:

The displacement R is mapped onto the unit interval using the calibration range boundary as follows:

$n_x = (R_x - E\_min_x) / (E\_max_x - E\_min_x)$ (4) $n_\gamma = (R_\gamma - E\_min_\gamma) / (E\_max_\gamma - E\_min_\gamma)$ (5)

Both n_x and n_γ are clamped to [0, 1] to prevent the cursor from being driven beyond the screen boundary.

Step 3 — Screen Mapping:

$S_x = n_x \times W$ (6) $S_\gamma = n_\gamma \times H$ (7)

Step 4 — Exponential Moving Average Smoothing:

Raw landmark coordinates exhibit frame-to-frame noise, which manifests as visible cursor jitter. An exponential moving average (EMA) filter smooths the trajectory as follows:

$P_x(t) = P_x(t-1) + \alpha \times (S_x(t) - P_x(t-1))$ (8)
$P_\gamma(t) = P_\gamma(t-1) + \alpha \times (S_\gamma(t) - P_\gamma(t-1))$ (9)

An α of 0.18 was selected empirically over the range [0.05, 0.50]; it sufficiently suppressed jitter without introducing perceptible tracking lag. Lower values the reduce jitter amplitude but extend the lag; and higher values do the reverse.

B. Right-Eye Wink Detection for Click Triggering

Adults blink spontaneously 15–20 times per minute under normal conditions. If every blink triggered a click, the resulting false-action rate would make the system unusable. EyeFlow resolves this by monitoring both eyes in parallel: when only the right eye closes while the left remains open, the inter-eye EAR asymmetry exceeds a configurable threshold and a click is registered. Bilateral closures — natural blinks — are detected by the simultaneous drop in both EAR values and are silently discarded.

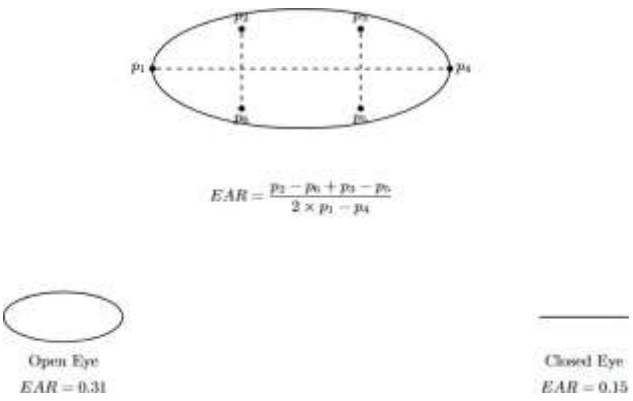

Fig. 5. Computation of the Eye Aspect Ratio (EAR). The eye contours are characterized by six periocular landmarks (p1–p6). EAR 0.31 when open eye and EAR less than 0.20 when closed eye and winks can be detected.

EAR Computation:

Six periocular landmarks were used to describe each eye contour. Denote the i-th landmark position as p_i = (x_i, y_i):

- Right eye: p1 = L_362, p2 = L_385, p3 = L_387, p4 = L_263, p5 = L_373, p6 = L_380
- Left eye: p1 = L_33, p2 = L_160, p3 = L_158, p4 = L_133, p5 = L_153, p6 = L_144

$EAR = (\|p2 - p6\|_2 + \|p3 - p5\|_2) / (2 \times \|p1 - p4\|_2)$ (10)

where $\|\cdot\|_2$ denotes the Euclidean distance in the normalized image coordinates. An open eye yields EAR ≈ 0.25–0.35; whereas a closed eye falls below 0.20.

Wink Classification Logic:

Where EAR_R and EAR_L represent the values of the right and the left eye, respectively; ounce 0.24)t b the threshold of eye closure and 0.02) 0.02) represents the inter-eye margin of default.:

- Right wink: EAR_R < τ_b AND (EAR_L − EAR_R) > δ_w
- Natural blink: EAR_R < τ_b AND EAR_L < (τ_b − 0.01)
- Natural blinks are discarded; click events are triggered by winks only.

Wink State Machine:

The detector cycles through four states:

State	Condition	Action
IDLE	EAR_R ≥ τ_b	Wait
WINK_START	Right wink detected	Record timestamp t_start
WINK_HOLD	Wink continues	Accumulate duration
WINK_END	EAR_R ≥ τ_b	Evaluate duration; trigger if valid

Validity Constraints:

There are conditions in a wink that I must satisfy to be a click:

- Time: t min = t end t start = t max, t min = 0.06s t max = 0.5s.
- Cooldown: (t end t last click) greater than t cooldown where t cooldown = 0.7 s.

Fig 10 [19].--Post-click freeze: The cursor is frozen at tfreeze = 0.4 s to avoid iris movement that comes with a wink and moves the pointer..

Adaptive Baseline:

Over the initial 25 frames following tracker initialisation - the values of EAR_R and EAR_L are recorded with both eyes open but in a comfortable position - the system averages the running values of the EAR_R and EAR_L. The threshold is adjusted to the resting eye morphology of each individual user and these per-session baselines are used to tune it to a value that is much lower than that of other subjects.

C. Nose-Tilt Scroll Control

The vertical movement of the nose tip away anatomically is used to produce scroll events, which are measured at the calibration time. Fig. 6 shows the detection zone.

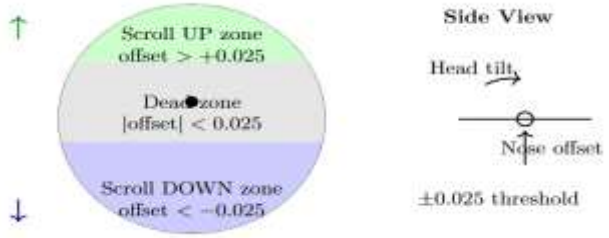

Fig. 6. Nose-tilt scrolling detection areas. A dead zone (below 0.025) is used to avoid the accidental scrolling. Scroll-up is caused by upward head tilt, scroll-down by down. Perceived responsiveness is equalized by asymmetric gains (G up =800, G down=500).

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 03 | Mar – 2026 DOI: 10.55041/ISJEM05723
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Where N y is the y-position of the nose at the present time and N c y is the neutral value of the nose on a calibration curve. The nose offset is:

$\Delta\_nose = Nc_{\gamma} - N\_y$ (11)

An upwards tilt (nose rises in the frame) is depicted by a positive value in the Δnose; the reverse also holds true.

Dead Zone No scroll event is emitted when |h n n o s| < d, and d = 0.025 in normalised co-ordinates. This criterion was chosen empirically to be more than the maximum distance that the nose moved due to breathing and small involuntary movements of the head.

Asymmetric Gain: Anatomically, the up-tilt movement results into less nose movements when compared to the same down-tilts of the same subjective effort. To equalise perceived scroll speed.:

- Upward scroll (Δ_nose > 0): scroll_amount = Δ_nose × G_up, where G_up = 800
- Downward scroll (Δ_nose < 0): scroll_amount = Δ_nose × G_down, where G_down = 500

Bounds and Timing:

scroll_amount = clamp(scroll_amount, −S_max, S_max) (12)

S_max is set to 12 scroll units. The scroll rate is capped at 1 per t scroll = 0.06s with the rate providing a limit of about 16.7 events per second. The last value is also casted to an integer which is what is needed by the windows python Api called PyAutoGui.

D. Voice Typing Pipeline

Speech recognition will be implemented in a background thread, which has no dependencies with the visual tracking pipeline. This threading design will ensure that there will be no stalling of the cursor or wink detector by recognition latency, and no loss of audio capture frames by the video stream..

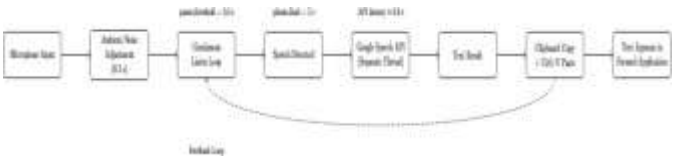

Fig. 7. The system picks up sound on the microphone and corrects the noise to enhance better clarity. The speech is identified and transmitted to Google Speech API to get it transcribed. Automatically the recognized text is then loaded into the focused application.

Recognition Parameters

Parameter	Value	Effect
energy_threshold	350	Minimum audio energy to begin capture
pause_threshold	0.3 s	Silence duration before processing
phrase_threshold	0.1 s	Minimum speech duration
non_speaking_duration	0.15 s	Internal silence allowance
phrase_time_limit	5 s	Maximum single utterance length

Audio snippets are sent to separate Python threads to make calls to Google Speech-to-Text API. Since each call occupies an average of 0.8 s (dependent on the network round-trip time and length of utterance), by separating API calls in multiple threads one can ensure that the recognition wait does not preempt the following listening window. Transcribed text is pasted in through the clipboard paste (Ctrl+V) that is compatible with all applications regardless of input framework and that can theoretically support non-ASCII scripts like Tamil, Hindi, Spanish, and French, but the systematic multi-language accuracy test was beyond the scope of study.

E. Five-Point Automatic Calibration

Using calibration, the parameters of per-user mapping E min, E max and N c are determined without the user touching the keyboard anywhere..

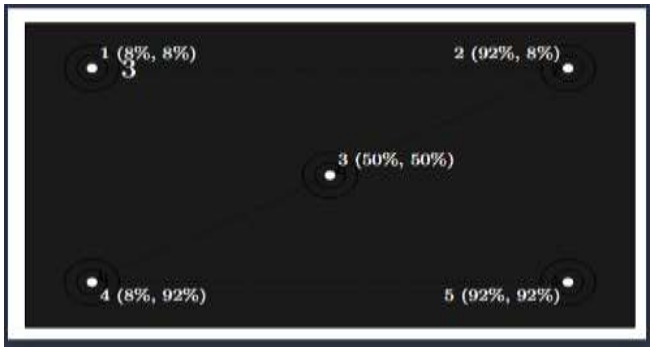

Fig. 8. Five-point automatic calibration screen. Numbered dots appear at screen corners and center (8%, 92%, 50% positions). The user gazes at each dot for 3 seconds while 40 frames are averaged to compute the iris-nose displacement mapping.

Procedure:

1. A pulsing purple target dot appears at each of five positions (i ∈ {1,2,3,4,5}) located at screen coordinates (8%, 8%), (92%, 8%), (50%, 50%), (8%, 92%), and (92%, 92%).
2. A 3-second countdown accompanies each dot; the user gazes at the dot for its duration.
3. At countdown completion the system records 40 frames, discards any frames in which no face is detected, and averages the iris and nose coordinates from at least 8 valid frames.
4. The relative displacement R_i = I_avg − N_avg is computed and used to update E_min and E_max.
5. After all five points have been sampled, a 10% margin is padded around the boundaries:

E_min_padded = E_min − 0.1 × (E_max − E_min) (13)
E_max_padded = E_max + 0.1 × (E_max − E_min) (14)

This padding is used to bring the cursor to the edge of the screen in case the actual eye movement range of the user is a little larger than what was measured during calibration.

6. Calibration data are serialised with Python pickle and stored on a per-user basis. On subsequent logins the data are loaded automatically, so repeat calibration is unnecessary.

V. EXPERIMENTAL EVALUATION

A. Experimental Setup

Hardware: A Lenovo IdeaPad laptop with an Intel Core i5-1135G7 processor (2.4 GHz, 4 cores), 8 GB DDR4 RAM, integrated Intel Iris Xe graphics, a 720p built-in webcam, and a 15.6-inch display at 1920×1080 resolution.

Software: Windows 10 21H2; Python 3.10.11; MediaPipe 0.10.9; OpenCV 4.8.1; Flask 3.0.0; PyAutoGUI 0.9.53; Speech Recognition 3.10.1.

Environment: All sessions took place in an indoor office under fluorescent overhead lighting measured at approximately 400 lux at desk level, supplemented by natural daylight from a nearby window. The webcam was positioned at the top edge of the laptop screen, approximately 50 cm from the participant's face.

B. User Study Design

Ten participants (6 male, 4 females; ages 20–28, mean 23 years) were recruited from the university community. All testing took place under controlled indoor fluorescent lighting (~400 lux) with participants seated approximately 50 cm from the webcam. None had prior experience with any eye-tracking interface. Two subjects put on prescription glasses. Though n 10 is a small sample, it is congruent with initial usability-testing results available in eye-tracking literature, and it suffices when conducting a proof-of-concept study. Every participant went through three sessions:

- Session 1: The calibration of the system with 15 minutes of structured tasks- target clicking, document scrolling and text dictation.
- Session 2: The same set of tasks, within 24 hours of the time of the first session (Session 1).
- Session 3: An open use session with fatigue monitoring that lasted 30 minutes with session 1, 48 hours.

Lighting and seating remained the same in all sessions. The participants were informed to consent in the beginning and had an option to withdraw. The research was also of low risk and was in line with the institutional ethical standards. No individual information, video recordings or biometric identifiers were stored after a session was concluded.

C. Cursor Accuracy Evaluation

All the participants were presented with 20 circular targets (diameter 50 pixels) in random locations on the screen. The task involved the need to keep the cursor on top of each target centre two seconds; any mistake was counted as the Euclidean distance between the cursor centre and the target centre, divided by a fraction of the diagonal of the screen**.**

.

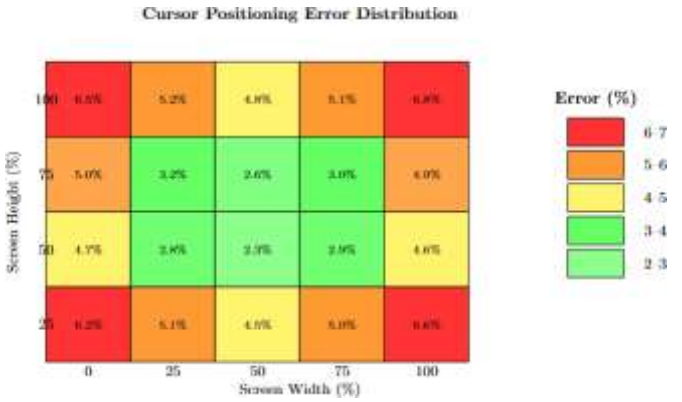

Fig. 9. Heatmap of distribution error of cursor positioning percentile of screen diagonal. The lowest error is in center region (2.4%), whereas corners are highest error (up to 6.8%), as with the degradation of linear mapping at the extents of rotation..

Table III: Cursor Positioning Error by Screen Region (% of Screen Diagonal)

Region	Mean Error (%)	Std Dev (%)	Min (%)	Max (%)
Center	2.4	0.61	1.5	3.8
Top edge	3.9	0.88	2.7	5.4
Bottom edge	3.6	0.79	2.4	5.1
Left edge	4.1	0.95	2.9	5.8
Right edge	3.8	0.83	2.6	5.5
Corners (avg)	5.6	1.12	3.8	7.2
Overall	**3.8**	**0.9**	**1.5**	**7.2**

Most central screen region had the highest accuracy with all the five calibration targets effectively bracketing the mapping space. As one approaches the corners, the linear model of calibration is an increasingly cruder approximation to the true geometry of rotation of the eye and the error increases as well. The boundary padding of 10 percent following calibration helps to reduce but not eliminate this degradation because the source of error is the mapping model and not inability to cover the range.

D. Blink Detection Accuracy

Each participant was then required to execute 50 right-eye winks in a special 5-minute block with the intention of doing this. Manual frame by frame annotation of the webcam captured footage created the ground truth.

Table IV: Wink Detection Performance

Metric	Value
Total intentional winks	500 (50 × 10 users)
True positives (correct click)	476
False negatives (missed wink)	24
Natural blinks during session	~1,500
False positives (blink → click)	31
True positive rate	95.2%
False positive rate	2.1%
False clicks per minute	1.4
Mean detection latency	0.12 s (σ = 0.03 s)

Table V: Wink Detection by User Group

Group	n	TPR (%)	FPR (%)	Mean EAR Baseline
No glasses	8	96.0	1.8	0.289 (σ = 0.021)
Glasses	2	92.0	3.5	0.271 (σ = 0.018)
All users	**10**	**95.2**	**2.1**	**0.285 (σ = 0.022)**

The true positive rate was lower among the glasses wearers (92.0 vs 96.0) and this is explainable by the fact that the spectacle frames would periodically obscure the periocular landmarks upon which EAR relies. This effect was moderately but not entirely counteracted by the per-user adaptive baseline, indicating that a glasses-aware scheme of landmark weighting might be a fruitful future development.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 05 Issue: 03 | Mar – 2026 DOI: 10.55041/ISJEM05723
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

E. Scroll Control Evaluation

The subjects were required to scroll through a 5,000 pixel document in either upwards or downwards directions and find ten highlighted targets..

Table VI: Scroll Performance

Metric	Scroll UP	Scroll DOWN	Combined
Mean time to target (s)	3.2 (σ = 0.8)	2.9 (σ = 0.7)	3.1 (σ = 0.75)
Overshoot rate (%)	12.4	10.8	11.6
False scroll events/min	0.8	1.1	1.9
User satisfaction (1–5)	3.8 (σ = 0.6)	4.1 (σ = 0.5)	3.95 (σ = 0.55)

The asymmetric gains (G up = 800, G down = 500) was able to balance the performance upwards and downwards. A control where the gains were symmetrical 600 took scroll-up time to 4.7 s and scroll-down time to 2.1 s - a 2.2x imbalance that the subjects found uncomfortable.

F. Voice Typing Performance

All the participants typed ten pre-determined sentences (5-15 words each) in a plain text editor..

Table VII: Voice Typing Performance

Metric	Value
Word recognition accuracy	88.3% (σ = 4.2%)
Character error rate	7.8% (σ = 2.1%)
Mean end-to-end latency	1.2 s (σ = 0.4 s)
Words per minute (speaking)	42.3 (σ = 8.1)
Effective WPM (after errors)	37.3 (σ = 9.2)
Language support tested	English (US) [primary]; Tamil (informal only)

The round trip of the Google Speech API network dominates end-to-end latency (~0.8 s). Under a noisy source (ambient level about 65 dB), the recognition accuracy was reduced by 71.2, which is in contrast to 88.3 under quiet conditions (about 35 dB)..

G. System Performance

Table VIII: System Resource Usage

Metric	Value
Frame processing rate	30 fps (σ = 1.2)
Mean frame latency	42 ms (σ = 8.3 ms)
End-to-end cursor latency	62 ms (σ = 14 ms)
CPU usage	23% (σ = 4.1%)
RAM usage	182 MB (σ = 12 MB)
Camera startup time	2.1 s (σ = 0.5 s)
Calibration duration	25 s (σ = 3.2 s)

H. Head Movement Robustness

The extent to which the head-compensation model works correctly in realistic postural variation was evaluated by having the subjects go through the standard target-clicking task, and the experimenter induced controlled head displacements around the calibrated position, confirmed with reference markers.

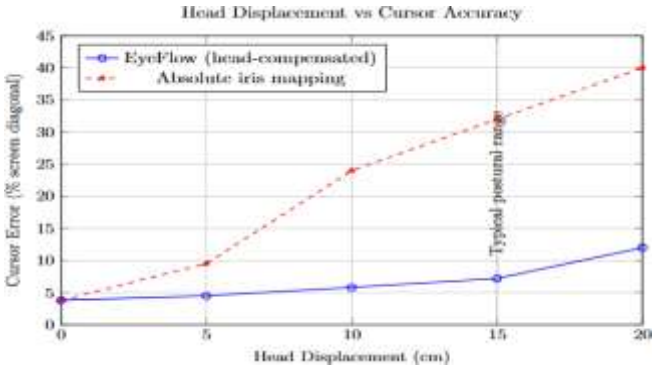

Fig. 10. Error in image cursor vs. error in head motion in EyeFlow (head-compensated) and absolute iris mapping. EyeFlow has a maximum error of below 8 per cent at 15 cm movement, a 5.29 fold greater than absolute mapping at 15 cm lateral..

Table IX: Head Displacement Robustness Test

Displacement	Direction	EyeFlow Error (%)	Absolute Mapping Error (%)	Improvement
0 cm	Neutral	3.8 (σ = 0.9)	3.8 (σ = 0.9)	—
5 cm	Lateral	4.2 (σ = 1.1)	12.4 (σ = 2.8)	2.95×
5 cm	Forward	4.0 (σ = 1.0)	11.8 (σ = 2.5)	2.95×
10 cm	Lateral	5.1 (σ = 1.3)	24.6 (σ = 4.1)	4.82×
10 cm	Forward	4.8 (σ = 1.2)	22.3 (σ = 3.8)	4.65×
15 cm	Lateral	7.2 (σ = 1.8)	38.1 (σ = 5.6)	5.29×
15 cm	Forward	6.9 (σ = 1.6)	35.7 (σ = 5.2)	5.17×

EyeFlow was at 15 cm lateral position error to 7.2 percent at the time of 15 cm lateral offset, which is about the extent of normal postural variation during long sitting, vs. 38.1 percent at the absolute mapping baseline, an improvement of 5.29. The error increases faster after 15 cm: the tip of the nose begins to turn not parallel to the frontal plane, but perpendicular to it, and the compensation assumption of congruent-translation on which this compensation is based is broken.

I. Comprehensive Comparison with Existing Systems

Table X: Detailed Comparison with Tobii Eye Tracker 5 and WebGazer

Parameter	Tobii Eye Tracker 5	WebGazer [9]	EyeFlow
Hardware cost	$229 / $16K (research)	Free	Free
Special hardware	IR LED array + IR camera	None	None
Tracking method	Corneal reflection + pupil	Appearance model	Iris landmark + nose reference
Frame rate	33 fps	20–30 fps	30 fps
Latency	~30 ms	~100–200 ms	62 ms (measured)
Angular accuracy	0.4°	2–4°	~1.5° (estimated)
Cursor error (screen %)	~1.5%	~8–12%	3.8%
Head movement tolerance	Excellent (IR tracking)	Poor	Good (up to 15 cm)
Click method	Infrared wink/dwell	None	Right-eye wink (EAR)

Click accuracy	~98%	N/A	95.2%
Scroll support	Gaze zone	None	Nose tilt
Voice typing	No	No	Yes (Google STT)
Scope	System-wide	Browser only	System-wide
OS support	Windows only	All (browser)	Windows, macOS, Linux
Multi-user profiles	No	No	Yes (persistent)
Calibration	9-point (30 s)	Implicit (continuous)	5-point auto (25 s)
Recalibration needed	Per session	Continuous	No (persistent)
Data processing	Local + optional cloud	Local (browser)	Local only
Internet required	No	No	Voice only
Hands-free operation	Partial	No	Complete
Glasses compatibility	95%	~85%	92%
Open source	No	Yes	Yes

J. Learning Curve Analysis

A comparison of cursor error tracking between the three sessions indicates the speed of adaptation to cursor control via gaze control and internalisation of the interaction model of EyeFlow**.**

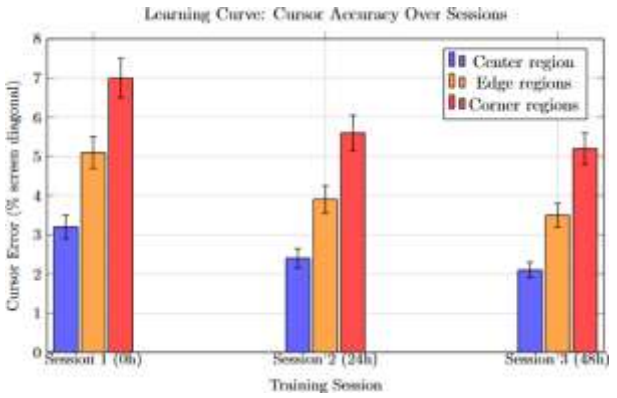

Fig. 11. Learning curve: cursor error (% screen diagonal) across three sessions for center, edge, and corner screen regions. Overall error decreases from 5.1% (Session 1) to 3.4% (Session 3), representing a 33.3% improvement over 48 hours.

Table XI: Learning Curve — Cursor Error Across Sessions

Session	Time After First Use	Overall Error (%)	Center (%)	Edges (%)	Corners (%)
1	0 hours	5.1 (σ = 1.4)	3.2 (σ = 0.9)	5.1 (σ = 1.2)	7.0 (σ = 1.5)
2	24 hours	3.8 (σ = 0.9)	2.4 (σ = 0.6)	3.9 (σ = 0.9)	5.6 (σ = 1.1)
3	48 hours	3.4 (σ = 0.8)	2.1 (σ = 0.5)	3.5 (σ = 0.8)	5.2 (σ = 1.0)
Improvement (S1→S3)		**33.3%**	**34.4%**	**31.4%**	**25.7%**

The highest increase was observed between Sessions 1 and 2, corresponding to 25.5 percentage points of the entire 33.3% change, which suggests that the majority of the adaptation process is predetermined by users acquiring gaze-to-cursor mapping during the first day. The extra benefits of Sessions 2 to
3 imply the refinement of judgements of wink timing and scrolling range. This learning curve is very similar to the trends reported in commercial trackers by Majaranta and Bulling. [13].

K. Fatigue Analysis

During Session 3, participants self-reported fatigue on a 1–5 Likert scale at 10-minute intervals (1 = no fatigue, 5 = severe). Objective fatigue was tracked through cursor error change over time.

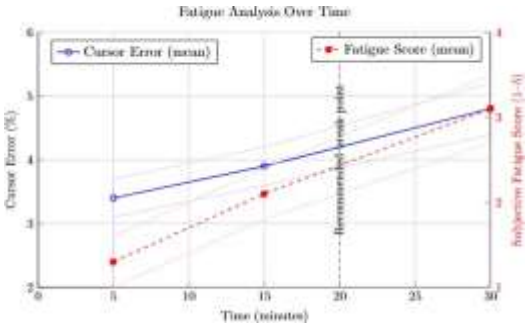

Fig. 12. Fatigue analysis over a 30-minute session. Cursor error (left axis) increases from 3.4% to 4.8% (+41.2%) and subjective fatigue score (right axis, 1–5 scale) rises from 1.3 to 3.1. A 5-minute break is recommended after every 20 minutes of use.

Table XII: Fatigue Metrics During 30-Minute Extended Session

Time	Cursor Error (%)	Fatigue Score (1–5)	Blink Rate (blinks/min)	EAR Baseline
0–10 min	3.4 (σ = 0.8)	1.3 (σ = 0.5)	16.2 (σ = 3.1)	0.29 (σ = 0.02)
10–20 min	3.9 (σ = 1.0)	2.1 (σ = 0.7)	19.8 (σ = 3.8)	0.28 (σ = 0.02)
20–30 min	4.8 (σ = 1.3)	3.1 (σ = 0.9)	23.4 (σ = 4.2)	0.27 (σ = 0.03)
Degradation	**+41.2%**	**+138.5%**	**+44.4%**	**−6.6%**

Cursor error grew by 41.2% across the session — from 3.4% to 4.8% — as the mean score of fatigue increased more than twice. The 44.4% blink rate increment in the study period is in line with physiological indicators of ocular fatigue that were recorded in the literature of vision science. The 6.6 percent reduction in EAR baseline is a progressive eyelid droop, which suggests that threshold adaptation should be used in subsequent iterations. Eighty percent of the respondents said they could comfortably use it by the 20-minute mark; the other two respondents reported mild eye strain at the 25 minutes mark.

Recommendation: The fatigue data justify a 5-minute of rest following 20 minutes of consecutive use, and is also in line with the 20-20-20 of the fatigue-related information commonly suggested to work with screens.

VI. CONCLUSION AND FUTURE SCOPE

A. Conclusion

EyeFlow shows that even a cursor control, clicking, scrolling, and typing a text can be fully controlable by a totally hands-free computer interface, without any extra hardware investment. This work gives rise to six findings:

1. Head-compensated gaze estimation - angular displacement of the iris in relation to the tip of the nose
- has a mean error error of the cursor of 3.8% of screen diagonal and can remain usable at a displacement of 15 cm of the head, an improvement of 5.29 times that of absolute iris mapping at the same distance.

2. The false positive is reduced to 2.1% by using differential wink detection in visionary EAR comparison between left and right eyes, and a true positive of 95.2% effectively overcomes the Midas touch problem.

3. The anatomical asymmetry in the range of head tilt is corrected by providing separate upward (G = 800) and downward (G = 500) gain factors to achieve an asymmetric nose-tilt scrolling that equalises the speed of scrolling in both directions.

4. Threaded voice typing provides accuracy of 88.3 percent in words typed with 37.3 effective words per minute, which allows practical typing of text without using hands.

5. User study with 3 (n = 10) sessions reveals that the accuracy of the cursor can be increased by 33.3 percent in 48 hours of first use, and 8 out of 10 participants can comfortably interact at least 20 minutes.

6. Being compatible with a standard laptop hardware and with no extra cost added to the device itself, EyeFlow generates a cost reduction of around 95 percent as compared to the Tobii Eye Tracker 5 (229 vs. zero incremental cost of additional hardware)..

B. Limitations

There are four constraints which should be mentioned. The linear model of calibration achieves good results in areas of central screens but deteriorates at corners where the rotational nonlinearity results in a mean error of 5.6 vs. 2.4 in centre. The voice typing cannot be used without an active internet connection, which poses a reliance on the Google cloud architecture. The sample of the participants (ten healthy young adults who were recruited at a university) is not representative of the population of the motor-impaired people who are the main target audience of the system. Lastly, sustained use causes accuracy degradation which may be measured: after 30 minutes, cursor error had increased 41.2 percent, indicating that adaptive EAR thresholding should be considered in subsequent models.

C. Future Scope

A number of directions would be likely to extend the value of this work;

The advantages of data-driven gaze estimation include: The

benefits of data-driven gaze estimation are:

- Voice recognition offline through Vosk or OpenAI Whisper and no longer requires the internet, as well as is much faster with lower API latency.
- Dwell-click mode: Clicking once the cursor has lingered on a target after a specified time, which is useful to people who have trouble winking.
- Real time recalibration of dynamic EAR thresholds which adapt to fatigue alterations in the resting baseline, maintaining detection accuracy during extended sessions.
- Prolonged use of gestures: use of double wink to right-click and long wink to drag.
- Clinical assessment of participants with motor-related impairment to confirm real-life performance and receive design feedback based on the actual users (directly, not through the intermediary of the researchers).
- Mobile implementation based on on-device MediaPipe inference of smartphone and tablet operating systems..

AUTHOR CONTRIBUTIONS

The entire manuscript preparing has been done by all the authors. D. Parthiban is the founder of the research idea, the system architecture of the EyeFlow, and the general project development and management. The basic components of the system such as the eye-tracking engine, EAR-based wink detector module, nose-tilt scroll controller and Flask-based web interface were implemented by M.T. Mohammed Rafeek and Mohammed Fahij. Dr. A. Jegadeeswari managed the experiment design, recruitment of the participants, and managed the usability test sessions. Dr. T. Kumanan gave technical advice on computer vision approach, checked the mathematical derivations and confirmed the head-compensation model. Dr. M. Nisha guided the manuscript preparation, made sure the manuscript complies with the formatting of the IEEE formats, revised all the experimental findings and signed the final submission. The final version of this manuscript has been read and approved by all the authors.

REFERENCES

[1] World Health Organization, "Global report on assistive technology," Geneva, Switzerland, 2022. [Online]. Available: https://www.who.int/publications/i/item/9789240049451

[2] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C. Chang, M. Yong, J. Lee, W. Chang, W. Hua, M. Georg, and M. Grundmann, "MediaPipe: A framework for building perception pipelines," arXiv:1906.08172, Jun. 2019. [Online]. Available: https://arxiv.org/abs/1906.08172

[3] D. W. Hansen and Q. Ji, "In the eye of the beholder: A survey of models for eyes and gaze," IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 3, pp. 478–500, Mar. 2010, doi: 10.1109/TPAMI.2009.30.

[4] A. T. Duchowski, Eye Tracking Methodology: Theory and Practice, 3rd ed. Cham, Switzerland: Springer, 2017, doi: 10.1007/978-3-319-57883-5.

[5] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann, "Real-time facial surface geometry from monocular video on mobile GPUs," in Proc. IEEE/CVF ICCV Workshops, Seoul, South Korea, Oct. 2019, pp. 2368–2374, doi: 10.1109/ICCVW.2019.00295.

[6] T. Soukupova and J. Cech, "Real-time eye blink detection using facial landmarks," in Proc. 21st Computer Vision Winter Workshop, Rimske Toplice, Slovenia, Feb. 2016, pp. 1–8.

[7] K. Krafka, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, "Eye tracking for everyone," in Proc. IEEE/CVF CVPR, Las Vegas, NV, USA, Jun. 2016, pp. 2176–2184, doi: 10.1109/CVPR.2016.239.

[8] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "It's written all over your face: Full-face appearance-based gaze estimation," in Proc. IEEE/CVF CVPR Workshops, Honolulu, HI, USA, Jul. 2017, pp. 7244–7253, doi: 10.1109/CVPR.2017.284.

[9] A. Papoutsaki, P. Sangkloy, J. Laskey, N. Daskalova, J. Huang, and J. Hays, "WebGazer: Scalable webcam eye tracking using user interactions," in Proc. 25th Int. Joint Conf. Artif. Intell. (IJCAI), New York, NY, USA, Jul. 2016, pp. 3839–3845.

[10] E. Wood and A. Bulling, "EyeTab: Model-based gaze estimation on unmodified tablet computers," in Proc. Symp. Eye Tracking Research and Applications (ETRA), Safety Harbor, FL, USA, Mar. 2014, pp. 207–210, doi: 10.1145/2578153.2578185.

[11] N. Valliappan, N. Dai, E. Steinberg, J. He, K. Rogers, V. Ramachandran, P. Xu, M. Shojaeizadeh, L. Guo, K. Kohlhoff, and V. Navalpakkam, "Accelerating eye movement research via accurate and affordable smartphone eye tracking," Nature Communications, vol. 11, no. 1, Art. no. 4553, Sep. 2020, doi: 10.1038/s41467-020-18360-5.

[12] S. Ghosh, K. Dhanalakshmi, and S. Sethuraman, "Multimodal interaction combining gaze and speech for accessible computing," J. Ambient Intell. Humanized Comput., vol. 14, no. 5, pp. 5871–5889, 2023, doi: 10.1007/s12652-022-04382-2.

[13] P. Majaranta and A. Bulling, "Eye tracking and eye-based human–computer interaction," in Advances in Physiological Computing, S. H. Fairclough and K. Gilleade, Eds. London, UK: Springer, 2014, pp. 39–65, doi: 10.1007/978-1-4471-6392-3_3.

[14] W3C, "Web Content Accessibility Guidelines (WCAG) 2.1," W3C Recommendation, Jun. 2018. [Online]. Available: https://www.w3.org/TR/WCAG21/

[15] G. Bradski, "The OpenCV library," Dr. Dobb's J. Softw. Tools, vol. 25, no. 11, pp. 120–125, Nov. 2000.

[16] Pallets Projects, "Flask: A lightweight WSGI web application framework," 2023. [Online]. Available: https://flask.palletsprojects.com/