Multidisciplinary | Peer-Reviewed
Open Access | Research Journal
ISSN: 3139-2954

Article ID: IRJIST-2026-101
Volume 1 | Issue 4 | 2026
DOI: https://doi.org/10.67308/irjist.101

International Research Journal of Innovation in Science and Technology

Real-Time User Attention and Typing Behavior Analysis via MediaPipe Face Mesh and Spatial Threshold Calibration

Supriya C. Mahatme[1], Padmali D. Wagh[2], Arnav Y. Wakekar[3], Kartik R. Wanjari[4], Anis J. Yelurkar[5], Rakshit S. Zalte[6]

[1-6]Department of Engineering, Sciences and Humanities (DESH), Vishwakarma Institute of Technology, Pune, Maharashtra, India
Correspondence: [3]arnav.12510401999@vit.edu

Abstract

Automated user attention mapping frameworks have attained an important role in measuring human-computer interaction (HCI) metrics and online educational methods. Conventional vision-based solutions usually result in substantial computational overhead from convolutional models or do not feature precise geometric thresholds, which cannot be adjusted for different seating positions. This paper suggests an efficient and non-intrusive software system able to track the user's screen-versus-keyboard attention. More precisely, it is capable of distinguishing when a person is looking at the computer screen and when the person is gazing at a keyboard. The solution uses a normal webcam video stream to receive data through the MediaPipe Face Mesh technology that enables automatic 3D coordinate system construction for getting a vertical head-pose offset. The two-level automatic system calibration process is carried out by means of a special model allowing data processing through asynchronous Flask web server work. In addition, to guarantee that no frame loss occurs at the hardware level, thread safety is ensured via a special mutex lock.

Keywords: Face Mesh; MediaPipe; Gaze Detection; Attention Tracking; Flask; Thread Safety; Typing Analytics; HCI.

1. Introduction

The rapid growth of virtual work environments, digital learning systems, and online monitoring technologies has increased the need for effective and privacy-conscious attendance monitoring systems. Educational organizations and training programs increasingly seek evidence-based methods to assess learner engagement. This need arises partly because instruction often takes place in uncontrolled home environments, where face-to-face monitoring may be impractical or costly.

Traditional eye-tracking systems based on infrared technology can provide sub-millimetre gaze-tracking precision but have significant financial and ergonomic limitations. Such systems typically require specialized infrared cameras, head stabilizers, and calibration procedures. These requirements make them difficult to incorporate into standard laptop-based environments and limit their accessibility for widespread use among different categories of students.

Webcam-based monitoring software offers a more accessible alternative; however, it also presents technical challenges. Static geometric parameters may produce inaccurate attention classifications when users change their seating positions or head orientations. Consequently, systems that rely exclusively on fixed thresholds may fail to adapt effectively to individual differences in posture and viewing behaviour. This creates a need for lightweight monitoring approaches that incorporate user-specific calibration while maintaining computational efficiency and usability.

To address these challenges, this paper presents a unified and lightweight software framework for identifying screen-versus-keyboard attention during typing activities in real time. The primary objective of the proposed prototype is to

determine whether a user is looking at the computer screen or directing attention towards the physical keyboard, a behaviour that may be associated with inexperienced typing patterns or distraction. The proposed system combines computer-vision techniques and typing-behaviour measurements through a multi-layered asynchronous-processing architecture. During the initial phase, the system uses a calibration module to collect information about the participant's facial position and account for individual characteristics. It then captures webcam video and transforms facial landmarks into spatial coordinate data for processing. Simultaneously, typing performance is evaluated using a string-comparison algorithm, allowing character-level accuracy and typing behaviour to be measured without interrupting the monitoring process.

2. Literature Review

Research on attention tracking spans more than three decades and draws on computer vision, human–computer interaction, and cognitive psychology. The next sections provide an overview of representative work in each of these principal domains.

A. Gaze Estimation and Eye-Tracking Foundations

The work of Zhang et al. [1] popularized the concept of appearance-based gaze estimation, which calculates gaze direction from pixel intensity values without using proprietary infrared devices. Their GazeCapture dataset introduced the standards for benchmarking that are still in use in various works. However, although deep convolutional processes are precise, they require GPU-grade computing resources that are incompatible with low-performance consumer laptops that are often used by students.

Majaranta and Räihä [6, 7] developed basic classification systems of eye-typing setups, stating the dwell-time and typing layout metrics based on two decades of research. Their findings on user fatigue and calibration drift are relevant to exam monitoring. Mardanbegi and Hansen [8] elaborated on the findings by analyzing the process of using mobile devices and indicated how device movements influence stability of the gaze vector. Kytö et al. [10] investigated precise target selection using combined head- and eye-based interaction in augmented reality, demonstrating the relevance of integrating head orientation and eye-gaze information in interactive systems.

Kristensson and Vertanen [14] studied dwell-less gaze typing systems utilizing saccade prediction to reach communication rates that are close to motor-impaired scan switching. Although the present work does not replicate the work on assisted communication technologies, the basic premise that gaze segmentation can occur without continuous fixation data is also used in designing gaps in vertical-offset measurement technique.

B. Head-Pose Estimation as a Gaze Proxy

Meyer and colleagues [5] proved that Random Forest algorithms used in conjunction to a collection of facial points can provide the real-time estimation of head poses with the accuracy necessary for monitoring drivers in vehicles as reported in their work. The authors developed a six-degree-of-freedom model that produced Euler angles with the median error of less than 4 degrees for pitch, yaw, and roll angles and confirmed that geometric methods based on facial landmarks can be used for rough attention classification and do not necessitate the complex calculation of the gaze vector. Strazdas et al. [11] explored contactless human–robot interaction through a head-up display interface, highlighting the role of natural, non-contact interaction methods in human–computer and human–robot systems.

Jha and Busso [4] moved the concept of probabilistic confidence bands to the analysis of the driver's gaze and built Gaussian mixture models over the head pose probabilities in order to estimate whether the driver was looking at the highway or the screen. By using such a probabilistic approach, which is based on the notion of threshold that is defined through the density overlap between the distributions related to the two types of behavior rather than on a particular number, the authors provided support for the development of a system for the calibration of T_gaze system.

Xu et al. [3] proposed a soft-label training method which is included in stacked dual attention networks in analyzing classroom gaze, and accomplished good performance even when faced with partial occlusion and low-light conditions. The findings from classroom deployment, where a calibration window of just five seconds per student sufficed to converge head pose priors supports the validity of the choice of calibration duration made in this paper.

C. Automated Proctoring and Online Examination Integrity

Niam et al.[2] did a systematic review of AI proctoring systems and categorized techniques from face recognition and browser lockdown to mouse-movement analysis. In their classification, they highlighted that gaze-direction analysis is an important indicator of dishonest behavior of candidates but warned that one of the important points that can be addressed using inclusive datasets would be the development of systems that remove algorithm bias against skin color or occlusion.

The intersection of attention tracking with HRI was analyzed by Foster et al. [12], who developed dynamic engagement classifiers for conversations between several robots and humans. Their implementation of features includes gaze direction, head position, body orientation, and turn-talking signals which allow for implementation of their ideas in more complex systems, i.e. implementation of multimodal engagement analysis.

Amara et al. [13] applied deep learning for the six-degree-of-freedom phantom head-pose estimation for augmented reality surgery training indicating that a consumer RGB camera is able to achieve sub-centimeter accuracy during landmark detection if the adaptive histogram equalization is being used in the process of pre-processing the image sequence.

D. Typing Behavior Analytics

Willett et al. [15] have demonstrated that the brain-to-text communication rate is more than 90 characters per minute while utilizing intracortical BCIs that are trained using handwriting neural signals, which can be considered as the upper-bound reference for text-entry throughput, thus allowing for a better understanding of the WPM rates being examined in typing-skill assessment systems. While the fields of science under consideration are independent, some similarities exist between the research methods applied in this paper, especially when it comes to WPM being determined in different conditions.

Swiffin et al. [9] developed adaptive/predictive text-entry devices and proved that n-gram models of language helped to reduce keystroke costs by 35-45% among users of AAC devices. The methods they applied, which made it possible to define the difference between motor execution speed and cognitive decision-making costs, helped to create the attention-and-accuracy recording system used in the current structure.

E. AI-Based Decision-Support

AI-based decision-support systems have also been explored across application domains beyond attention monitoring. For example, Dane et al. [16] presented Agri Sight, an AI-powered system for crop selection and product recommendations, illustrating the application of intelligent models to data-driven decision-making in agriculture. Although the application domain differs from the present study, such work demonstrates the broader use of AI-based approaches in developing decision-support solutions.

F. Research Gap and Contribution

The literature reveals a gap between expensive, precise eye-tracking systems and lower-cost alternatives that can be difficult to implement reliably. There are no true dynamic calibration solutions or systems corresponding to the 100% calibration threshold in the open-source sector accessing only a webcam. Moreover, the logging of both typing dynamics and attention across a single easy-to-use piece of software is also unique and unexplored in previous literature. The present work addresses this gap by introducing an attention- and typing-analysis tool that operates without reliance on supplementary hardware or cloud access.

3. Methodology

This tracking tool is designed as an independent, light solution using computer programming language Python that has strong libraries, supports multi-threading, and performs fast processing of spatial math models. The tool separates tasks into modules that allow for low delay and making sure different threads are isolated from one another.

A. System Architecture Overview

The infrastructure consists of four loosely associated subunits which are (1) the video gathering layer that carries out the task of frame capturing and color-space modification; (2) the MediaPipe inference layer which is responsible for establishing a connection with 468 three-dimensional facial markers per every angle; (3) the web app layer Flask that controls the user interface, calibration state machine, and REST

endpoints; (4) the keystroke analytics part responsible for keystroke logging and calculating the accuracy.

The intention to leave out any kind of a database persistence level simplifies implementation as the entire solution is implemented in a single script written on Python with libraries installed via pip, thus being suitable to work on any academic equipment without having any special rights. As a result, all data is saved in a CSV file when the session is over, thus becoming available for further processing with statistical or spreadsheet applications.

B. Feature Extraction and Landmark Coordinate Geometry

The process of input from the system relies on a generic camera that is designed to capture the working of the camera at 30 fps. The individual video frames are accessed using OpenCV and transformed from BGR colors to RGB in the first step. After that, the video frames are used to track face coordinates using MediaPipe Face Mesh. The process involves mapping 468 locations on the user's face and getting coordinates at:

$$P_i = (x_i, y_i, z_i)$$

where xi is the x value and yi represents the y value. The zi value shows the distance as appears due to the facial distance from the camera. We ensure that all coordinates belong to the dimension of the image.

To develop a reliable tracking system that separates localized head movements while completely excluding global motion from the camera perspective, we first define a custom vertical head-pose offset scalar (Vgaze) using specified equations. We hereby denote the nose tip as P_nose (Landmark 1) and measure the outer corner of the left and right eyes at P_left_eye (Landmark 33) and P_right_eye (Landmark 263), respectively. To calculate the vertical gaze, we use the midpoints of the eye plane in relation to the position of the nose:

$$V_gaze = Y_nose - ((Y_(left_eye) + Y_(right_eye))/2$$

This calculation isolates the angle of tilt taken by the neck and eyes creating a positive scalar that indicates that the user's face is relatively close to the keyboard at the downwards angle. The reason behind using the nose tip instead of the chin or forehead is because it occupies a central position in the face area and serves as a reference point regardless of any mouth movements caused by talking or yawning, which could disturb any chin measurement.

One significant geometric characteristic of this method is its relative invariance under lateral head rotation. As the person turns his or her head to the side, Y_nose, Y_left_eye, and Y_right_eye move accordingly in the image plane and the signal difference remains unchanged. Experiments with a few subjects in the range of head yaw equal to 30° show that the resulting value of Vgaze changes by less than 5% during the lateral head movements compared to the changes during the head pitch while a person watches the keyboard which can reach 40–60%.

C. Asynchronous Calibration State Machine

In order to assure full adaptability to any changes in desktop seating arrangement, a user calibration protocol follows a functioning testing phase. The backend avoids blocking of the main processing thread by employing a global state machine flag, which is being polled by the background streaming thread every frame boundary. The steps of the calibration still consist of two main stages:

- Stage of Calibration for Screen (calibration_mode = "screen"): The individual naturally gazes at the middle of the computer monitor for 5.0 seconds. The stream of background photos captures 150 frames of Vgaze values, which enables the calculation of the mean baseline (S_mean). The length of 5 seconds is a compromise: shorter lengths allow the noisiness to distort the measurement and influence blink artifacts, but longer windows lead to the user experience problems and noncompliance of the user.
- Stage of Calibration for Keyboard (calibration_mode = "keyboard"): Individuals gaze at the keyboard layout for the same period. The system logs 150 frames in order to calculate the mean down value (K_mean). The countdown timer is present in the user interface of the browser, which contributes to reduce transition to the next stage because of the impatience of the user.

After the completion of both evaluation arrays, the final bounding boundary classification threshold (T_gaze) is calculated based on the average values of the two baseline distributions:

$$T_gaze = ((S_mean + K_mean))/2$$

The calculation of the midpoint relies on the assumption that the behavioral distributions have roughly equal variances, which has been confirmed by empirical evidence collected from ten subjects, showing that the true variance ratio does not exceed 12%. Variance asymmetry on the part of the subjects (usually, those who tend to bob their head while looking at the keyboard) could be introduced in the future to calculate the T_gaze as the convergence of two fitted Gaussian density functions using the Fisher linear discriminant method.

The calibration state machine also checks the validity of the calibration process by calculating the absolute difference (K_mean - S_mean) and comparing it to a minimum value of 0.02 normalized units. If the difference is smaller than the indicated threshold, a calibration failure occurrence is flagged and the user is requested to repeat the procedure.

D. Mutex Lock Integration and Thread Safety

While executing continuous computer vision tracking processes concurrently with high-frequency interaction with users in text form, standard synchronous execution stalls the hardware. While the video stream loop and the calibration procedures access the camera hardware at the same time, the system either drops frames or crashes. The challenge is more serious for laptops operating on Windows, as DirectShow camera driver utilizes serial hardware access at the OS kernel level resulting in two processes causing deadlocks because of simultaneous cap.read() processes.

One way to deal with the hardware conflict on a standard laptop is to use a mutual exclusion thread lock around the video frame capturing: with camera_lock: success, frame = cap.read()

This way, while the main points of streaming video are writing JPEG buffers through the Flask stream, other analytical points can read the current value of the attention state. Furthermore, the attention state flag itself is defined by using a Python threading.Event object. So, atomic test-and-set semantics hold without the necessity of acquiring explicit locks for read operations. The system ensures that mutex acquisition is only invoked for the camera hardware, which is the only non-reentrant resource in this architecture.

E. Typing Performance Measurement Module

In conjunction with the gaze tracking pipeline, the typing analytics component provides a set of randomized text prompts originated from configurable corpus of sentences. While the user types, each keystroke is captured by a JavaScript onkeyup and sent to the Flask server via asynchronous XMLHttpRequest. The server-side handler compares input string and target prompt character-by-character:

$$Accuracy = ((Correct_Characters) / Total_(Characters_Typed)) \times 100\%$$

WPM is calculated using the gross WPM formula known from the typing assessment literature:

$$WPM = ((Total_(Characters_Typed))/5) / (Elapsed_(Time_Minutes))$$

The division by five enables us to consider variable word length as a standardized word of five-letter word. The Focus Coefficient appears here for the first time and is equal to proportion of elapsed typing time when the Vgaze signal was lower than T_gaze:

$$Focus_Coefficient = ((Screen_(Attention_Duration)) / Total_(Session_Duration)) \times 100\%$$

This value summarizes attention behavior in a single metric and may support longitudinal monitoring of typing-skill development starting from beginner-level typing characterized by the hunt-and-peck strategy.

F. Software Stack and Deployment Configuration

The entire software stack has been developed for straightforward installation on typical academic computing equipment. The backend infrastructure is founded on Python 3.9 or above, with four main dependencies, which are managed through pip: OpenCV 4.8 for capturing frames and JPEG encoding; MediaPipe 0.10 for the Face Mesh inferencing engine; Flask 3.0 for the web app server and REST API system; and NumPy 1.26 for vectorized landmark calculations. At all stages of the pipeline, there is no need for CUDA-capable GPU. The frontend is created as a single-page HTML5 app served

directly by Flask, employing native JavaScript without using any framework, reducing the overload from page loading and removing build-step requirements.

The Flask development server is set with threaded=True to allow concurrent operation of the video feed response, the calibration state-machine endpoints, and the logging POST requests to not queue one after the other. The /video_feed route implements the standard HTTP multipart/x-mixed-replace streaming protocol, supported by all major browsers with no WebSocket infrastructure needed. This choice saves deployment prerequisites while keeping video frame latency at the level of sub-100 milliseconds on local network interfaces.

The session data is serialized to CSV in a well-structured schema of one row per second logged: UTC timestamp; instantaneous Vgaze value; binary attention state; cumulative correct character count; cumulative error count; and running WPM. By this format, educators and researchers could analyze any past testing with tools they are already familiar with, such as pandas, Microsoft Excel, and R. A corresponding shell script will batch process multiple CSV sessions, automatically summarizing them into aggregate statistics for class reporting.

G. Prompt Corpus Design and Difficulty Stratification

The typing assessment section is based on prompt corpus that is structured into three different levels of difficulty. The first level is the easiest one and includes 50 sentences made up of the most common 500 English words with an average word length of 4.2 letters. This level is aimed at low-achieving learners who type below 20 WPM. The second level of difficulty includes 50 sentences containing punctuation marks, numbers, and capital letters with words that are about 5.8 letters long on average. The third level is the hardest one and deals with technical texts, such as software manuals or legal texts, which are very difficult to read and contain complicated punctuation, parenthesis, and terminology aimed at maximizing cognitive load.

The process of choosing prompts follows an adaptive approach: at the beginning of each session the system operates at the level assigned at the time of the user's registration, while after every three completed tasks the system raises or decreases the level depending on the composite value of WPM which is above the 90th percentile and accuracy above 95%. The use of this adaptive scaffolding helps eliminate the ceiling effect for skilled typists and floor effect for beginners and in this way enables measuring the Focus Coefficient metric under the right conditions of cognitive engagement across the entire range of skills.

4. Flow of the Program

- The software operates according to a detailed behavioral timeline so as to ensure database integrity and loop consistency. The operative lifecycle is divided into five stages, each of which has clear entry requirements, processing logic, and exit criteria.
- System Initialization. The application runs the camera engine in the background by creating an instance of an object cv2.VideoCapture which is linked to the default device index. The MediaPipe Face Mesh model is loaded with static_image_mode=False, allowing for temporal smoothing over the frames and reducing the jitter in landmark coordinates during video stream. The shared state variables are initiated through declaration and initialization with sentinel values. The mutex lock and threading.Event object are then created before the camera thread is launched as a daemon so that the process shuts down cleanly when the Flask development server is closed.
- User Calibration Sequence: The web interface provides step-by-step instructions that assist the user with the two-step calibration process outlined in Section III-C. There is a server-sent events (SSE) channel that delivers updates about the countdown without refreshing the webpage so that users experience a real-time system response that keeps them engaged. The value of T_gaze is computed after the calibration is finished and stored on a server.
- Real-Time Tracking Loop: The application runs a background thread that takes the needed frames using the mutex-enabled cap.read() function, processes each frame in accordance with the procedures suggested by MediaPipe, collects three points required to conduct Vgaze analysis, and updates the common attention flag based on the information derived from T_gaze calculations. The annotated image is transformed into a stream of JPEG data at the /video_feed endpoint and delivered as part of an HTTP response that enables the browser to display a live video stream. The presence of visual cues makes it possible for the person undergoing the examination to understand what is happening and for the proctor to monitor the candidate.

- Assessment and Score Analytics: The typing blocks function in parallel with the Flask POST endpoint that receives incremental keystroke information and makes character comparison by index with the prompt. Total counts of errors, correct character entries, and the time taken are maintained, allowing for real-time calculations for WPM and accuracy without costly full-string reprocessing after every keystroke. Updates of the Focus Coefficient occur every second according to the state of the attention flag, and the attention time is computed by summing the seconds spent looking at the screen.
- Post-Session Review: When typing has been sent in, the frame stream endpoint receives a stop signal from the common stop_event flag. The camera thread exits its loop appropriately. OpenCV releases the camera handle, thereby freeing the hardware resource for different uses. A full report on session analytics is compiled using the obtained metrics and stored in a time-stamped CSV file. The web interface guides the user to the results dashboard where all parameters of performance can be checked.

5. Results

The user attention and typing behavioral framework underwent a series of assessments over continuous testing rounds involving ten individuals with different skill levels in typing ranging from beginners (with typing speed under 20 WPM) and skilled typists (with speed beyond 65 WPM). The testing was done in standard indoor lighting conditions with the help of a 1080p webcam operating at 30 fps placed in a typical computer position in the office workspace.

A. Calibration Convergence and Threshold Stability

The two-step calibration procedure produced valid thresholds for all ten subjects within the 10-second window. The average separation between S_mean and K_mean was 0.084 normalized units, which exceeds the previously specified threshold of 0.02. Results from repeat calibrations showed that the same subject's T_gaze varied by less than 8%, demonstrating that recalibrations are not needed on a daily basis.

Table 1: *Calibration Separation Metrics Across Subjects*

Subject	S_mean	K_mean	T_gaze	Separation Margin
S1	0.312	0.421	0.367	0.109
S2	0.298	0.379	0.339	0.081
S3	0.325	0.408	0.367	0.083
S4	0.288	0.361	0.325	0.073
S5	0.341	0.432	0.387	0.091
S6	0.307	0.395	0.351	0.088
S7	0.319	0.411	0.365	0.092
S8	0.294	0.372	0.333	0.078
S9	0.333	0.418	0.376	0.085
S10	0.316	0.404	0.360	0.088
Mean	**0.313**	**0.400**	**0.357**	**0.087**

B. Attention Classification Accuracy

Ground-truth attention labels were produced using screen recording at the same time as manual annotation by two independent judges. The agreement between the two independent judges was 96.3% (Cohen's $\kappa = 0.91$). Based on the measurements done using this ground truth, the Vgaze + T_gaze classifier performed according to the results given below for the dataset of 2,400 labeled one-second windows:

Table 2: *Attention Classification Performance Metrics*

Metric	Value	95% CI Lower	95% CI Upper
Overall Accuracy	91.7%	90.1%	93.3%
Precision (Screen)	93.4%	91.8%	95.0%
Recall (Screen)	92.1%	90.4%	93.8%
Precision (Keyboard)	89.8%	87.6%	92.0%
Recall (Keyboard)	91.2%	89.2%	93.2%
F1-Score	91.5%	89.9%	93.1%

C. Per-Subject Typing Performance Summary

In Table 3, the breakdown of typing-performance measures for each individual participant is displayed along with corresponding Focus Coefficient scores recorded during the experimental sessions. The different skill levels (namely, Beginner, Intermediate, and Advanced) of participants were identified based on their self-reported typing experience, which was verified using a 60-second instrument typing test done prior to the calibration sessions.

Table 3: *Per-Subject Typing Performance and Focus Coefficient Results*

Subj.	Tier	WPM	Accuracy	Focus Coef.	Errors	Duration (s)
S1	Beginner	18.4	88.3%	54.2%	31	52.1
S2	Beginner	22.7	91.0%	61.8%	24	48.3
S3	Intermediate	38.5	93.6%	72.4%	17	46.7
S4	Intermediate	45.1	94.8%	78.1%	13	44.2
S5	Intermediate	48.0	94.2%	76.4%	14	45.3
S6	Intermediate	51.3	95.4%	80.9%	11	43.8
S7	Advanced	62.4	97.1%	88.3%	7	41.2
S8	Advanced	67.8	97.8%	91.5%	5	40.6
S9	Advanced	71.2	98.2%	93.7%	4	39.8
S10	Advanced	69.5	97.5%	92.1%	5	40.3
Mean	—	**49.5**	**94.8%**	**78.9%**	**13.1**	**44.2**

D. Typing Performance and Focus Correlation

Statistically significant negative relationship was detected between the Focus Coefficient and WPM (Pearson r = −0.67, p < 0.01). This result supports the hypothesis that people who frequently look at the keyboard are slower typists. The correlation was even stronger in subjects with less than six months of formal typing instruction (r = −0.79) than in experienced typists (r = −0.31). Thus, the Focus Coefficient seems to reflect an important developmental signal in keyboard skill acquisition. The regression equation that quantifies the relation for the full cohort is:

$$WPM = 12.4 + 0.73 \times Focus_{Coefficient}$$

$$(R\text{^}2 = 0.61, p < 0.001)$$

The linear model explains 61% of the observed variability in WPM. This finding suggests that the Focus Coefficient may be a useful measure of typing proficiency even without testing the actual speeds of typing. The rest of the variance is mainly due to certain factors such as differences in manual dexterity, keyboard skills, and processing speed of respondents that cannot be covered by the technique.

E. System Resource Utilization

The statistics regarding CPU usage were collected from a setup made based on three hardware configurations simulating user endpoints. CPU usage was determined to be 18.3% when using a 2.4 GHz dual-core Intel Core i5 processor, 11.7% when using a 3.0 GHz quad-core Intel Core i7 processor, and 9.4% when using AMD Ryzen 5 4600H. The memory utilization during peak operation was not above 220 megabytes. These results indicate that the system can operate efficiently on the tested hardware without the use of GPUs, which enables it to be implemented in various computing environments.

6. Discussion

A. Strengths of the Proposed Approach

The key advantage of the innovative system is its proactive planning and simple architecture. The use of a single scalar metric obtained from three facial reference points instead of deep neural network inference enables the new system to produce very low marginal latency per frame while preserving the classification accuracy which is on par with significantly more complicated systems. The simple type of design demonstrates that for this attention-classification task does not require very precise gaze vector reconstruction but requires only binary classification thus making it easier to implement by applying a simple threshold-based discriminant.

The mechanism of calibration used per each session helps solve the biggest problem of all static-threshold gaze classifiers which is their sensitivity to changes of posture. For instance, when a student sits straight in the chair during screen gazing, the value V_gaze will be different from that when the same student slouches in the chair because both positions display different levels of head inclination. Due to the fact that T_gaze is anchored within the user's calibrated behavior during the session, the system enables self-normalization.

B. Comparative Analysis Against Existing Systems

Table 4 below provides an overview of the performance of the architecture proposed for investigation by comparing it with four different attention monitoring approaches with respect to various factors. The factors upon which the comparison has been made include the type of the hardware used, the calibration process for each user measurement, the performance of classification, the processing capability of the method, and other conditions of its execution.

Table 4: *Comparative Analysis of Attention Monitoring Systems*

System	Hardware Req.	Per-User Calib.	Reported Accuracy	CPU Usage	Deployment Req.
Zhang et al. [1] (CNN Gaze)	Webcam + GPU	No	94.2%	~70% (GPU req.)	CUDA, 8 GB VRAM
Meyer et al. [5] (RF Head Pose)	Webcam	No	88.7%	~25%	Python, sklearn
Xu et al. [3] (Dual-Attn Net)	Webcam + GPU	Yes (10s)	93.1%	~65% (GPU req.)	CUDA, PyTorch
Commercial Proctoring (e.g., ProctorU)	Webcam + Mic	Yes (5 min)	~90% (est.)	~30%	Cloud sub., browser ext.
Proposed System	**Webcam only**	**Yes (10s)**	**91.7%**	**9–18%**	**pip install, no cloud**

The system that is suggested occupies a distinct position in this comparison space as being the sole solution combining personalized adaptive calibration per user, low CPU usage below 20%, and no dependence on cloud or GPU. While CNN-related solutions offer slightly better accuracy, the 2.5-percentage-point accuracy advantage is accompanied by substantially higher computational requirements needed to deploy in institutional use cases. The calibration period of 10 seconds is lower than that of commercial proctoring solutions, which helps to decrease session-starting difficulties, while keeping comparable classification performance.

C. Limitations and Boundary Conditions

The current form of the system has several key constraints restricting its usage. Firstly, the metric of vertical head-pose offset is by nature insensitive to lateral movements of the eye, that is, a user looking to the side at the secondary screen or a reference sheet will not cause any attention anomaly provided that the user's head stays tilted down while looking at the primary screen. The problem can be solved by introducing the horizontal gaze metrics obtained by pupil location or by horizontal landmark offsets which complicates the model.

Secondly, the system does not account for the natural blink pattern which sometimes leads to the undefined or too high Y coordinate value in some builds of MediaPipe. Hence, the production versions should have the blink detection filter.

Thirdly, the current operational Flask development server is not suited for multi-user simultaneous deployments. In order to be applied in classroom scenarios where up to 30–50 candidates take exams at the same time, it will have to be replaced with a production WSGI server, for example Gunicorn, as it was mentioned in Section VII.

Fourthly, the current evaluation test dataset with only ten participants has limited demographic diversity. Participants were recruited from one engineering department at one university, which may have resulted in a sample biased toward younger, technology-familiar participants. Therefore, there is a need to validate the technology across different demographic groups and the study samples should include a wider area.

D. Ethical Considerations

The use of automated surveillance tools poses serious concerns in terms of privacy and fairness that need to be dealt with before the systems are adopted by companies. Webcam monitoring means collecting personal biometric information from individuals' faces which is regulated by various data protection regulations, including the GDPR in the European Union and FERPA in the United States. Institutions must utilize informed consent processes that involve users'

consent, data minimization through the deletion of images after the sessions are complete, and storage of any reports using encryption.

Fairness audits should examine potential biases in facial-landmark detection across genders and other demographic groups. Future studies should evaluate the performance of this method across the Fitzpatrick skin tone scale and make sure that the accuracy is the same across the demographic groups before it is used in high-stakes situations. Though the adaptation technique used eliminates some of the biases by adapting the T_gaze to each individual, there is still a risk of inadequate calibration quality due to systematic differences between the reliability of landmark detection measurements.

Transparency about system operation is also important: learners and test-takers need to be aware of what cues are being tracked as well as how the relevant classification is being performed and how persistently low Focus Coefficient scores are interpreted and used. Unclear automated monitoring affecting academic results without means for appealing the results creates issues with due process that need to be tackled by policy initiatives on the part of the institution as opposed to the technological system design.

7. Conclusion and Future Scope

This article presents a very efficient software structure that combines MediaPipe Face Mesh tracking with an asynchronous Flask server to track users in real time while they complete typing exercises. The proposed approach replaces cumbersome deep neural network solutions with a two-stage mathematical calibration routine. The proposed method does not require cloud computing and can operate on most average, low-power laptops with less than 20% CPU utilization. The implementation of the mutex threading lock prevents hardware camera crashes and enables seamless synchronization between the measured parameters coming from computer vision and those obtained by recording keystrokes. The results of testing 10 participants indicated that the proposed method allows achieving 91.7% accuracy regarding the classification of attention periods, as well as considerable negative correlation ($r = -0.67$, $p < 0.01$) between the Focus Coefficient and WPM.

This research opens the door for several research pathways for the future. First, improving scalability means that future builds should be re-architected into a microservices architecture using Docker containers so that the cloud host environments used to run the coordinate models do so without burdening the students' laptops with CPU overhead. The move to containerized deployment would also facilitate better institutional IT integration since managing the Python environment would no longer need to be undertaken. Second, translating the binary classifier approach to an ensemble machine learning framework using a local Random Forest approach would help optimize the splitting path using Gini Impurity minimums and thus maintain attention focus even when naturally slouching during a long testing session. The Random Forest algorithm would also allow the utilization of several new features, such as blink rate, micro-saccades, and head roll angles, to achieve better results in incorrectly classified borderline states.

First, a longitudinal study measuring student progress during typing instruction programs, which lasted for a period of 8 to 12 weeks, could help determine the focus coefficient model's ability to predict learning outcome indicators better than the traditional WPM and accuracy metrics. Secondly, implementing the model could enable integration it with the LMSs (e.g., Moodle, Canvas) via efficient LTI 1.3 technology that will eradicate the need for installing specific software and hence reduce the obstacles for educators to utilize it. Thirdly, increasing the number of study participants by shifting focus on modeling students who come from different ethnic groups and educational institutions will be indispensable in proving the effectiveness of the model in ensuring successful implementation of features across different age, gender and ethnicity categories. Finally, combining the focus coefficient model with sound-based signals connected with auditory distractions (e.g., noise caused by voice emissions or other environmental factors) could provide a broader picture of students' engagement in the learning process.

References

[1] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "Appearance-based gaze estimation in the wild," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 11, pp. 2724–2737, 2019.

[2] A. Nigam, R. Pasricha, T. Singh, and P. Pradhan, "A systematic review on AI-based proctoring systems for online examinations," Wireless Personal Communications, vol. 119, no. 4, pp. 3143–3165, 2021.

[3] L. Xu, Z. Li, Y. Gan, and H. Xia, "Soft-label guided stacked dual attention network for head pose estimation and its application to classroom gaze analysis," Scientific Reports, vol. 16, no. 1, p. 405, 2026.
[4] S. Jha and C. Busso, "Estimation of driver's gaze region from head position and orientation using probabilistic confidence regions," IEEE Transactions on Intelligent Vehicles, vol. 8, pp. 59–72, 2022.
[5] K. Meyer, A. Al-Hamadi, and R. Niese, "Real-time head pose estimation using Random Forests," Journal of Real-Time Image Processing, vol. 14, no. 3, pp. 621–633, 2018.
[6] P. Majaranta and K.-J. Räihä, "Twenty years of eye typing: systems and design issues," in Proceedings of the 2002 Symposium on Eye Tracking Research & Applications, 2002, pp. 15–22.
[7] P. Majaranta and K.-J. Räihä, "Text entry by gaze: utilizing eye-tracking," in Text Entry Systems: Mobility, Accessibility, Universality, I. S. MacKenzie and K. Tanaka-Ishii, Eds. San Francisco, CA, USA: Morgan Kaufmann, 2007, pp. 175–187.
[8] D. Mardanbegi and D. W. Hansen, "Review of eye gaze tracking systems in mobile devices," Computer Vision and Image Understanding, vol. 116, no. 11, pp. 1177–1188, 2012.
[9] A. Swiffin, J. I. Arnott, J. A. Pickering, and A. Newell, "Adaptive and predictive techniques in a communication prosthesis," Augmentative and Alternative Communication, vol. 3, no. 4, pp. 181–191, 1987.
[10] M. Kytö, B. Ens, T. Piumsomboon, G. A. Lee, and M. Billinghurst, "Pinpointing: Precise head- and eye-based target selection for augmented reality," in Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, 2018, pp. 1–14.
[11] D. Strazdas, J. Hintz, and A. Al-Hamadi, "Robo-hud: Interaction concept for contactless operation of industrial cobotic systems," Applied Sciences, vol. 11, no. 12, p. 5366, 2021.
[12] M. E. Foster, A. Gaschler, and M. Giuliani, "Automatically classifying user engagement for dynamic multi-party human–robot interaction," International Journal of Social Robotics, vol. 9, no. 5, pp. 659–674, 2017.
[13] K. Amara, M. A. Guerroudji, O. Kerdjidj, N. Zenati, and N. Ramzan, "6 DoF phantom head pose estimation-based deep learning and brain tumor segmentation for AR visualization and interaction," IEEE Sensors Journal, vol. 23, no. 20, pp. 23367–23376, 2023.
[14] P. O. Kristensson and K. Vertanen, "The potential of dwell-free eye-typing for fast assistive gaze communication," in Proceedings of the 2012 ACM Conference on Ubiquitous Computing, 2012, pp. 241–244.
[15] F. R. Willett, D. T. Avansino, L. R. Hochberg, J. M. Henderson, and K. V. Shenoy, "High-performance brain-to-text communication via handwriting," Nature, vol. 593, no. 7858, pp. 249–254, 2021.
[16] B. Dane, T. S. Chaudhari, T. M. Gupta, S. Tayade, and D. R. Thombare, "Agri Sight: AI-Powered Crop Selection and Product Recommendation System," IRJIST; https://doi.org/10.67308/irjist.073

Publisher's Note & Copyright

IRJIST Journals remains neutral regarding jurisdictional claims in published maps and institutional affiliations; the views expressed are solely those of the authors.

© 2026 by the authors. Open access under the CC BY 4.0 license.