Multidisciplinary Journal of Engineering and Technology
Print ISSN: 3007-6897
Online ISSN: 3007-6900
DOI: https://doi.org/10.61784/mjet3039

DESIGN AND IMPLEMENTATION OF AN EXAMINATION ANTI-CHEATING SYSTEM IN COMPUTER LABORATORIES BASED ON LIGHTWEIGHT COMPUTER VISION

ZiRui Wu, FanHao Ye, ZhenNan Zhang, LuTing Wu, XinYin Lan, MuCong Chi*
School of Artificial Intelligence, Wenzhou Polytechnic, Wenzhou 325035, Zhejiang, China.
**Corresponding Author: MuCong Chi*

Abstract: With the deep advancement of educational informatization, computer-based examinations in university computer laboratories have become a critical approach for evaluating students' practical skills. However, traditional centralized video invigilation systems impose severe computation and network bandwidth strains, making them impractical for legacy hardware deployments. To address these bottlenecks, this paper designs and implements a lightweight, multi-dimensional behavior auditing invigilation system based on decentralized edge computing. By offloading image capturing, posture estimation, and decision-making workflows entirely onto the localized client side, the proposed system restricts network transmissions to low-frequency, structured text logs, fundamentally eliminating the risk of campus network congestion caused by concurrent high-definition video streaming. Specifically, on the algorithmic layer, the system leverages the native features of the 4×4 homogeneous transformation matrix from the WebAssembly-accelerated MediaPipe Tasks-Vision framework. It performs inverse kinematics for head yaw and pitch via basic scalar operations in the JavaScript runtime, completely bypassing heavy dependencies such as OpenCV.js. Concurrently, a geometric mathematical model utilizing Euclidean distance ratios of iris and pupil landmarks is established for eye gaze tracking. Furthermore, a dual-layer cooperative anti-false-alarm mechanism integrating spatial directional branching and a temporal sliding window filter is proposed, which smoothly filters out transient physiological noises and benign gestures. Experimental results on a simulated low-end host demonstrate that the system achieves an exceptionally low single-frame computation latency of 0.62 ms and restricts the total memory footprint to 146.5 MB. Under steady-state operations, it reduces the false positive rate (FPR) of environmental and physiological noise to 1.3% while maintaining a high true positive rate (TPR) of 96.3% for actual cheating behaviors. The system exhibits outstanding efficiency, robust anti-interference capability, and high deployment feasibility for inclusive application on legacy educational equipment.
Keywords: Lightweight computer vision; Behavior auditing; Face landmarker; Sliding window filter

1 INTRODUCTION

With the profound advancement of educational informatization, computer-based examinations in university computer laboratories (such as final exams for professional courses, computer-based non-paper certification exams, etc.) have become an essential means to evaluate students' practical and hands-on capabilities. However, traditional invigilation modes in computer labs rely heavily on manual patrolling. Facing densely arranged workstations, invigilation personnel are highly susceptible to visual fatigue. Moreover, it is exceedingly difficult for human proctors to fully cover subtle and covert misconducts, such as "turning the head to peer at neighboring screens" or "lowering the head to read cheating materials."

Existing commercial intelligent invigilation systems typically require expensive infrared eye-trackers or high-performance computing servers, making them difficult to popularize on a large scale in ordinary school training computer rooms. Therefore, researching a "low-hardware-cost" anti-cheating system that can directly utilize the built-in cameras or ordinary USB cameras of computer labs possesses vital practical significance. By leveraging lightweight computer vision technology, this system achieves low-latency, high-precision monitoring of students' body postures and gaze trajectories without modifying any computer lab hardware. While reducing the labor cost of invigilation, it effectively safeguards the fairness and seriousness of campus examinations.

2 TECHNICAL OVERVIEW

2.1 Edge-Based Computer Vision and MediaPipe Tasks-Vision Framework

Traditional deep learning object detection and human posture recognition models (such as OpenPose, ResNet, etc.) usually contain tens of millions of model parameters. They exhibit an extremely high dependence on hardware graphical processing units (GPUs), making them unable to run smoothly in real-time on the low-end central processing units (CPUs) commonly configured in university computer labs. To break through this engineering bottleneck for practical deployment, this paper introduces Google's next-generation advanced modular machine learning framework, MediaPipe Tasks-Vision [1].

MediaPipe Tasks-Vision decouples and encapsulates complex machine vision pipelines into semantic-clear "Tasks". The

core of the proposed system selects the Face Landmarker task [2]. Internally, this task relies on a single-stage face detection model optimized specifically for edge devices. Through a single forward propagation, it can instantly regress up to 478 high-precision three-dimensional facial landmarks and natively integrates iris tracking algorithms.
Compared with the legacy traditional MediaPipe JavaScript SDK, the new Tasks-Vision framework runs on the latest WebAssembly (WASM) compilation standard within the Electron client environment (Chromium kernel) [3]. By introducing SIMD (Single Instruction Multiple Data) and multi-threading hardware acceleration mechanisms, it dramatically minimizes the memory copy overhead between the V8 engine and the underlying C++ source code [4]. Consequently, even under the legacy or low-end CPU environments frequently found in universities, the system successfully maintains a real-time inference frame rate of above 30 FPS with less than 20% processor load, establishing a solid foundation for the lightweight algorithmic base.

2.2 Asynchronous Backend Development with FastAPI and WebSockets

In computer lab examination scenarios, the anti-cheating system must perform uninterrupted real-time monitoring facing dozens of examinees simultaneously. Although the core AI vision solution has been fully offloaded to the examinee's standalone edge client (Electron), the cheating suspect status converted at high frequencies from the algorithmic layer (e.g., generating dozens of text logs per second regarding head tilt, squinting, or screen switching) still needs to be immediately uploaded to the teacher's main control terminal for centralized auditing. Traditional synchronous blocking back-end frameworks (such as Django, Flask) are highly prone to thread exhaustion when handling dozens of concurrent high-frequency HTTP requests, leading to server response latency or even crashes.
To guarantee absolute low latency for data transmission within the computer lab local area network (LAN), this paper selects FastAPI, an asynchronous full-stack Web framework based on Python 3.9 [5]. Built upon the ASGI (Asynchronous Server Gateway Interface) standard and backed by Starlette architecture and the Pydantic data validation library, FastAPI natively supports the async/await asynchronous coroutine mechanism, enabling it to process tens of thousands of concurrent connections with minimal memory consumption.
Furthermore, regarding the data transmission protocol, this system abandons the traditional HTTP polling mode and adopts the full-duplex WebSockets protocol [6]. WebSockets establishes a persistent long connection between the client and the teacher's server. Its data frame header is extremely small (only a few bytes), and it allows the server to actively push instant alerts to the frontend. This restricts the end-to-end latency from the occurrence of a cheating event to the highlighted alarm on the teacher's dashboard within milliseconds.

2.3 Cross-Platform Desktop Client via Electron

As a comprehensive examination anti-cheating system, relying solely on web-based camera acquisition possesses severe software-level vulnerabilities, as examinees can easily switch tabs (Alt+Tab), utilize multi-screen extensions, or use shortcuts to open other chat applications and browsers to search for answers directly. Therefore, the system must possess the software engineering capability of "strong regulation" at the client side.
This paper chooses Electron [7], a cross-platform desktop application development framework based on Node.js and the Chromium browser, to construct the examinee's test-taking frontend. Electron allows developers to write highly interactive interfaces using modern frontend technologies (Vue.js, HTML5, CSS3) while directly invoking the low-level APIs of the operating system through the underlying Node.js runtime (via C++ plug-ins or Inter-Process Communication, IPC).
In the practical application of anti-cheating business, the Electron client primarily undertakes two core tasks:
1. System-Level Kiosk Mode: By invoking main process APIs [8], the examination window is locked to full screen and always on top. It disables the Windows key and conventional tab-switching combinations via low-level hooks. Regarding the kernel-protected Ctrl+Alt+Del sequence, because it cannot be bypassed from the OS base level, the client handles it via software-level blur event tracking and modal alert restrictions, forcing the examinee's vision and software operations to remain strictly within the exam interface.
2. Multimedia Flow Regulation and AI Scheduling: Through the WebRTC interface of the Chromium kernel [9], the local USB camera video stream is securely captured and injected at a high frame frequency into the @mediapipe/tasks-vision module introduced via NPM [10]. The system finishes behavior evaluation standalone on the local machine and then reports structured anomalous results, which delivers a smooth and seamless user interaction experience while ensuring absolute exam integrity.

3 SYSTEM DESIGN AND IMPLEMENTATION

3.1 Overall System Software Architecture Design

To ensure the low-overhead operation of the system under the legacy hardware environments of university public computer laboratories, and to guarantee absolute low latency during the concurrent transmission of multi-channel invigilation data, this system abandons the traditional architecture of "centralized video stream uploading and unified cloud-side computing inference." Instead, it adopts a four-layer software engineering architecture featuring "decentralized edge computing + lightweight central data auditing." This architecture completely offloads the image resolution and behavioral decision-making workflows, which heavily consume computational resources, to the

examinee's standalone edge client. The central server only undertakes the tasks of low-frequency structured alert log reception and global status visualization. The topological design of the four core layers of the system is delineated as follows:

1. Hardware Perception Layer: This layer directly relies on ordinary USB cameras or built-in laptop cameras configured at the examinee's workstation. Through the WebRTC media interface within the Electron client's rendering process, the system silently captures the original frame-by-frame video stream at a resolution of 640×480 and 30 FPS. This layer only maintains the raw video frames in memory without performing video encoding or network transmission operations, thereby achieving zero local area network (LAN) bandwidth consumption.
2. Edge Algorithmic Layer: As the core computational power layer of the system, this layer runs entirely within the examinee's local Electron client. By introducing the @mediapipe/tasks-vision dependency, the system initializes the FaceLandmarker asynchronous task. Under the underlying WebAssembly environment, this task performs single-channel forward propagation inference on the video frames input from the perception layer, in-situ regressing 478 high-precision three-dimensional facial landmarks, including the iris.
3. Business Determination and Filtering Layer: Running likewise on the local client, this layer is responsible for converting the raw 3D coordinates and high-dimensional homogeneous matrices output by the algorithmic layer into concrete anti-cheating business determination logic. It comprises two core mathematical modules: the "facial homogeneous matrix posture solver" and the "iris gaze ratio calculator." To prevent false positives triggered by examinees' occasional eye-rubbing, tilting their heads to ponder questions, or normal cervical spine movements, this layer specifically introduces a sliding window temporal filter based on a First-In-First-Out (FIFO) queue to perform smoothing and denoising on instantaneous cheating signals.
4. Data Visualization and Application Layer: When the filtering layer confirms that an examinee has engaged in persistent non-compliant behavior, the Electron client immediately transmits structured, lightweight cheating logs (containing workstation numbers, cheating types, and timestamps) to the central master control terminal via the WebSockets (full-duplex network communication protocol) protocol. The teacher terminal receives the data based on a Python FastAPI backend and dynamically highlights the status of the corresponding non-compliant workstation with a red bounding box via a front-end visualization dashboard, issuing dual audio-visual alarms to the invigilator.

3.2 Mathematical Modeling of Core Algorithms

3.2.1 Three-dimensional head pose estimation model

The estimation of facial orientation is traditionally formulated as a classic Perspective-n-Point (PnP) problem in computer vision [11]. Conventional engineering implementation strategies typically require configuring the bulky OpenCV.js library on the client side to match and solve 2D pixel points against standard 3D spatial points using its built-in-iterative solvers, which severely escalates the memory footprint and CPU load of the frontend runtime.

To achieve ultra-lightweight engineering deployment, this system fully exploits the underlying feature output potential of the FaceLandmarker task within @mediapipe/tasks-vision. During the client initialization phase, the native facial transformation matrix output function is enabled by configuring outputFacialTransformationMatrixes: true. When this task executes forward propagation in the WebAssembly edge environment, its underlying algorithm automatically completes the three-dimensional rigid transformation solution based on sparse anchors, directly returning a 4×4 homogeneous transformation matrix $\mathbf{M}$ to the frontend [12]. Within the frontend JavaScript runtime, this matrix is stored as a one-dimensional column-major array, whose mathematical structure is formulated as follows:

$$\mathbf{M}=\begin{bmatrix} m[0] & m[4] & m[8] & m[12] \\ m[1] & m[5] & m[9] & m[13] \\ m[2] & m[6] & m[10] & m[14] \\ m[3] & m[7] & m[11] & m[15] \end{bmatrix} \tag{1}$$

Wherein, the upper-left 3×3 matrix $\mathbf{R}$ represents the precise facial spatial rotation matrix. Due to the column-major alignment of the array, the components R_{13}, R_{23}, and R_{33} strictly correspond to $m[8]$, $m[9]$, and $m[10]$ in the array, respectively. Utilizing the Electron rendering process, the system directly derives the yaw angle (Yaw, left-right head rotation) and pitch angle (Pitch, up-down head tilt) by combining the matrix components with the camera coordinate system characteristics. The mathematical derivation formulas are as follows:

$$\text{Yaw}= \arctan 2\,(-m[8], m[10]) \times \frac{180}{\pi} \tag{2}$$

$$\text{Pitch} = \arcsin\,(-m[9]) \times \frac{180}{\pi} \tag{3}$$

It should be particularly emphasized that the negative sign ($-m[8]$) in the yaw angle formula is an engineering correction introduced to adapt to the mirrored front-facing cameras commonly found in computer labs and the standard camera coordinate system direction. This design allows the client to perform lightweight scalar operations entirely dependent on the native high-dimensional outputs of MediaPipe without loading any third-party matrix calculation libraries, compressing the spatial resolution time of the head pose to under 1 millisecond. Combined with ergonomic sampling testing and sitting posture distribution data analysis in standard examination environments, the system establishes rational determination thresholds: when the absolute value of the yaw angle satisfies $|\text{Yaw}|>30^\circ$ (determined as turning the head to peer at neighboring screens) or the pitch angle satisfies $\text{Pitch}<-25^\circ$ (determined as excessively lowering the head to read cheating materials on the lap), the algorithmic layer outputs a "head posture anomaly" signal to the subsequent stage.

3.2.2 Iris gaze deviation tracking model

Certain examinees refrain from large head movements when cheating, instead employing the covert method of "squinting at neighboring desks" to peer at answers. To counteract such behaviors, this system establishes a spatial geometric mathematical model based on the relative positions of binocular iris landmarks to ensure the integrity of feature extraction.

Utilizing the FaceLandmarkerResult from @mediapipe/tasks-vision, the system accurately localizes the examinee's ocular regions. To eliminate the depth information z-axis noise interference caused by the examinee's forward-backward head displacements or forward body leaning in the three-dimensional space, the system projects the facial landmarks onto a two-dimensional image coordinate plane for scalar ratio computation. The system selects the left boundary point (facial landmark 33) and the right boundary point (facial landmark 133) of the left orbital region, alongside the left pupil center point (facial landmark 468) extracted via the iris tracking functionality (the right eye symmetrically corresponds to facial landmark 473). In the horizontal direction, the gaze deviation ratio formula R_{gaze} is constructed as follows:

$$R_{gaze}=\frac{\|P_{pupil}-E_{left}\|_2}{\|E_{right}-E_{left}\|_2} \tag{4}$$

Wherein, $\|\cdot\|_2$ denotes the Euclidean distance between two points in the two-dimensional image coordinate system (x,y), which is calculated as:

$$\|P_1-P_2\|_2=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2} \tag{5}$$

✧ Ideal State: When an examinee looks directly ahead at the screen to answer questions, the pupil is positioned at the horizontal center of the eye orbit, yielding $R_{gaze}\approx 0.5$.

✧ Cheating State: When an examinee extremely squints leftward or rightward toward neighboring desks, the pupil boundary approaches the corner of the eye. Combined with experimental data analysis of exam behavior sampling, the system defines a dynamic auditing boundary: when $R_{gaze}<0.35$ or $R_{gaze}>0.65$, it is determined that the examinee's gaze has severely deviated from the current examination screen, and the algorithmic layer outputs a "gaze anomaly" signal.

3.3 Implementation of Anti-False-Alarm Business Logic and Sliding Window Temporal Filter

In authentic school computer lab environments, examinees inevitably exhibit normal physiological activities during examinations lasting several hours (such as rubbing eyes, twisting necks to relieve fatigue, or temporarily looking up in a daze while contemplating questions). If the system were to directly determine cheating and trigger an alarm based solely on a single-frame threshold violation from the algorithmic layer, it would yield an exceptionally high false positive rate, severely disrupting exam order. To address this core pain point in software engineering deployment, this system designs a sliding window temporal filter within the business logic layer.

The system maintains a First-In-First-Out (FIFO) queue Q with a fixed length of $N=30$ in the client memory for each examinee. Since the camera acquisition frequency is 30 FPS, this queue comprehensively records the examinee's continuous physical postures within the past 1 second. Each frame of the image, after being resolved by the algorithm in Section 3.2, yields a boolean instantaneous state value f_{frame} (where 1 indicates anomaly and 0 indicates normalcy), which is appended to the tail of the queue while the oldest data at the head of the queue is evicted. The system periodically computes the proportion of anomalous frames P_{cheat} within the current window:

$$P_{cheat}=\frac{1}{N}\sum_{i=1}^{N} f_i \tag{6}$$

The system defines the security auditing rule: only when the proportion of anomalous frames within the sliding window satisfies $P_{cheat}\geq 70\%$ (i.e., at least 21 frames out of the most recent 30 frames continuously trigger anomalies) will the system confirm this as a genuine cheating incident. This design effectively filters out occasional eye-rubbing or normal head-turning movements whose duration is typically less than 0.3 seconds, thereby guaranteeing the fairness and impartiality of the system evaluation.

4 SYSTEM TESTING AND RESULTS ANALYSIS

In order to verify the feasibility and lightweight operational performance of the decentralized anti-cheating system designed in this paper under the legacy hardware environments of university public computer laboratories, as well as the sensitivity of the multi-dimensional behavioral auditing mechanism to real cheating behaviors and its anti-interference capability against physiological noises, a highly simulated digital examination room experimental environment was constructed. Quantitative performance evaluations were then conducted on the core indicators of the system.

4.1 Experimental Environment and Test Settings

The performance and accuracy testing of this system completely focused on the examinee's standalone edge client. To maximize the simulation of typical "low-configuration and legacy" hardware characteristics in university public computer labs, an early-released, low-configuration compact desktop host was intentionally selected as the baseline testing equipment. The specific experimental software and hardware environment configurations of the system are detailed in Table 1.

Table 1 Classification of Experimental Elements and Detailed Configuration of Core Parameters and Components

Experimental Element Category	Core Parameters and Component Configuration Details
Testing Hardware Equipment	CPU: Intel Core i3-6100 @ 3.70GHz Memory: 8GB DDR3 1600MHz Graphics Card: Intel HD Graphics 530 Camera: Ordinary external USB camera
System and Runtime Environment	Operating System: Windows 10 Professional 64-bit Application Architecture: Electron v28.0.0 Runtime Kernel Edge Algorithmic Dependency: @mediapipe/tasks-vision v0.10.8 (WebAssembly Version)

4.2 Edge-Side Lightweighting and Resource Consumption Testing

In the practical engineering deployment design in Chapter 3, the system enabled the native homogeneous transformation matrix output of the underlying MediaPipe by configuring outputFacialTransformationMatrixes: true. The algorithmic layer directly extracts components from a one-dimensional column-major array to perform inverse solutions at the scalar level, eliminating the need to dynamically load and compile heavy third-party vision libraries (such as OpenCV.js) that are tens of megabytes in size on the frontend.

To evaluate the actual resource consumption of this lightweight architectural design on a low-end edge device, the Electron client was launched on the low-spec host shown in Table 1. An external USB camera was connected, and the input source was adjusted to a standard 640×480 resolution, 30 FPS frame-by-frame video stream. The system was kept running continuously under a high workload for 30 minutes. Every 10 seconds, the single-frame processing latency, CPU core utilization rate, memory overhead, and actual throughput frame rate (FPS) of the algorithmic layer were read via the operating system's Performance Monitor and the core V8 engine's memory snapshot tool. The statistical results are shown in Table 2.

Table 2 Performance Evaluation Metrics and Statistical Results

Performance Evaluation Metric Name	Steady-State Operational Average	Peak Range Interval
Single-frame spatial geometric resolution latency	0.62 ms	0.55 ms ~ 0.81 ms
Electron rendering process CPU utilization rate	12.1%	11.2% ~ 14.5%
Client overall memory footprint	146.5 MB	141.2 MB ~ 152.0 MB
Real-time image throughput frame rate (FPS)	29.7 FPS	29.1 FPS ~ 30.0 FPS

From the steady-state test data in Table 2, the following conclusions can be drawn: First, regarding resolution efficiency, the average single-frame spatial geometric resolution latency of the system is a mere 0.62 ms. This is because the column-major transformation matrix **M** has already been synchronously regressed along with the forward propagation inference in the underlying WebAssembly compilation environment; the frontend JavaScript runtime only executes basic arithmetic operations and the Math.atan2 scalar function, completely avoiding the serialization and memory copy overhead generated by passing a large batch of 3D landmarks to high-dimensional iterative solvers in traditional engineering. Second, in terms of host resource overhead, the average CPU utilization rate of the system on the dual-core i3 processor is controlled at 12.1%, and the overall operational memory overhead stabilizes at around 146.5 MB. This data demonstrates that after successfully breaking free from the dependency on massive machine vision libraries, the system not only significantly streamlines the heap memory overhead of the Electron rendering process but also suppresses computational power loss to an extremely low level. Throughout the entire testing cycle, the image throughput rate remained stable at 29.7 FPS, a full-frame state close to the upper limit of the physical camera, without any interface stutters caused by memory leaks or CPU overload. This fully validates the feasibility of implementing an imperceptible, low-load deployment of this system under the environment of legacy public computer laboratories in universities.

4.3 Multi-Dimensional Behavior Auditing Accuracy and Anti-False-Alarm Capability Testing

To verify the sensitivity of the spatial geometric mathematical models established in Section 3.2 (head yaw angle, pitch angle, and binocular iris gaze ratio) in capturing real cheating behaviors, and to emphatically test the smoothing and interception effects of the "dual-layer cooperative anti-false-alarm mechanism" designed in Section 3.3 (spatial geometric directional branching + temporal sliding window filter) against authentic physiological noises in university computer labs, 10 volunteers were recruited to conduct behavioral sampling evaluations for standard cheating and non-cheating actions.

Based on the actual workstation layout and ergonomic characteristics of university computer labs, the test behaviors of the volunteers were classified into two major categories:

1. Physiological normal interference actions (testing anti-false-alarm capability): Transient eye-rubbing (lasting approximately 0.3 seconds, which easily triggers instantaneous gaze ratio threshold violations) and looking upward to contemplate questions (lasting approximately 1.2 seconds, which has a long temporal span but an opposite direction).

2. Typical non-compliant cheating actions (testing cheating detection rate): Prolonged head-turning to peer at neighboring screens (lasting > 3.0 seconds), large-angle head-lowering to read materials on the lap (lasting > 3.0 seconds), and maintaining a still face while extremely squinting toward neighboring desks (lasting > 3.0 seconds).

The evaluation sampled a cumulative total of 300 times each for normal physiological interference and non-compliant cheating actions. To quantify the actual contribution of the dual-layer cooperative anti-false-alarm mechanism, the test was conducted under two modes for ablation evaluation: Mode 1 (the cooperative anti-false-alarm mechanism is disabled, meaning an alarm is triggered immediately once a single-frame geometric threshold is exceeded); Mode 2 (the spatial directional branching and the sliding window filter with N=30,P_{cheat}≥70% designed in this system are enabled). A comparative analysis of the actual True Positive Rate (TPR) and False Positive Rate (FPR) of the experiments is shown in Table 3.

Table 3 Comparative Analysis of Ablation Experimental Results for the Anti-False-Alarm Mechanism

Action Type and Auditing Sample Distribution	Evaluation Metric Type	Mode 1 (Single-frame threshold violation immediate alarm scheme)	Mode 2 (The dual-layer cooperative mechanism of this system)
Normal Physiological Interference Actions (Eye-rubbing/Upward head-tilting in deep thought, 300 times in total)	Number of triggered false alarms	136 times	4 times
	False Positive Rate (FPR)	45.3%	1.3%
Typical Non-Compliant Cheating Actions (Prolonged head-turning / head-lowering / squinting, 300 times in total)	Number of correct diagnoses	296 times	289 times
	True Positive Rate (TPR)	98.7%	96.3%

Analyzing the ablation experimental data and the internal decision logs of the client in Table 3 allows for a profound clarification of the inner operational mechanism of the multi-dimensional cooperative auditing mechanism: In Mode 1, where the cooperative anti-false-alarm mechanism is not incorporated, although the system maintains an exceptionally high detection rate (98.7%) for cheating actions, its false positive rate for normal actions is as high as 45.3%. The logs indicate that when volunteers performed the eye-rubbing action, the instantaneous gaze ratio R_{gaze} derived by the system dropped below the safety boundary of 0.35 due to eyelid occlusion. The noise impact of single-frame data directly led to high-frequency false alarms from the system. If such performance were directly applied to a real examination room, it would severely disrupt normal invigilation order. In Mode 2, where the dual-layer cooperative mechanism of this system is enabled, the system demonstrates excellent noise branching and smoothing capabilities, with the false positive rate plummeting from 45.3% to 1.3%. Because real cheating behaviors (such as sideways copying or prolonged reading of materials) possess strong temporal continuity (with durations generally greater than 3.0 seconds, far exceeding the sliding window's 0.7-second determination threshold line), the proportion of suspicion within the queue can rapidly and stably soar to 100%. This allows the True Positive Rate of this system to still firmly remain at a high level of 96.3%.

In summary, the anti-false-alarm system designed in this chapter can precisely lock onto real cheating intentions while almost completely filtering out common physiological environmental noises in university computer labs through multi-dimensional cooperation across the spatial and temporal domains. Its engineering deployment value and auditing fairness are solidly backed by the experimental data.

5 CONCLUSIONS AND FUTURE OUTLOOK

5.1 Summary of the Work

Aiming at the pain points of constrained computational power and network bandwidth when traditional centralized video invigilation systems are deployed on legacy hardware in university public computer laboratories, this paper designs and implements a lightweight, multi-dimensional behavior auditing invigilation system based on decentralized edge computing. The system completely isolates image capture, posture estimation, and state decision-making within a closed loop on the localized edge client, transmitting only low-frequency structured text logs to the teacher terminal. This fundamentally eliminates the risk of local area network (LAN) paralysis caused by concurrent high-definition video streaming.

Concurrently, the system deeply exploits the features of the underlying native transformation matrix, directly executing inverse solutions for posture angles and gaze ratios via basic scalar arithmetic operations within the JavaScript runtime, which thoroughly eradicates the dependency on heavy libraries such as OpenCV.js. Furthermore, this paper proposes a dual-layer cooperative anti-false-alarm mechanism that integrates spatial directional boundaries with a temporal sliding window filter. This mechanism not only channels benign long-term gestures, such as tilting the head upward in deep thought, through directional determination but also smoothly absorbs transient noises like eye-rubbing. Experimental results demonstrate that on a low-spec standalone host, the single-frame resolution latency of the system requires only

0.62 ms, and the memory footprint is streamlined to 146.5 MB. While restricting the false positive rate (FPR) of environmental noise to 1.3%, the true positive rate (TPR) for actual cheating behaviors firmly remains at a high level of 96.3%. The proposed system possesses substantial value for inclusive deployment and practical application on legacy educational equipment.

5.2 Outlook for Future Work

Although the proposed system has achieved ideal experimental outcomes regarding lightweighting and false alarm mitigation, further in-depth research across multiple dimensions is still required to confront more complex examination room environments in the future.

First, future work will introduce edge-side adaptive image equalization algorithms and temporal correction mechanisms to enhance the feature tracking robustness of the system under extreme lighting conditions, such as strong backlighting, partial dark shadows, and high-frequency lens reflections from glasses. Second, the research will explore multimodal behavior auditing technologies that integrate the temporal dynamics of workstation keystrokes and mouse movements with page defocus and screen-switching data. By performing late weighted decision fusion with head and facial visual features, the auditing blind spots of highly covert cheating methods can be thoroughly eliminated. Finally, targeting rural or exceptionally outdated testing centers with severely deficient conditions, the team will further carry out research on INT8 integer quantization and micro-structural pruning of edge-side models. Combined with WebGL hardware acceleration technology, the spatial and computational overhead of the algorithms will be compressed to the maximum extent, thereby achieving highly inclusive compatibility and comprehensive coverage across varied educational equipment.

COMPETING INTERESTS

The authors have no relevant financial or non-financial interests to disclose.

REFERENCES

[1] Lugaresi, Camillo, Tang Jiuqiang, Nash Hadon, et al. Mediapipe: A framework for perceiving and processing reality. Third workshop on computer vision for AR/VR at IEEE computer vision and pattern recognition (CVPR), Long Beach, CA, 2019.

[2] Jakhete Sumitra A, Nilima Kulkarni. A comprehensive survey and evaluation of mediapipe face mesh for human emotion recognition. 2024 8th International Conference on Computing, Communication, Control and Automation (ICCUBEA), IEEE, 2024.

[3] Haas, Andreas, Rossberg Andreas, Schuff Derek L, et al. Bringing the web up to speed with WebAssembly. Proceedings of the 38th ACM SIGPLAN conference on programming language design and implementation, 2017.

[4] Engel Mychael. Performance Comparison of Tauri and Electron Frameworks in Multiplatform Desktop Application Development, 2026: 3875-3882.

[5] Tiangolo S. (n.d.). FastAPI: FastAPI framework, high performance, easy to learn, fast to code, ready for production. 2026. https://fastapi.tiangolo.com/.

[6] Fette Ian, Alexey Melnikov. The websocket protocol. No. rfc6455, 2011.

[7] Electron contributors. (n.d.). Introduction | Electron. 2026. https://www.electronjs.org/docs/latest.

[8] Electron Core Team. BrowserWindow | Electron, ElectronJS. 2026. https://www.electronjs.org/docs/latest/api/browser-window#winsetkioskflag.

[9] Jennings C, Bruaroey J, Boström H, et al. Media Capture and Streams. W3C Candidate Recommendation Draft. 2025. https://www.w3.org/TR/2025/CRD-mediacapture-streams-20251009/.

[10] GOOGLE LLC. MediaPipe Tasks Vision JavaScript API Document. https://ai.google.dev/edge/api/mediapipe/js/tasks-vision.

[11] Fischler Martin A, Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24.6, 1981: 381-395.

[12] Szeliski Richard. Computer vision: algorithms and applications. Springer Nature, 2022.