Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

A Multi-Modal AI Proctoring System for Remote Examinations

Shardool Patil, Sudarshan Bankar, Irfan Patel, Vallabh Patil, and P. P. Joshi
Department of Computer Engineering, Pune Institute of Computer Technology, Pune, India

***Abstract*—The large scale transition in higher education from an analog environment to digital through remote learning has created a need for effective methods to evaluate students remotely, and thus required that there be reliable solutions to support this type of evaluation. First generation on-line proctoring tools were based largely upon single modality surveillance; such as simple facial detection. However, these types of early solutions are now being exploited by sophisticated methods of academic dishonesty; such as using peripheral vision or having someone assist you while you take an assessment. Thus, we developed a new type of automated proctoring solution called *ProctorGuard*. This article presents a Multi-Modal Fusion Engine that combines in parallel geometric head pose estimation (PnP) with eye gaze at the fine grain level using L2CS-Net; object detection in real time using YOLOv8; and audio forensics. With an initial dynamic calibration of baselines followed by use of smart temporal priorities for all sensor modalities, ProctorGuard is able to distinguish between normal movement and actual attempts to cheat. In addition, the authors have developed an end-to-end architecture for software applications incorporating both evidence logging as well as examiner dashboards.**

***Index Terms*—Online Proctoring, Multi-Modal Fusion, YOLOv8, Head Pose Estimation, L2CS-Net, Audio Forensics, Academic Integrity, Deep Learning.**

I. Introduction

There is no doubt that the last five years have caused an earthquake in the world of academics. There was an explosion in the number of people looking for ways to access educational content remotely, as well as a massive move toward digitalizing all aspects of higher education. As a result of this trend, remote exams went from being second best option to first-best option for millions of students across the globe. On the other hand, the increased distance between where students take tests and where they are monitored has created a significant crisis concerning student's academic integrity.

Without the physical monitoring of a professional invigilator, students now have countless methods available to them to use advanced cheating techniques. Prior to 2015, automated proctoring systems had been developed on single modal monitoring. These systems mainly employed simple facial recognition and/or minimal movement analysis. By the time most of these systems began to become widespread, it quickly became evident that there were numerous limitations in using unimodal monitoring. As soon as technology allowed for sophisticated surveillance (using multi-modal monitoring) that could monitor multiple body parts at once, a large amount sophisticated methods for evading proctoring were developed.

For example, if a candidate can maintain eye contact direcly with their computer screen and utilize their peripheral vision to glance at cheat sheets that are positioned outside of the frame, then this specific "eyes forward, head down" method would be completely ineffective against current automated proctoring systems.

In addition, systems that rely solely on video will never be able to detect auditory cheating. Examples of auditory cheating include whispering information to someone located near your desk who cannot be seen on camera; reading test questions out loud so that you can hear what is on the test; or hearing your partner provide you with correct answers via nearly imperceptible ear pieces. Therefore, in order to eliminate these forms of cheating, today's proctoring systems must transition into utilizing Multi-Modal Behavioral Analysis.

In this paper, we detail the implementation of **ProctorGuard**, a system that operationalizes the theoretical models explored in recent academic surveys. ProctorGuard is a multi-modal AI proctoring system that goes beyond traditional eye-tracking. It utilizes a Multi-Modal Fusion Engine combining Computer Vision, Deep Learning, and Audio Forensics. The core contributions of this implementation paper are:

- PnP + L2CS-Net geometric head pose integration to improve upon fine-grained gaze estimation by eliminating "head-forward, eyes-away" vulnerabilities.
- Use of YOLOv8 for Real-Time Contraband Detection.
- Development of a dynamic calibration module which will learn a users' normal resting position so as to avoid structural false positives.
- Multi-Modal Temporal Prioritizer; aggregation of multiple modalities over time; prioritization of harder evidence against softer visual cue evidence.
- End-To-End Architecture of examiner dashboard and automatic evidence logging.

II. Background and Previous Research

Automated invigilation has been continually advancing toward finer grain in collected data. This paper's implementation is built upon many of the same key technologies as previous implementations.

A. Estimating Head Pose and Gaze

The orientation of a person's head can provide an initial indication that a student is focused. In addition to head pose,

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

estimating the direction or ”gaze” of a student also provides a fine-grain measure of their attentiveness.

In early implementations of automated invigilation, solutions used the PnP problem to identify 2D facial landmarks and then solve for 3D translation. However, the use of head pose alone is generally insufficient. Therefore, in addition to estimating head pose, it is equally important to estimate a student's gaze. More recent architectures such as L2CS-Net have shown high levels of success using a classification-regression approach to estimate 3D vectors representing a students' gaze from standard webcams. These models have shown significant ability to handle the uncertainty associated with unconstrained environments.

B. Forensic Analysis of Objects and Auditory Data

While visual attention tracking is a valuable tool in determining whether a student is attentive during exams, tracking visual attention alone does not account for all aspects of a student's behavior. As such, both audio and video forensic analysis are necessary to determine the level at which a student was paying attention.

Visual forensics uses computer vision techniques such as object detection to detect if the student was looking at something they were supposed to be focusing on, i.e., a screen displaying a test question. Object detection is accomplished through various machine learning-based approaches. One of these approaches includes the YOLO (You Only Look Once) framework, which has greatly improved the speed and efficiency of real-time object detection. A version of this model called YOLOv8 is now being utilized as an anchor-free detector for small objects including mobile phones that may be partially obscured while being held in the student's hand.

Audio forensic analysis focuses on processing the audio stream to detect verbal assistance provided by another individual. Voice activity detection (VAD), is one method of analyzing the audio stream. VAD detects when there is sufficient energy present in the signal to classify it as voice.

III. System Architecture

ProctorGuard has been developed with a modular design as well as an asynchronous pipeline. This was done to provide for the capability of ProctorGuard to operate in real time on commodity hardware. The overall system will be made up of three different sensory streams that are inputted into a fusion engine which will provide output. In addition to this there will also be an additional component that allows us to report our results externally.

A. Visual Stream: Head Orientation Estimation

The visual stream takes in video feed from a standard 30 frame per second webcam. Using the Perspective-N-Point (PnP) geometric head pose estimation algorithm, we can estimate the head's orientation (Roll, Pitch, Yaw). First, using the MediaPipe Face mesh, we have extracted 468 three-dimensional facial points. From these 468 three-dimensional facial points, we isolated six two-dimensional points: the tip of the nose; chin; left corner of each eye; and left and right corners of each mouth.

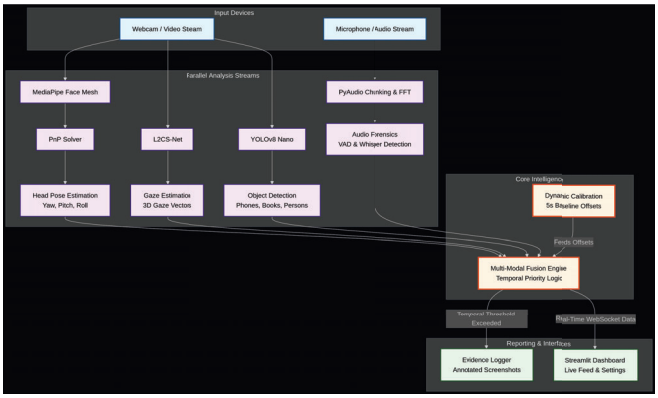

Fig. 1. ProctorGuard High-Level System Architecture, illustrating the flow of video and audio streams into the Fusion Engine.

Using a generic 3D human face model, the camera matrix K, and distortion coefficients derived from the webcam, we solve the PnP problem to find the rotation vector R and translation vector T:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R|T \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

where (u, v) are the 2D image coordinates and (X, Y, Z) are the 3D world coordinates. The rotation vector is then converted to Euler angles to extract actionable Yaw and Pitch metrics.

B. Vision Stream: Gaze Estimation (L2CS-Net)

In order to identify the ”head-forward, eyes-away” abnormality in the video stream, we use L2CS-Net (looking to listen and Catching Stares Network). In our experiment, we have utilized pre-trained weight (`l2csnet_gaze360.pkl`) that has been trained under unconstrained conditions. To predict where a person is looking based on their eyes, the L2CS-Net model extracts a small rectangular portion of the image by cropping the localized face bounding box. This cropped rectangle is then fed into a resnet-50 backbone. After processing the cropped rectangle through this backbone, the L2CS-Net model outputs two continuous values; gaze yaw (g_y) and gaze pitch (g_p), which are independent of each other. Therefore, if there is an error in detecting where the head is located within the scene (head pose), the system will still be able to independently measure the direction of where the eyes are pointing at.

C. Object Detection Stream (yolov8)

Using yolov8-nano (`yolov8n.pt`), we detect objects that are not allowed in school. Since yolov8-nano is designed to run as fast as possible on low-end cpus (which may include some student computers), it is used here to keep CPU temperature low and to avoid overheating. For now, the system is designed

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

to find three types of prohibited items (”Mobile Phones,” ”Books,” and ”Persons”) and we also include ”person” in case someone enters the room who should not be there.

As previously discussed, one major problem with using computer vision on people holding phones is that they tend to move them around quickly before putting them back in pockets. Because yolov8-nano operates at a maximum speed of 30 FPS, and because camera frames arrive so frequently, motion blur becomes a significant challenge. To address this, we implemented a temporal-confidence buffer. When a class of object is detected with high confidence in frame t, we temporarily reduce the confidence threshold in the same localized area in frames $t+1$ to $t+3$. This prevents motion-blurred images of objects from being ignored simply because the object moved too quickly.

D. Audio forensics stream

We use `audio_monitor.py`, a python module that uses PyAudio to sample audio from the default microphone in discrete chunks. Using a fast fourier transform (FFT), we perform a spectral analysis of these samples. We employ a dual-threshold method: 1. **amplitude thresholding:** this detects both loud events and normal conversation. 2. **Whisper Detection using spectral energy analysis:** whispers often lack fundamental frequency pitch but contain substantial amounts of spectral energy in high frequency ranges (e.g., hissing sounds). We filter-out frequencies below 2000 hz and above 5000 hz from all sampled audio data to identify whispering or other forms of stealthy voice communication.

IV. Core Implementation Modules

A. Dynamic Calibration Phase

Early proctoring systems were pretty rigid. They essentially assumed everyone sits perfectly straight, which just isn’t how bodies work. Different desk setups, spinal curves, and natural resting positions mean everyone’s ”normal” looks a bit different. To fix this, ProctorGuard kicks things off with a Dynamic Calibration Phase.

For the first N seconds (we usually set this to 5 seconds), we just ask the user to look right at the center of their screen. Using the Perspective-n-Point (PnP) algorithm, we map their 2D facial landmarks into 3D space to grab their actual head orientation. Then, we average out their head and gaze angles to establish a personal baseline:

$$O_{hy} = \frac{1}{N}\sum_{i=1}^{N} h_{y,i}, \quad O_{gy} = \frac{1}{N}\sum_{i=1}^{N} g_{y,i} \tag{2}$$

where O_{hy} is the Head Yaw Offset, and O_{gy} is the Gaze Yaw Offset. The `FusionEngine` class stores these offsets via the `set_calibration()` method, ensuring all subsequent analyses are calculated relative to the user’s natural posture.

B. The Fusion Engine and Priority Logic

The `fusion_engine.py` script is essentially the brain of our system. After a personal baseline has been determined using the sensors, it continuously calculates how much each sensor has deviated (delta) from a person’s individual center point:

$$\Delta h_p = \text{head\_pitch} - O_{hp} \tag{3}$$
$$\Delta h_y = \text{head\_yaw} - O_{hy} \tag{4}$$
$$\Delta g_p = \text{gaze\_pitch} - O_{gp} \tag{5}$$
$$\Delta g_y = \text{gaze\_yaw} - O_{gy} \tag{6}$$

If, as mentioned previously, multiple sensors are measuring different things; i.e., a smartphone in front of the camera with the students head directly in line with the middle of the screen and the camera. In this case, if all other factors were equal, you would likely receive many conflicting messages. A solution was created for this problem. We built a decision tree based on priority levels. There are two types of evidence: “Hard,” and “soft.” Hard evidence – Anytime YOLOv8 detects a phone or FFT determines whispering in real-time, then it will immediately trigger a violation. Soft evidence – Whenever the facial mesh detects some sort of odd head position/tilt; it is not hard evidence by itself, so it needs another level of review before it can be considered an actual violation.

C. Temporal Filtering

Temporal filters (as seen in algorithm 2) add a layer of common sense. Humans aren’t statues. They stretch, look down at our keyboards, and glance away when we’re trying to remember an answer. If the system was flagging each individual one of those micro-movements, there would be no way for students to even blink during their tests.

To make up for the lack of “human patience,” we applied temporal filters. We set a timer when a ”soft” visual anomaly occurs (such as looking to one side very intensely). A soft visual anomaly will trigger a full 1.5 second (the default value of our `TIME_THRESH`) for the student to continue in this manner for the system to consider it an actual alert. In other words, if a student simply glances at her keyboard while typing and momentarily is looking down at her keyboard, she will have triggered an alert as the instant angle exceeded the threshold established in the `FusionEngine` (the default setting for `TIME_THRESH` was set to 1.5 seconds), however since this occurred within one second or less, she would also fail the time-based requirement.

D. Evidence Logging and Dashboarding

When the system finally determines that an actual violation occurred, the violations are captured by `logger.py` module. The exact video frame that contained the violation is captured along with the timestamp and the specific violation data (for example, ”eyes looking side +32 degrees”), and then securely recorded to provide reviewers irrefutable audit trails.

We also created a real-time invigilation dashboard as part of our implementation. The dashboard was built using Streamlit

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Algorithm 1 Temporal Priority Logic Evaluation

Require: $\Delta h_y, \Delta h_p, \Delta g_y, \Delta g_p$, ObjectState, AudioState
$is\_suspicious \leftarrow$ False
$reason \leftarrow$ ""
if ObjectState == 'Phone Detected' **then**
  $is\_suspicious \leftarrow$ True
  $reason \leftarrow$ "Unauthorized Object (Phone)"
else if AudioState == 'Whisper' OR 'Speech' **then**
  $is\_suspicious \leftarrow$ True
  $reason \leftarrow$ "Auditory Violation"
else if $|\Delta h_y| >$ HEAD_YAW_THRESH$(30°)$ **then**
  $is\_suspicious \leftarrow$ True
  $reason \leftarrow$ "Head Turned Side"
else if $|\Delta h_p| >$ HEAD_PITCH_THRESH$(35°)$ **then**
  $is\_suspicious \leftarrow$ True
  $reason \leftarrow$ "Head Tilted Up/Down"
else if $|\Delta g_y| >$ GAZE_THRESH$(30°)$ **then**
  $is\_suspicious \leftarrow$ True
  $reason \leftarrow$ "Eyes Looking Side"
end if
if $is\_suspicious$ **then**
  Update temporal buffer
  if Duration $>$ TIME_THRESH$(1.5s)$ **then**
    return True, $reason$
  end if
else
  Reset temporal buffer
end if
return False, ""

Fig. 2. Pseudocode for the Temporal Priority Logic Evaluation

(`app.py`). We were able to tie in this application to allow proctors to view the stream in real time while viewing overlaid bounding box information along with 3D gaze vector overlays. This represents the best of both worlds; scalable AI processing combined with the use of human judgment.

V. Experimental Setup and Evaluation

To assess whether ProctorGuard works as intended, we created a testing environment that mimics a typical test-taking experience using common equipment found in homes or offices; this represents how students would likely be tested remotely. We used off the shelf hardware so that our results can represent a real-world use case.

A. Testing Environment

It wouldn't mean much if this only worked on a $10,000 supercomputer. We needed to know ProctorGuard could run smoothly on the kind of hardware an average student actually uses at home. So, we tested it on a mid-tier workstation: an Intel Core i7 processor, 16 GB of DDR4 RAM, and a standard 720p built-in webcam. We also specifically skipped using a dedicated GPU for the deep learning models to make sure the CPU could handle the load entirely on its own.

B. Performance Metrics

We tested the effectiveness of ProctorGuard by simulating student behavior based upon a set of predetermined parameters and categorized those behaviors into two groups: "benign" behaviors (i.e., a student shifting position in their chair, taking time to think about answers, etc.) and "malicious" behaviors (e.g., a student using his/her cell phone, a student whispering to another person, etc.). In machine learning, you can't just look at basic "accuracy," especially since actual cheating is relatively rare. We leaned heavily on the F1-Score to see how well we balanced finding the real cheaters (Recall) without falsely accusing honest, fidgety students (Precision).

TABLE I
Detection Accuracy by Modality and Fusion

Configuration	Precision	Recall	F1-Score
Pose-Only Baseline	0.72	0.65	0.68
Pose + Gaze	0.81	0.78	0.79
YOLOv8 + Audio Only	0.88	0.70	0.78
ProctorGuard Fusion	**0.92**	**0.89**	**0.90**

Using just head pose would not have been sufficient as students could just freeze their necks and move their eyes to cheat, leaving us with a terrible F1-score of 0.68. But when we fed everything into the Fusion Engine (pose, gaze, YOLOv8 object detection, and audio analysis), our F1-score shot up to 0.90. We hit 92 percent precision specifically because the temporal filtering successfully weeded out the innocent movements, while keeping recall high. The improvement in performance metrics such as precision and recall shown in the Table I shows that combining all three modalities significantly improves the performance of the overall ProctorGuard System.

C. Computational Overhead

Running four heavy deep learning networks concurrently is tough on a standard CPU. To keep laptops from overheating and crashing mid-exam, we had to get creative with our optimization. We used the "nano" version of YOLOv8 and only ran the object detection every fifth frame. A phone doesn't magically materialize in a millisecond, so we didn't lose any real security by doing this. This trick kept CPU usage hovering safely between 45 percent and 55 percent. The whole pipeline takes about 65 milliseconds from start to finish, giving us a smooth perceived rate of over 15 frames per second—more than enough detail to catch any funny business without melting the student's computer.

VI. Challenges and Limitations

ProctorGuard represents an important technological development; nevertheless, a number of both technical and environmental limitations continue to challenge the deployment of Edge-Based Proctoring Systems:

- **Low-Light Environments:** Both MediaPipe Face Mesh and L2CS-Net experience significant degradation in performance under low-light conditions or severe backlit

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

conditions. As well, when light source shadows a subject's face, their eye-gaze vectors tend to become unpredictable and may lead to false positive determinations.

- **VFOA Ambiguity:** Vision Based False On Action (VFOA) ambiguities have yet to be fully solved. While the technology will be able to accurately determine within approximately 30 degrees the direction of the focus of attention that a student is visually fixating upon to the right; because this system can use no camera(s) other than those provided by the second computer running the program there is simply no way of knowing for sure whether a student is looking at a stickie-note tacked up on the side of their desk as they consider an answer, or some form of unauthorized sticky-note.
- **Adversarial Virtual Cameras:** A technically savvy student may record their own video recordings using OBS studio virtual camera software, etc. Even though our system works very well with local inference on the video feed being recorded, our current version of the application does NOT include live cryptographic liveness detection to ensure that the source of the video stream is coming from an actual peripheral device and not simply from a file on a computer.

VII. Future Scope

ProctorGuard's future versions will implement a cloud-hybrid model to provide increased security compared to ProctorGuard Edge Processing.

- **Liveness Verification:** Developing challenge-response methods to verify whether a student is actively present at the computer by requesting that the student follow a randomly moving point displayed on the screen, etc. to prevent virtual injection of camera images.
- **Multi-Camera Support:** Use a smart phone as an additional Hand Camera connected through the internet to view the desk and eliminate the VFOA ambiguity and capture misuse of devices outside of the video frame.
- **Transformer-Based Temporal Models:** Replace the priority tree logic with a spatio-temporal graph convolutional network (ST-GCN), or vision transformer (ViT) to recognize and learn sequential patterns of behavior occurring over time.

VIII. Conclusion

The security and integrity of online examinations can no longer rely on simple, uni-modal tracking methodologies. Online testing has to move beyond unimodal tracking techniques; tracking is one way to detect cheating, but it will have to become more sophisticated as students continue to cheat using increasingly advanced tactics.

In this paper we introduced ProctorGuard which uses a multi-modal fusion engine to create a robust behavioral representation of the test-taker. By incorporating geometric head pose estimation, l2cs-net based eye-tracking, yolo v8 object detection and spectrogram audio forensic analysis, ProctorGuard creates a high fidelity behavioral representation of each test taker. We show how our dynamic calibration phase, combined with our temporal priority logic engine results in lower false positive rates than most existing solutions. The new framework focuses on empirical evidence; runs on commodity hardware; and presents a more accurate, more detailed, and fairer method of monitoring candidates in today's academic environment.

Acknowledgment

The authors would like to sincerely thank the Department of Computer Engineering at Pune Institute of Computer Technology (PICT) for providing the computational resources and academic guidance necessary to complete this project. Special thanks to the open-source communities maintaining the MediaPipe, YOLO, and L2CS-Net repositories.

References

[1] S. Patil, S. Bankar, I. Patel, V. Patil, and P. P. Joshi, "Multi-Modal Online Proctoring: A Survey on Fusion of Head Pose, Gaze, Object Detection, and Audio Forensics," *Unpublished Survey Paper*, Pune Institute of Computer Technology, 2026.
[2] The Element of Choice: Online Students' Perceptions of Online Exam Proctoring - OJDLA (2026).
[3] Digital proctoring in higher education: a systematic literature review - Emerald Insight (2026).
[4] Deep Learning-Based Multimodal Cheating Detection in Online Examinations (2026).
[5] AutoOEP - A Multi-modal Framework for Online Exam Proctoring - arXiv (2025).
[6] How Emerging Tech Is Disrupting the Cheating Industry Through Smarter Proctoring (2025).
[7] A. Researcher et al., "Enhancing online proctoring efficiency: Utilizing AI," ResearchGate, 2026.
[8] Audio Flagging in Online Proctoring: Enhancing Exam Security - ThinkExam (2026).
[9] Virtual camera detection: Catching video injection attacks - arXiv (2025).
[10] Next-Gen Liveness Detection for Deepfake and Injection Attacks - ROC (2025).
[11] A. Farkhana et al., "L2CS-Net: Fine-Grained Gaze Estimation in Unconstrained Environments," ResearchGate, 2026.
[12] Ultralytics YOLO Evolution: An Overview of YOLO26, YOLO11, YOLOv8 - arXiv (2025).
[13] YOLO11 vs. YOLOv8: The Evolution of Real-Time Object Detection - Ultralytics (2024).
[14] Forensic Voice Comparison: The Essential Guide — PHONEXIA (2026).
[15] ProctorEdge: Advanced AI Examination Monitoring and Security System (2025).