WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

AI-Based Multi-Student Monitoring and Suspicious Activity Detection in Examination Halls Using Holistic Detection

Manojkumar M, Mrs. Gayathri N*

Student, Department of Artificial Intelligence and Machine Learning, Sri Shakthi Institution of Engineering and Technology
*Project Guide, Department of Artificial Intelligence and Machine Learning, Sri Shakthi Institution of Engineering and Technology

ABSTRACT

Monitoring a large examination hall is not easy. An invigilator has to keep an eye on many students at the same time, and some events may go unnoticed. This paper describes an AI-based system developed to support that work. It brings together student and object detection, posture and head-pose analysis, gaze tracking, optional audio anomaly monitoring, risk scoring, evidence storage, and a web dashboard.

The system is implemented in Python using OpenCV, YOLOv8, MediaPipe-related processing, SQLite, and Flask. YOLOv8 detects people and selected objects in the camera view. Other modules examine posture, head direction, and gaze-related patterns. When audio monitoring is enabled, microphone input is also checked for unusual sound-level patterns. A temporal risk engine combines events over time, reduces scores as older evidence becomes less relevant, and applies a cooldown to limit repeated alerts. Event details and available evidence can be stored and reviewed through the dashboard and reports.

Development notes indicate that the live camera, dashboard, audio alerts, database initialization, and PDF reporting have been tested in the project environment. A previous software test run reported 275 passing tests. This is a software-testing result, not a measurement of detection accuracy. The system is intended to assist invigilators; an alert or risk score should always be checked by a person before any conclusion is made.

Keywords: Holistic detection; examination monitoring; YOLOv8; MediaPipe; gaze tracking; head-pose estimation; audio anomaly monitoring; temporal risk scoring; Flask; SQLite.

I. Introduction

Maintaining fairness during examinations is an important responsibility for educational institutions. In a large hall, however, one invigilator may not be able to observe every student continuously. Reviewing long recordings after an incident can also take considerable time.

This project, titled “AI-Based Multi-Student Monitoring and Suspicious Activity Detection in Examination Halls Using Holistic Detection,” explores how computer vision and optional audio analysis can support this task. Instead of relying on one classifier, it combines object detection with posture, head-pose, and gaze-related cues. A temporal scoring component helps organize events over time, while the dashboard and database make incidents easier to review later. The system is designed as a support tool for invigilators, not as a replacement for their judgment.

II. Problem Statement

During an examination, it can be difficult for a supervisor to notice every unusual movement or prohibited object, particularly when many candidates are seated in the same room. A basic system that classifies a single frame as normal or suspicious may miss context, repeat alerts for the same event, or fail to keep a useful record of what happened.

The project addresses this by combining several types of information, associating detected objects with student regions where possible, tracking event scores over time, and saving incidents for review. There are still important limitations: lighting, camera angle, crowding, ordinary student movements, and background noise can all affect the output. For this reason, the system highlights events that may need attention rather than treating them as confirmed misconduct.

III. Objectives and Motivation

The aim is to build a practical monitoring prototype that can help an invigilator review potential incidents in a multi-student camera view.

The main objectives are to:

Detect students and selected objects in camera frames using YOLOv8.

Analyze posture and head orientation and track gaze-related cues over time.

Check audio anomalies when the microphone-monitoring component is enabled.

Combine events using temporal risk scoring, score decay, and cooldown handling.

Save incident details and relevant evidence in SQLite.

Display the live stream, event information, and risk values in a Flask dashboard.

Make stored events available for reporting and later review.

Keep a human reviewer involved in decisions based on system alerts.

IV. Existing System and Proposed System

The reference paper supplied for this work describes a CNN-based approach. It processes camera frames, classifies activity as normal or suspicious, captures an image when an event is detected, and sends an email alert. The paper also notes possible weaknesses such as limited gesture coverage, false detections, sensitivity to lighting, and dependence on camera and computer performance.

The present project uses a broader pipeline. Its implementation notes describe YOLOv8 for person and object detection, OpenCV-based posture and head-pose analysis, MediaPipe Holistic processing, gaze smoothing and filtering, optional microphone-based audio anomaly checks, a temporal risk engine, incident evidence management, SQLite storage, and a Flask dashboard with live video and reporting functions.

Although these components are implemented in the project, the available information does not include a verified benchmark dataset, a ground-truth evaluation, or measured precision, recall, F1-score, or detection accuracy. Those results should not be assumed.

V. Methodology

The monitoring process follows a series of stages:

1. Video capture. OpenCV reads frames from the selected camera. A demonstration mode is also available to test the workflow without a live camera.

2. Student and object detection. YOLOv8 detects people and selected objects. The project configuration identifies class IDs for person (0), cell phone (67), laptop (63), and book (73). Where possible, object locations are compared with student regions to provide context.

3. Behavior analysis. The behavior modules examine posture-related events, including leaning or standing, and estimate head orientation. The project notes describe 3D head-pose estimation using OpenCV's solvePnP method.

4. Gaze tracking. Gaze-related estimates are smoothed and filtered over time so that a brief fluctuation is less likely to create repeated events.

5. Audio monitoring. If enabled, the audio module reads microphone samples and checks signal-level or anomaly patterns. Background noise and microphone placement can affect these results.

6. Temporal risk scoring. The suspicious-event engine combines event signals into a risk percentage from 0 to 100. Older evidence can gradually lose influence, and cooldown handling helps prevent repeated alerts for the same continuing event.

7. Evidence and storage. When configured conditions are met, event details and timestamps are recorded, and relevant evidence can be saved. SQLite provides local incident storage.

8. Dashboard and reports. The Flask dashboard provides a live MJPEG video stream and monitoring endpoints. Project notes also describe controls, CSV audit export, HTML reports, and PDF reporting.

VI. Hardware and Software Requirements

Software used or identified in the project notes:

Programming language: Python 3.12 environment

Computer vision: OpenCV

Object detection: Ultralytics YOLOv8

Landmark processing: MediaPipe

Dashboard: Flask

Database: SQLite

Audio input: sounddevice

Data and reporting utilities: NumPy, pandas, and HTML/PDF reporting utilities

Development environment: Windows with VS Code or Cursor

Hardware:

A computer capable of running Python, OpenCV, and the detection model

A webcam or camera for live video

A microphone when audio monitoring is required

Sufficient memory, processing power, and storage; optional GPU resources may help depending on the model and desired frame rate

The project notes do not establish a verified minimum hardware specification or measured frame-rate target. These should be reported only after testing the final system on the intended machine.

VII. System Architecture and Flow

The system begins with a live camera stream. Each frame is processed by the detection and behavior modules. Audio information, when enabled, is analyzed separately. The temporal risk engine combines available event cues and applies filtering rules. Relevant incidents are then stored with their available evidence and displayed for review through the dashboard and report functions.

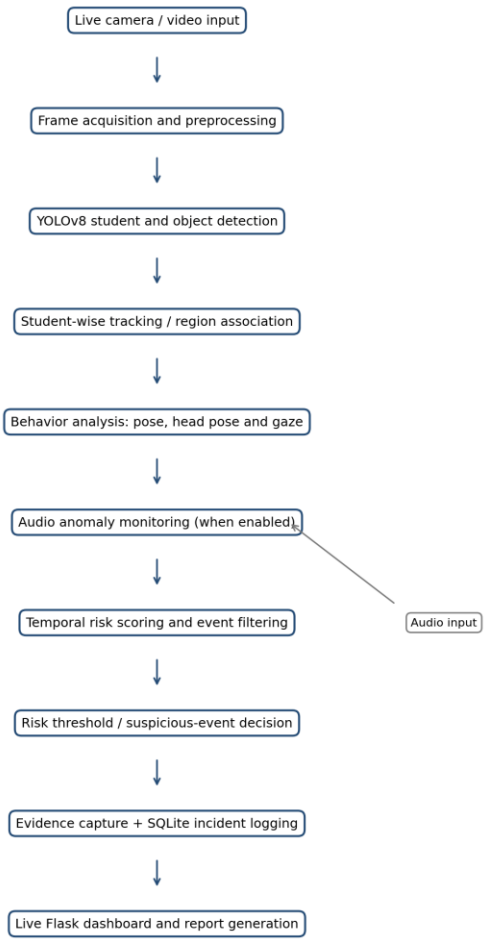


Fig. 1. High-level flow of the implemented monitoring pipeline.

VIII. Module Description and Algorithm

A. Camera module. Opens the configured camera and supplies frames to the monitoring loop. A demo source supports repeatable workflow checks.

B. Detection module. Uses YOLOv8 to detect people and selected objects. Spatial association helps connect object detections with student regions when the geometry supports it.

C. Holistic and behavior modules. Use landmark and image-processing information to examine posture and head orientation. These signals may indicate an event for review, but they do not reveal a student's intent.

D. Gaze tracker. Smooths gaze-direction estimates and filters short-lived or repeated events.

E. Audio engine. When enabled, samples microphone input and evaluates signal-level and anomaly conditions. Its performance depends on the microphone, background noise, and calibration.

F. Suspicious-event engine. Maintains risk values over time, combines event contributions, reduces the influence of older events, and applies cooldown handling.

G. Evidence manager and database. Saves incident metadata and evidence references. SQLite is used for local persistence.

H. Dashboard and report generator. Flask provides live video and monitoring endpoints. Stored incident information can be reviewed and exported through the available reporting workflow.

Algorithm:

1. Load configuration, database, detection model, camera, behavior and gaze modules, optional audio monitoring, and dashboard services.
2. Read a frame from the camera.
3. Detect students and selected objects.
4. Associate objects with student regions where possible.
5. Estimate posture, head pose, and gaze-related cues.
6. Process audio cues if audio monitoring is enabled.
7. Update temporal risk values and apply decay and cooldown rules.
8. Record qualifying events and capture evidence when configured.
9. Update dashboard information and make stored events available for reports.
10. Repeat until monitoring stops or the input source fails.

IX. Implementation and Validation

The project notes describe a working Flask dashboard, SQLite initialization, live camera capture, YOLOv8 model loading, MediaPipe-based processing, an active monitoring loop, audio alerts, and PDF reporting. A previous automated test run reported 275 passing tests with two warnings. This indicates that the test suite passed in that development run; it does not establish how accurately the system detects suspicious behavior. The tests should be rerun against the final submitted version.

No verified test-set size, confusion matrix, precision, recall, F1-score, latency, or multi-camera throughput measurements were provided. A formal evaluation should use labeled and appropriately consented recordings, separate training and testing by session to reduce data leakage, and report precision, recall, F1-score, false alerts per hour, processing latency, and resource usage across different lighting and seating conditions.

X. Advantages

Helps monitor several people within the camera view.

Combines object and behavior-related cues.

Uses temporal scoring to reduce reliance on isolated detections.

Stores incidents and related evidence for later review.

Provides a dashboard and report-generation workflow.

Supports optional audio anomaly monitoring.

Includes a demo mode for repeatable workflow testing.

XI. Limitations and Challenges

Camera angle, image quality, lighting, occlusion, and crowding affect detection.

Ordinary movements or looking away may trigger false alerts.

Head pose and gaze are indirect signals and do not prove cheating.

Audio results can be affected by background noise and microphone calibration.

Object detection and spatial association cannot guarantee that every prohibited item is found or linked to the correct student.

Processing speed depends on the computer, camera frame rate, and model configuration.

Production deployment, multi-site scaling, and external authentication need separate validation.

Video, audio, and behavior information can be sensitive; appropriate notice, access controls, retention rules, and institutional approval are important.

XII. Future Scope

Future work can include testing on a documented and consented examination-hall dataset, tuning risk weights against human-reviewed examples, and publishing measured precision, recall, F1-score, and false-alert rates. Other improvements could include optimized processing for multiple cameras, role-based access and audit controls, secure evidence retention and deletion, configurable institutional rules, and integration with a modern frontend. Any face-based identity verification should be assessed for reliability and bias and should follow applicable law and institutional policy. Human review should remain part of any consequential decision.

XIII. Conclusion

This project brings together YOLOv8 detection, MediaPipe-related processing, posture and head-pose analysis, gaze tracking, optional audio anomaly monitoring, temporal risk scoring, SQLite incident storage, evidence capture, and a Flask dashboard. Together, these components are intended to help invigilators identify events that deserve closer review and keep incident information organized. The available development notes report that key monitoring and reporting features are working and that a previous test run passed 275 tests. However, verified benchmark results were not available, so no numerical detection-accuracy claim is made here. Further controlled evaluation, privacy safeguards, and human oversight are needed before using the system for consequential examination decisions.

Acknowledgment

The guide's name and designation have not yet been supplied, so a placeholder is retained. The final paper can acknowledge the project guide, department, institution, and contributors after their details and contributions are confirmed.

References

The following entries were transcribed from the bibliography in the supplied reference paper and are included as related-work leads. The original publications should be checked to confirm the bibliographic details before formal submission. The first entry's author spelling is unclear in the supplied PDF.

[1] S. M. Rezaee, M. R. Khosravi, and M. K. Moghimi, "A survey on deep learning-based real-time crowd anomaly detection for secure distributed video surveillance," Personal and Ubiquitous Computing, vol. 28, no. 1, pp. 135–151, Feb. 2024. Author spelling requires verification against the original source.

[2] M. Perez, A. C. Kot, and A. Rocha, "Detection of real-world fights in surveillance videos," in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), May 2019, pp. 2662–2666.

[3] C. V. Amrutha, C. Jyotsna, and J. Amudha, "Deep learning approach for suspicious activity detection from surveillance video," in Proceedings of the 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Mar. 2020, pp. 335–339.

[4] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2018, pp. 6479–6488.

[5] J. Wei, J. Zhao, Y. Zhao, and Z. Zhao, “Unsupervised anomaly detection for traffic surveillance based on background modeling,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Jun. 2018, pp. 129–136.