

Exam Malpractice Detector: A Real-Time, Rule-Based Computer Vision System with Seat-Level Risk Heatmaps and Automated Evidence Reporting

Dhakchitha C, Kaviyashree S, Rithika K, Subashini R

Department of Artificial Intelligence and Data Science Sengunthar Engineering College (Autonomous), Tiruchengode, Tamil Nadu, India

Abstract — Maintaining integrity in invigilated examinations is labor-intensive, and human proctors cannot continuously observe every candidate. This paper presents Exam Malpractice Detector, a lightweight, real-time monitoring system built with OpenCV and Python that flags suspicious behavior from a single camera feed. The system combines four inexpensive visual cues: multiple-face detection using a Haar cascade, frame-differencing-based motion analysis, face-position-based "looking away" estimation, and Canny-edge contour analysis for unexpected objects. Detections are mapped to a three-tier status (Normal, Doubt/Review, Confirmed Malpractice) with confidence values. Each alert triggers cooldown-limited, hash-deduplicated, JPEG-compressed evidence capture. Alerts are also projected onto a 5×7 seat grid, where weighted scores produce a color-coded hall heatmap. At session end, a structured PDF report containing session metadata, violation summary, heatmap, event table, and screenshots is generated automatically for staff review. The design keeps a human in the loop: the system flags and documents, and staff decide. Because it needs no GPU and no training data, it can run on commodity hardware.

Keywords— exam proctoring, malpractice detection, OpenCV, Haar cascade, motion detection, heatmap, automated reporting.

I. INTRODUCTION

Examinations remain a primary mechanism for assessing learning, and their credibility depends on effective invigilation. In large halls, an invigilator can attend to only a limited number of candidates at once, and post-incident disputes are often difficult to resolve without objective records. Camera-based monitoring can supplement human supervision by continuously observing the scene and preserving evidence, but many published proctoring systems rely on deep networks, specialized hardware, or cloud services that are costly or raise privacy concerns for smaller institutions.

This work describes a compact alternative. Exam Malpractice Detector uses classical computer vision to analyze a webcam stream once per second and applies transparent rules to classify each sample. Instead of declaring guilt, it grades each observation as Normal, Doubt/ Review, or Confirmed Malpractice, with the latter reserved for the least ambiguous cue (more than one face in view). A student who is simply not visible is explicitly treated as normal, to avoid penalizing camera-framing issues.

The main contributions are: (1) a multi-cue detection pipeline using only OpenCV primitives; (2) an evidence manager combining cooldown, perceptual-style hashing, and compression to keep the evidence set small and non-redundant;

(3) a seat-level scoring and heatmap scheme that highlights where in the hall suspicious activity concentrates; and (4) automated generation of a text log and a PDF report for staff. The remainder of this paper is organized as follows. Section II reviews related work, Section III gives the system overview, Section IV details the methodology, Section V covers implementation, Section VI presents outputs, discussion, and an evaluation protocol, Section VII discusses limitations and ethics, and Section VIII concludes.

II. RELATED WORK

Face detection with boosted cascades of Haar-like features [1] remains popular for real-time applications owing to its speed on CPUs, and is available through OpenCV [2]. Deep detectors such as MTCNN [3], YOLO [4], and residual networks [7] offer higher accuracy and object-level recognition but require greater computation. Edge-based analysis using the Canny operator [5], a standard tool in classical image processing [10], provides a cheap way to extract structural contours without learning, and gradient-based descriptors such as HOG [8] and facial-behavior toolkits such as OpenFace [9] address pose and gaze at higher cost.

Automated online proctoring has been studied extensively; for instance, Atoum et al. [6] combined multiple sensing streams, including head pose, gaze, and audio, to detect cheating in

remote exams. Such systems are powerful but heavier and designed around a single examinee at a personal computer. In contrast, the present system targets a lower-cost, in-hall scenario, favoring interpretability, low latency, and staff-oriented reporting over maximal accuracy.

Beyond online settings, in-hall monitoring has attracted interest because it avoids per-student software installation and can observe several candidates with one camera. The principal difficulties are varying illumination, seat-dependent viewing angles, partial occlusion by neighbors, and the need to keep false alarms low enough that staff trust the output. Many commercial products address these with cloud inference; however, institutions with limited bandwidth or strict data-residency requirements often prefer on-premise processing.

Classical motion analysis by frame differencing and background modeling remains a staple of surveillance pipelines [10] owing to its simplicity and determinism. Perceptual and average hashing, combined with a cryptographic digest [11], are widely used to detect repeated images cheaply. Our system applies these ideas in a combined pipeline specifically tailored to examination reporting: the output is not a raw alert stream but a curated, human-readable document that links each flag to time-stamped evidence and a physical seat region.

III. SYSTEM OVERVIEW

1. Problem Statement and Objectives

Manual invigilation is subjective, fatiguing, and rarely leaves verifiable records. The objectives of this work are: (i) to detect a small set of well-defined, observable malpractice cues in real time on commodity hardware; (ii) to grade each observation by certainty so that staff can prioritize review; (iii) to store limited, non-redundant evidence; (iv) to localize suspicious activity to seats; and (v) to produce a complete, shareable report without manual effort.

2. Architecture

Fig. 1 shows the architecture. A webcam stream is read frame by frame, but heavy analysis runs at a configurable interval (1 s), which bounds CPU load. Each sampled frame is converted to grayscale and passed through four independent detectors. A rule-based decision engine merges their outputs into a single status. Non-normal outcomes feed three sinks: the evidence capture module, the seat-level risk accumulator, and the session text log. When the operator presses Q or ESC, the system finalizes the log, renders the heatmap, and compiles the PDF report.

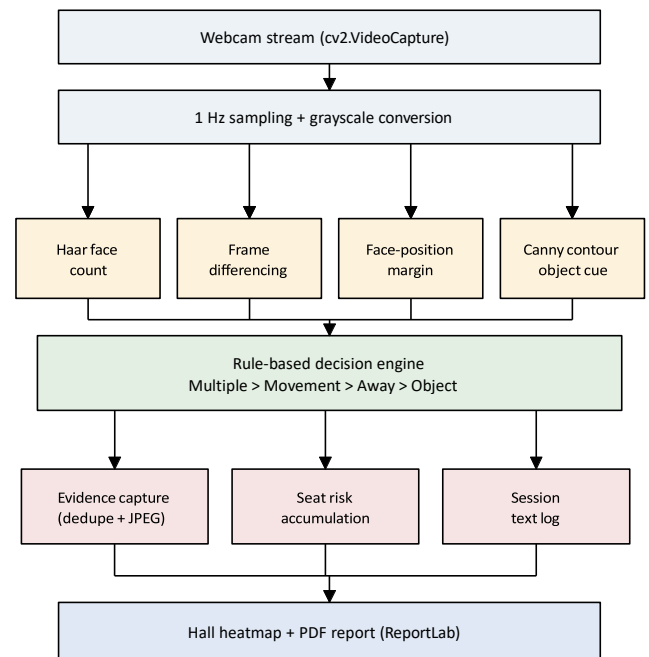


Fig. 1. Architecture of the Exam Malpractice Detector.

IV. METHODOLOGY

1. Multiple-Student Detection

Faces are located with OpenCV's pre-trained frontal-face Haar cascade (scaleFactor = 1.3, minNeighbors = 5). If more than one face is found in the frame, the sample is labeled Confirmed Malpractice with 95% confidence, and each face rectangle is recorded as a suspicious region. This is the only cue given the Confirmed label, because the presence of a second person is the least ambiguous of the four signals.

2. Suspicious Movement

The current grayscale frame is compared with the previous sampled frame by absolute difference. The difference image is binarized at intensity threshold 25, and the sum of the binary image is compared with MOVEMENT_THRESHOLD (50,000). Exceeding the threshold, when no higher-priority cue has fired, yields Doubt / Review with 60% confidence. Since the motion cue is not localized, the central third of the frame is used as its region.

3. Looking Away

For each detected face, the horizontal center is computed. If it lies within 150 pixels of the left or right image border, the system assumes the candidate has turned or leaned toward a neighbor, and assigns Doubt / Review (55%). This is a positional proxy rather than true gaze or head-pose estimation.

4. Object Detection

Canny edges (thresholds 50 and 150) are extracted and external contours are found. Contours whose area lies between 5,000 and 60,000 pixels and which do not overlap any detected face are treated as potential unauthorized objects (e.g., a phone or paper). The first such contour is highlighted and the sample is labeled Doubt / Review (50%).

5. Decision Logic

Cues are evaluated in a fixed priority order, and a lower-priority cue never overrides an earlier label. Table I summarizes the modules.

Table 1. Detection Modules and Confidence Assignment

Cue	Method / trigger	Label	Conf.
Multiple students	Haar cascade; more than 1 face	Confirmed	95%
Suspicious movement	Frame diff (thr. 25); sum > 50,000	Review	60%
Looking away	Face center within 150 px of border	Review	55%
Object detected	Canny contour, area 5,000–60,000, no face overlap	Review	50%

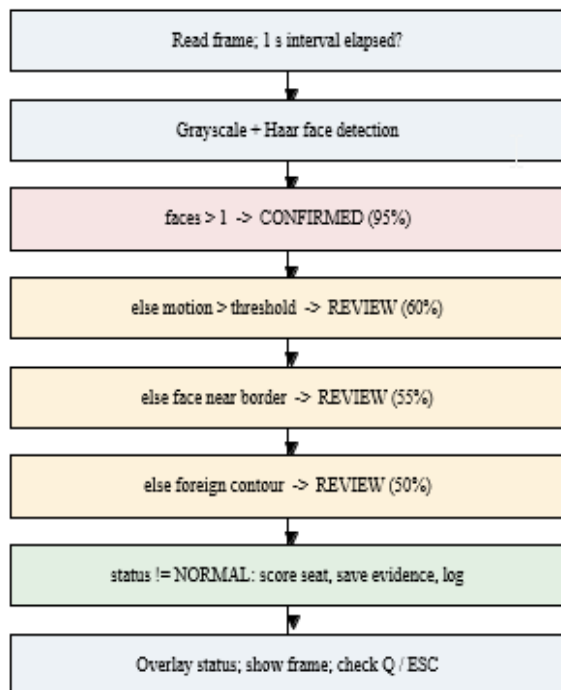


Fig. 3. Per-frame decision flow of the monitoring loop.

6. Evidence Management

To avoid flooding storage with near-identical frames, evidence is saved only if (i) at least CAPTURE_COOLDOWN (5 s) has elapsed since the last capture and (ii) the frame is not a duplicate. Duplicate checking resizes the frame to 32×32, converts it to grayscale, applies histogram equalization, binarizes it against its mean, and takes an MD5 digest of the resulting bit-matrix; digests already stored are rejected. Accepted frames are written losslessly as PNG and also re-encoded as optimized JPEG (quality 55) for compact inclusion in reports. Every event is appended to the text log with timestamp, status, reason, confidence, and evidence path.

7. Seat Mapping and Risk Heatmap

The camera frame is divided into a 5×7 grid (35 seats). The center of every suspicious region is mapped to a seat index in row-major order (Fig. 2):

$$\text{seat} = \text{row} \times 7 + \text{col}, \text{col} = \text{int}(7x/W), \text{row} = \text{int}(5y/H)$$

S01	S02	S03	S04	S05	S06	S07
S08	S09	S10	S11	S12	S13	S14
S15	S16	S17	S18	S19	S20	S21
S22	S23	S24	S25	S26	S27	S28
S29	S30	S31	S32	S33	S34	S35

Fig. 2. Frame-to-seat mapping (S12 highlighted as an example cell).

Each event adds a cue-specific weight to the seat score and increments its event counter. With $w_k = 5, 4, 3, 2$ for multiple students, object, looking away, and movement (1 for any other reason), the accumulated score of seat i is

$$S_i = \text{sum over } k \text{ of } (w_k \times n_{i,k})$$

where $n_{i,k}$ is the number of events of type k at seat i . Scores are bucketed into risk levels (Table II). The heatmap renderer draws a hall layout with desks, a glowing color per seat, seat labels and event counts, a risk-summary panel, and the top-four risk seats.

Table 2. Seat Risk Levels

Seat score	Risk level	Color
0	Low	Green
1–3	Medium	Yellow
4–6	High	Orange
7 or more	Critical	Red

8. Automated Report Generation

At shutdown, the system produces report.txt and a PDF (via ReportLab) containing: session ID and start/end times; total evidence count; a violation-count table; the hall heatmap; a table of time-stamped events; and the compressed evidence screenshots with captions. Each session is stored in its own timestamped folder with separate screenshot, compressed-screenshot, and report subfolders, preserving an auditable trail.

Table 4. Structure of the Generated Pdf Report

Report section	Content
Session summary	Session ID, exam start/end, evidence count
Violation summary	Count of each cue (4 categories)
Heatmap	Seat-wise risk visualization (JPEG)
Event table	Time, status, reason, confidence, file
Evidence gallery	Compressed screenshot per saved event

The text log is flushed after every event, so a partial record survives even if the program is interrupted. Because report generation runs after the capture loop ends, it does not compete with real-time analysis for processing time.

9. Mathematical Details of the Cues

Haar cascade. Haar-like features are evaluated in constant time using the integral image [1]:

$$ii(x, y) = \text{sum of } i(x', y') \text{ over all } x' \leq x, y' \leq y$$

A feature value is the difference between pixel sums of adjacent dark and light rectangles, and a window is accepted as a face only if it passes every stage of the boosted cascade. Increasing scaleFactor (1.3 here) reduces the number of scales searched and thus the cost, at some loss in sensitivity to faces of intermediate size; minNeighbors = 5 suppresses isolated detections.

Frame differencing. Let gt denote the grayscale frame at sample t . The binary change map and the motion score are

$$Bt(x, y) = 1 \text{ if } |gt(x, y) - gt-1(x, y)| > 25, \text{ else } 0 \quad Mt = 255 \times (\text{number of pixels with } Bt = 1)$$

because the thresholded image stores the value 255 for

Frame hash. After resizing to 32×32 , grayscale conversion, and histogram equalization, each pixel is reduced to one bit, $b_{ij} = 1$ if $g_{ij} > \text{mean}(g)$, else 0. The 1024-bit matrix is hashed with MD5 [11]; identical digests mark duplicates. Because

the digest is exact, two frames are treated as duplicates only when their bit patterns match completely.

10. Computational Complexity

Let $W \times H$ be the frame size, E the number of recorded events, and $S = 35$ the number of seats. Per analyzed sample, grayscale conversion, frame differencing, Canny edge detection, and contour extraction are each $O(WH)$. The Haar cascade scans a multi-scale image pyramid and typically dominates the cost; early cascade rejection keeps the average cost low. Seat updates are $O(|R|)$ for $|R|$ suspicious regions. Heatmap rendering is $O(S)$ and runs once at shutdown, and PDF assembly is $O(E)$. Since analysis is limited to one sample per PROCESS_INTERVAL , the average load scales with $1/\text{PROCESS_INTERVAL}$, which gives operators a simple knob to trade latency for CPU usage.

V. IMPLEMENTATION

The system is implemented in Python using OpenCV for capture and image analysis, NumPy for score arrays, Pillow for JPEG compression, hashlib for deduplication, and ReportLab for PDF output. Operation requires only a webcam and a CPU. Key parameters are given in Table III; they are exposed as constants so invigilators can tune them to hall size, camera placement, and lighting.

Table 3. Principal Configuration Parameters

Parameter	Value	Purpose
PROCESS_INTERVAL	1.0 s	Analysis period
CAPTURE_COOLDOWN	5 s	Min. gap between evidence
MOVEMENT_THRESHOLD	50,000	Motion trigger
LOOKING_AWAY_MARGIN	150 px	Border band for gaze proxy
JPEG_QUALITY	55	Evidence compression
ROWS $\times$ COLS	5×7 (35)	Seat grid

Table 5. Software and Hardware Requirements

Component	Requirement / library
Hardware	Webcam; CPU-only PC (no GPU needed)
Language	Python 3
Vision	OpenCV (cv2), Haar cascade XML
Numerics	NumPy (seat score arrays)
Imaging	Pillow (JPEG compression)
Reporting	ReportLab (Platypus) for PDF
Integrity	hashlib (MD5 frame digest) [11]

The runtime is a single synchronous loop: capture, periodic analysis, drawing, and key polling. On non-analysis frames

when

$$x \ y \ w \ h$$

$$x < fx + fw, x + w > fx, y < fy + fh, y + h > fy$$

only the capture and display steps run, keeping the preview smooth. Output folders are created per session using the start and such contours are excluded so that the face itself is not reported as an object.

timestamp, so repeated runs never overwrite earlier evidence.

Session Directory Structure

Each run creates the folder hierarchy shown in Listing 1. Original PNG frames and their compressed JPEG counterparts are kept separately, so the full-quality image remains available if a dispute arises while the report stays small.

```
exam_sessions/
  session_YYYYMMDD_HHMMSS/
    screenshots/      (original PNG)
    compressed_screenshots/ (JPEG, q = 55)
    reports/
      report.txt
      malpractice_heatmap.jpg
      Exam Malpractice Report.pdf
```

Listing 1. Per-session output folder layout.

Threshold Calibration

All thresholds are expressed in raw pixel units, so their meaning depends on camera resolution. For a 640×480 frame (307,200 pixels), the motion threshold corresponds to a changed-area fraction of

$$r = (T_{\text{move}} / 255) / (W \times H) = 196 / 307,200 = 0.064\%$$

which is very small. A more portable design expresses the threshold as a fraction of the frame area, and sets it from a baseline session recorded with a compliant candidate. Similarly, LOOKING_AWAY_MARGIN and the contour-area limits should scale with resolution and with the camera-to-candidate distance. We recommend a short dry-run before each exam to confirm that a well-behaved candidate produces few alerts.

VI. RESULTS AND DISCUSSION

Running the system produces, per session, a live annotated display (status, reason, and face count overlays), a chronological text log, original and compressed evidence images, a seat heatmap, and a final PDF report, together providing the invigilator with a concise, reviewable summary instead of hours of raw video. The cooldown and hash filter suppress repetitive captures of an unchanged scene, and JPEG compression at quality 55 keeps reports compact enough for sharing.

Fig. 4 shows the style of the generated heatmap using simulated seat scores for illustration (not measured data); each seat glows green, yellow, orange, or red according to Table II. The text log follows the format sketched in Fig. 5.

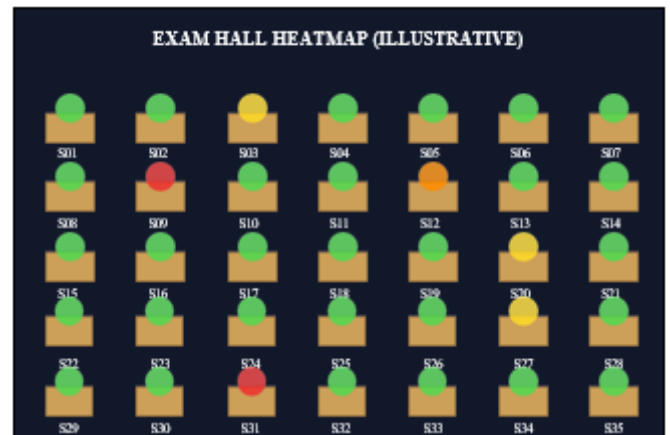


Fig. 4. Illustrative heatmap rendering with simulated seat scores.

```
[YYYY-MM-DD HH:MM:SS] CONFIRMED MALPRACTICE
| Multiple Students | Confidence: 95%
| Evidence: compressed_screenshots/alert_1.jpg
[YYYY-MM-DD HH:MM:SS] DOUBT / REVIEW
| Suspicious Movement | Confidence: 80%
| Evidence: compressed_screenshots/alert_2.jpg
```

Fig. 5. Format of log entries in report.txt (placeholders for time stamps; line breaks added).

Quantitative evaluation. [Insert measured results here: number of test sessions, scripted scenarios (e.g., second person entering, turning sideways, holding a phone), and per-cue precision, recall, and false-alarm rate.] We recommend reporting these per cue, because their reliabilities differ considerably.

Design observations. The multiple-face cue is the most dependable and is therefore the only confirmed class. The remaining cues are heuristic. Notably, MOVEMENT_THRESHOLD is compared against the sum of a 0/255 image, so a value of 50,000 corresponds to roughly 196 changed pixels; this is very sensitive and will likely require tuning or normalization by frame area in practice. Similarly, the Canny-contour object cue responds to any large non-face structure (furniture, clothing, backgrounds) and should be treated as a prompt for human review, not a determination.

Proposed Evaluation Protocol

To measure effectiveness, we recommend scripted sessions in which volunteers perform each behavior in a controlled order, with ground truth annotated by time. Table VI lists the scenarios and the cue expected to respond.

Table 6. Scripted Test Scenarios

Scenario	Expected cue	Expected label
Single student, seated, facing front	None	Normal
Second person enters frame	Multiple students	Confirmed
Large hand/body movement	Suspicious movement	Review
Student leans toward frame edge	Looking away	Review
Phone/paper held in view	Object detected	Review
Student leaves frame	None (not counted)	Normal

For each cue, detections are matched against annotated intervals to obtain true positives (TP), false positives (FP), and false negatives (FN). Standard metrics are then

$$\text{Precision} = \text{TP} / (\text{TP} + \text{FP}), \text{Recall} = \text{TP} / (\text{TP} + \text{FN}) \text{ F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{(\text{Precision} + \text{Recall})}$$

and the false-alarm rate is reported as FP per hour of monitored video. Processing speed should also be reported as analyzed samples per second on the target machine, and robustness should be checked under varying illumination, camera angle, and number of candidates in view.

Comparison with Learning-Based Proctoring

Table VII contrasts the general characteristics of the proposed rule-based design with typical deep-learning proctoring pipelines [6], [7]. The comparison is qualitative.

Table 7. Qualitative Comparison

Aspect	Proposed system	Deep-learning systems
Hardware	CPU, webcam	Often GPU or cloud
Training data	None required	Large labeled sets
Interpretability	High (explicit rules)	Lower (learned features)
Cue accuracy	Limited, heuristic	Potentially higher
Setup cost	Low	Higher
Data leaves site	No (local processing)	Sometimes (cloud)

Operating Workflow

The invigilator starts the program, positions the camera so that the candidates fall inside the frame grid, and leaves it running. The overlay shows the current status, reason, and face count, giving immediate feedback. At the end, pressing Q or ESC stops capture and triggers report generation; on Windows the PDF opens automatically. Fig. 6 summarizes the workflow.

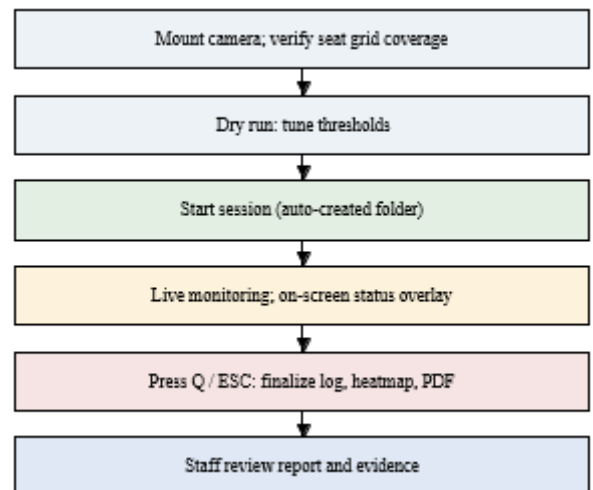


Fig. 6. Operating workflow for invigilators.

Severity Weights

The seat-scoring weights encode the relative certainty of each cue (Fig. 7): the multiple-student cue is weighted highest, and generic movement lowest. Under Table II, one multiple-student event (weight 5) moves a seat to High, while a second such event (score 10) makes it Critical; at least four movement events (score 8) are needed to reach Critical on movement alone. This ordering lets the heatmap emphasize seats with repeated or high-certainty flags rather than seats with incidental motion.

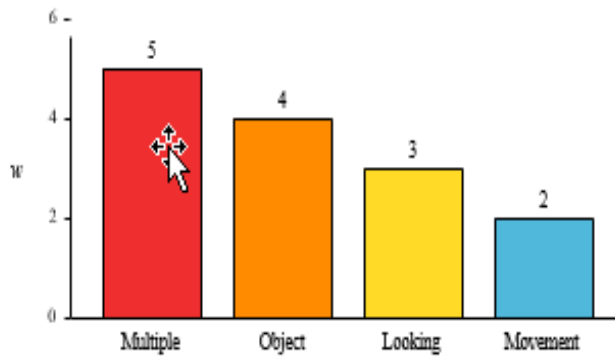


Fig. 7. Per-event severity weights used in seat scoring.

Failure Modes

Because the cues are heuristic, each has characteristic failure modes. Table VIII lists likely causes of false alarms and misses together with practical mitigations; these are design expectations to be confirmed experimentally.

Table 8. Expected Failure Modes and Mitigations

Cue	False alarm	Miss	Mitigation
Multiple students	Poster face; staff passing	Occluded/turned face	Require persistence over several samples
Movement	Light change; auto-exposure	Small, slow motion	Normalize by area; smoothing
Looking away	Seat near frame edge	Turn without center shift	Head-pose estimation; per-seat calibration
Object	Furniture, clothing edges	Small objects	Background model; learned detector

Deployment Guidelines

Table IX summarizes practical recommendations for installing the system in an examination hall.

Table 9. Deployment Recommendations

Factor	Recommendation
Camera placement	Front or elevated rear position with all seats inside the 5×7 grid
Lighting	Even illumination; avoid strong backlight and flicker
Resolution	Fixed; rescale thresholds if changed
Calibration	Dry-run session to tune thresholds
Storage	Access-controlled folder; retention policy
Review	Staff verify every flag before action

Worked Scoring Example

To illustrate the scoring scheme, consider a hypothetical session in which the events in Table XI are recorded. Scores follow $S_i = \text{sum of } w_k \times n_{i,k}$ with the weights of Fig. 7, and risk levels follow Table II. These values are constructed for explanation only and are not measured results.

Table 11. Worked Example of Seat Risk Scoring (Hypothetical)

Seat	Events recorded	Score	Risk
S05	None	0	Low
S09	2 movement	$2 \times 2 = 4$	High
S12	1 multiple + 1 looking away	$5 + 3 = 8$	Critical
S20	1 movement	2	Medium
S31	1 object + 1 movement	$4 + 2 = 6$	High

In this example, S12 is ranked first in the Top Risk Seats panel with two events, followed by S31 and S09. Note that S09 and S31 both fall in the High band even though their composition differs, so the seat event counter shown beside each desk is a useful complement to the color. Staff reading the report would first inspect the screenshots linked to S12.

Scalability

The grid dimensions are configurable constants (ROWS and COLS), so the same code can describe halls of different layout provided the camera sees all seats. For very large halls, a single camera would resolve each face at too few pixels for reliable cascade detection; in that case, one instance per camera, each with its own seat map, is the natural scaling strategy, with reports merged by session. Because processing is local and sampled at 1 Hz, several instances can share one workstation when the cascade cost per sample is small.

Threats to Validity

Any evaluation of this system must account for several threats. First, scripted volunteers may behave differently from real candidates, so staged scenarios can overstate or understate real-world performance. Second, results depend on lighting, camera quality, and seating geometry, limiting generalization. Third, ground-truth annotation of “suspicious” behavior is itself subjective; multiple annotators and an agreement measure are advisable. Finally, since thresholds are tuned by hand, they should be fixed before testing to avoid fitting them to the evaluation data.

Limitations and Future Work

Current limitations include: reliance on frontal Haar detection, which degrades with pose, occlusion, masks, and poor lighting;

the absence of true gaze or head-pose estimation; non-specific object detection; a single-camera, frame-position seat mapping that assumes a fixed camera aligned with the seat layout; and the lack of audio analysis. Because the system processes video of individuals, deployment should include informed consent, restricted access to stored evidence, and defined retention periods. Likewise, all flags must be reviewed by staff before any action is taken.

Ethical and privacy considerations. Exam monitoring involves recording identifiable individuals. Candidates should be informed of what is captured and why; evidence should be stored on access-controlled storage and deleted after the review window; and a flagged event should be treated as a prompt for human judgment, never as proof of misconduct. Because the heuristics can misfire (for example, a candidate adjusting posture or reaching for permitted material), a documented appeal and review process is essential.

Future work includes replacing the cascade with a CNN-based face detector, adding head-pose and gaze estimation, using a trained detector (e.g., YOLO [4]) for phones and papers, supporting multiple cameras with calibrated seat maps, adaptive thresholds based on per-session baselines, audio cues, and a live dashboard for invigilators.

Evidence integrity. Stored evidence can be challenged if its authenticity is unclear. A natural extension is to compute a cryptographic digest of each evidence file at capture time, record it in the log, and keep the log on write-protected storage, so that later tampering becomes detectable.

Table 10. Proposed Enhancement Roadmap

Phase	Enhancement	Benefit
1	Percent-area, per-session adaptive thresholds	Portability across cameras
2	CNN face detector; head-pose and gaze	Fewer misses/false alarms
3	Learned phone/paper detector (e.g., YOLO)	Specific object cues
4	Multi-camera, calibrated seat maps	Larger halls
5	Audio cues; live invigilator dashboard	Richer, real-time context

VII. CONCLUSION

This paper presented Exam Malpractice Detector, a CPU-only, rule-based system that turns a single webcam into an assistant

for exam invigilation. By combining four visual cues, a tiered confidence scheme, filtered evidence capture, seat-level heatmaps, and automated PDF reporting, it offers transparent and inexpensive decision support that keeps human judgment central. The modular design allows individual cues to be upgraded to learned models as accuracy requirements grow.

Acknowledgment

The authors thank the Department of Artificial Intelligence and Data Science, Sengunthar Engineering College (Autonomous), Tiruchengode, for supporting this work.

REFERENCES

1. P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), 2001, pp. I-511–I-518.
2. G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.
3. K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," IEEE Signal Process. Lett., vol. 23, no. 10, pp. 1499–1503, Oct. 2016.
4. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proc. IEEE CVPR, 2016, pp. 779–788.
5. J. Canny, "A computational approach to edge detection," IEEE Trans. Pattern Anal. Mach. Intell., vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.
6. Y. Atoum, L. Chen, A. X. Liu, S. D. H. Hsu, and X. Liu, "Automated online exam proctoring," IEEE Trans. Multimedia, vol. 19, no. 7, pp. 1609–1624, Jul. 2017.
7. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE CVPR, 2016, pp. 770–778.
8. N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in Proc. IEEE CVPR, 2005, pp. 886–893.
9. T. Baltrušaitis, P. Robinson, and L.-P. Morency, "OpenFace: An open source facial behavior analysis toolkit," in Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV), 2016, pp. 1–10.
10. R. C. Gonzalez and R. E. Woods, Digital Image Processing, 4th ed. New York, NY, USA: Pearson, 2018.
11. R. Rivest, "The MD5 message-digest algorithm," RFC 1321, Apr. 1992.
12. C. R. Harris et al., "Array programming with NumPy," Nature, vol. 585, pp. 357–362, Sep. 2020.

13. ReportLab Inc., ReportLab PDF Toolkit User Guide.
[Online]. Available: <https://www.reportlab.com/docs/>