

Real-Time Stampede Risk Prediction and Alert-Driven Officer Dispatch Using YOLO-Based Crowd Density Analysis

Radhika S. Mokati¹, Krunal S. Vighre², Mahesh D. Khadilkar³, Gokul K. Tong⁴
Yogesh D. Yadav⁵, Ritik M. Dhodharmal⁶

^{1,2,3,4,5,6}*Department of Computer Science and Engineering Priyadarshini College of Engineering, Nagpur, India*

Abstract—Crowd crushes at festivals, pilgrimages, railway stations and stadiums are usually recognised only after people are already hurt, and even a correct early warning is of little value if it never reaches someone able to act. This paper describes a software system that joins detection and response in a single pipeline. A YOLO-based person detector processes uploaded recordings or live camera feeds; detections are binned into a 6×8 spatial grid, and a rule-based analyser combines the peak head-count in any one cell with the frame-to-frame growth of the total count to assign one of five risk levels. A decaying heat accumulator gives operators a visual overlay. When the level reaches HIGH or CRITICAL, the alert is stored, authorities are notified by e-mail, and a dispatch engine offers the incident to the nearest on-duty field officer for a limited time, moving to the next-nearest officer after a decline or timeout. The officer's phone application reports position, receives Web Push notifications, and advances the incident through accepted, en-route, arrived and resolved states. We present the architecture, the thresholds and the state machine, report 104 scripted consistency checks on the dispatch logic (all passing), and report a preliminary evaluation on 12 labelled video windows, in which the system's alerts agreed with human labels in 66.7% of windows (recall 0.400 for RISK and 0.857 for NO_RISK). The risk analyser is not yet learned from data and the evaluation set is very small, so we state the resulting limitations plainly.

Index Terms—crowd safety, stampede prediction, person detection, YOLO, crowd density, risk assessment, emergency dispatch, Web Push, real-time video analytics

I. INTRODUCTION

Dense gatherings are a routine part of public life, yet they remain a recurring source of mass-casualty events. In most documented crowd disasters, the physical cause is simple: local density rises beyond what the space can safely hold, movement becomes

involuntary, and people at the front cannot escape the pressure from behind. Analyses of such events point less to a single dramatic trigger than to a chain of small organisational failures, among them late recognition that a problem was developing [1]. Camera coverage is often already present at the venues concerned, but a human operator watching many screens cannot reliably judge density across all of them, and a warning raised on a control-room screen still has to be turned into a person standing in the right place.

Computer vision can supply the first half of that chain. Object detectors trained on large public corpora now run at interactive speed on commodity hardware, which makes it practical to count people in every frame of a video stream [2], [3], and successive releases such as YOLOv7 [4] and the Ultralytics implementations [5] have continued to improve the speed-accuracy trade-off. The second half, getting a responder to the location, is usually treated as a separate human process. The system described here deliberately treats the two as one problem: an alert that is not acted upon has not prevented anything.

We make three contributions. First, we describe a detection-based pipeline that converts raw video into a spatial density grid and a five-level risk indicator using two interpretable signals, peak local density and rate of growth. Second, we describe an alert-to-dispatch mechanism that offers each high-risk incident to one officer at a time, nearest first, and protects the incident record against double acceptance, duplicate incidents and illegal status transitions. Third, we provide an honest account of what has and has not been verified: the dispatch logic is covered by scripted checks,

whereas the agreement of the risk analyser with human labels is reported only as a preliminary result on 12 labelled windows and is not claimed as validated performance.

The remainder of the paper is organised as follows. Section II reviews background. Section III gives the architecture. Section IV details the methods. Section V covers implementation. Section VI reports verification, the evaluation protocol and preliminary results. Section VII discusses limitations, and Section VIII concludes.

II. BACKGROUND AND RELATED WORK

A. Crowd Dynamics and Density Standards

Helbing and co-workers modelled escape panic with interacting self-driven particles and showed that faster, more urgent movement can reduce overall outflow and produce dangerous pressure at bottlenecks [6]. Their later study of the Love Parade disaster treated the event as a systemic failure and examined how density and flow patterns developed before the crush [1]. Pedestrian-planning practice has long used density-based levels of service as a design and assessment aid; the classic reference is Fruin's handbook [7], and Still provides a broader treatment of crowd science for event management [8]. These works motivate using local density, rather than total attendance, as the primary risk signal.

B. Vision-Based Crowd Counting

Two families of approaches dominate image-based counting. Density-regression networks estimate a continuous density map directly from the image and integrate it to obtain a count. The multi-column design of Zhang et al. handles variation in head size [9], and CSRNet uses dilated convolutions to cope with highly congested scenes [10]; Sindagi and Patel survey the field [11]. Detection-based approaches instead locate individuals with bounding boxes. The YOLO family frames detection as a single forward pass over the image [2], and models of this kind are commonly pre-trained on the COCO corpus, in which "person" is one of the categories [3]. Detectors give per-person locations that are easy to bin spatially, but they can lose individuals under heavy occlusion, which is why benchmarks such as MOT20 were assembled specifically from crowded scenes [12]. Point-based methods such as P2PNet [13] predict head locations

directly and so narrow the gap between the two families, and a recent overview surveys the use of YOLO for crowd counting and people-density detection [14].

C. Recent Stampede-Oriented Systems

Recent work has moved from counting towards stampede-specific analysis. Rao et al. [15] combine a YOLOv8 person detector with heatmap-based density classification and LSTM trend analysis, and report alerts to authorities together with control actions such as gate regulation or voice announcements. Cob-Parro et al. [16] take a different route, classifying dense optical-flow features frame by frame to decide whether a stampede is under way in low- and medium-density crowds. Yue and Wang [17] review deep learning for crowd behaviour prediction and recognition more broadly. These systems differ in how the risk is estimated; the part of the chain that follows an alert, namely who is sent and how the assignment is tracked, receives comparatively little attention in their descriptions.

D. Positioning of This Work

Most counting research reports error against annotated crowds and stops at the estimate. Our emphasis differs: we use an off-the-shelf pre-trained detector, keep the risk logic transparent, and invest the engineering effort in what happens after a risky frame is found, namely persistence, notification and responder assignment. The trade-off is that detection accuracy in extremely dense scenes is bounded by the detector, a point we return to in Section VII.

III. SYSTEM ARCHITECTURE

Fig. 1 summarises the system. Video enters through one of two modes. In Mode 1, an authenticated user uploads a recording that is processed in a background thread, and the dashboard polls a status endpoint until processing ends. In Mode 2, a webcam, RTSP stream or IP camera is read by a threaded reader and its frames are passed to the same analysis stages, either through a Fluvio topic or, when no Fluvio cluster is reachable, directly in-process. Because both modes call the same detector, density estimator and risk analyser, behaviour measured on recordings carries over to live feeds.

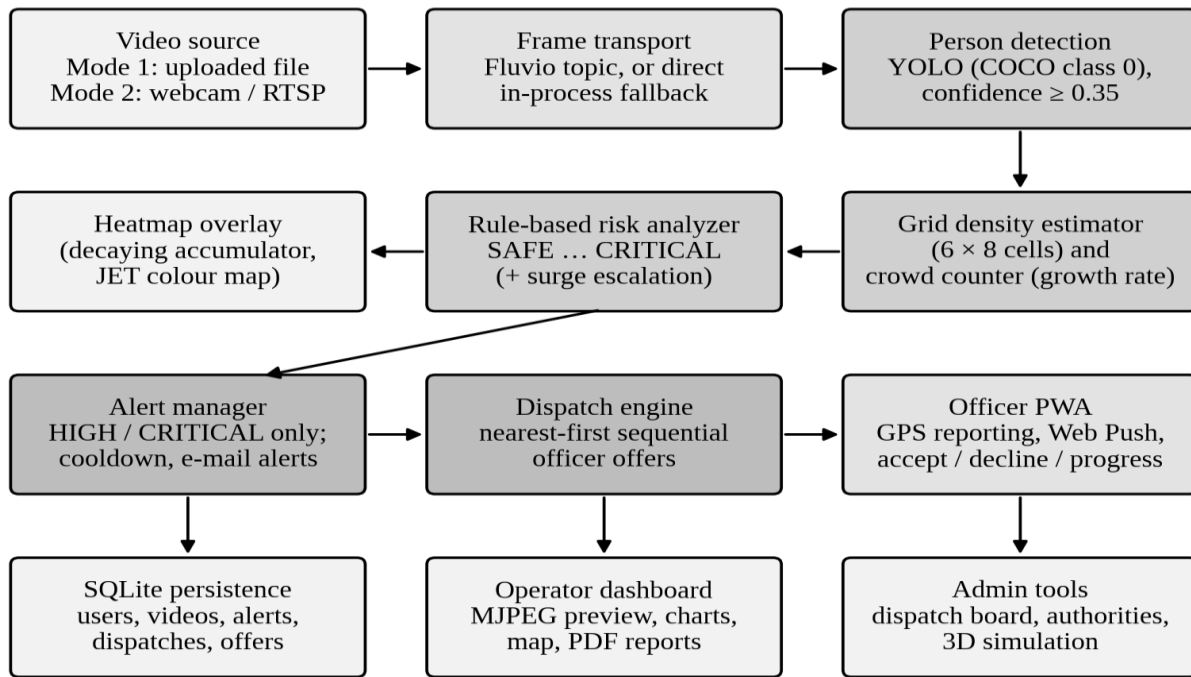


Fig. 1. Overall system architecture. Rows run left to right in a snake pattern: acquisition and detection, density and risk analysis, alerting and dispatch, and persistence and operator interfaces.

Downstream of analysis, any frame that reaches HIGH or CRITICAL produces a persisted alert. Alerts feed two consumers: an e-mail notifier for registered authorities and the dispatch engine. State is stored in a relational schema containing users, videos, alerts, authority contacts, alert deliveries, camera locations, dispatches, dispatch offers and push subscriptions. Three user roles exist: operator, administrator and field officer; account registration is protected by a one-time code sent by e-mail.

IV. METHODOLOGY

A. Person Detection

Each analysed frame is passed to a pre-trained Ultralytics YOLO model [5], using the nano variant of the YOLO26 family shipped with the project. The call is restricted to COCO class 0 (person), uses an inference size of 640 pixels and discards detections below a confidence of 0.35. For a box with corners (x_1, y_1) and (x_2, y_2) , only the centroid is retained for later stages:

$$c = ((x_1 + x_2)/2, (y_1 + y_2)/2) \quad (1)$$

To keep CPU-only machines usable, inference runs on every second frame by default, and live frames are resized to a width of 640 pixels before detection. The remaining frames reuse the most recent result.

B. Grid-Based Density Estimation

A global head-count cannot distinguish twenty people spread across a plaza from twenty people pressed into one corner, and the second case is far more dangerous. The frame of width W and height H is therefore divided into $R = 6$ rows and $K = 8$ columns. The count in cell (r, k) is

$$G(r, k) = \sum_j 1[\lfloor c_{x,j} K/W \rfloor = k \wedge \lfloor c_{y,j} R/H \rfloor = r] \quad (2)$$

where indices at the frame edge are clamped to the last row or column. The quantity used for risk assessment is the peak cell count

$$D_{\max} = \max_{r,k} G(r, k) \quad (3)$$

The mean count over occupied cells and the total count are also recorded for reporting.

C. Crowd Growth Rate

A short rolling history of per-frame totals is kept. With n_t the total at the current inference step, the growth rate is

$$g_t = (n_t - n_{t-1}) / n_{t-1} \quad (4)$$

with the conventions that $g_t = 0$ when fewer than two observations exist, and that when $n_{t-1} = 0$ the rate is 0 if n_t is also 0 and 1 otherwise. The peak count over the session is tracked as well.

D. Rule-Based Risk Analysis

The analyser first maps $D_{\max}$ to the highest level whose threshold it meets (Table I). If the growth rate is at least 0.35 and the level is not already CRITICAL, the level is raised by one step, so that a fast-building

crowd is treated as more serious than a static crowd of the same peak density. Fig. 2 illustrates the mapping.

TABLE I. RISK LEVELS AND DENSITY THRESHOLDS

Level	Threshold on $D_{\max}$	Smallest integer count that qualifies	Triggers alert and dispatch
SAFE	0.0	0	No
LOW	1.5	2	No
MEDIUM	3.0	3	No
HIGH	5.0	5	Yes
CRITICAL	7.5	8	Yes

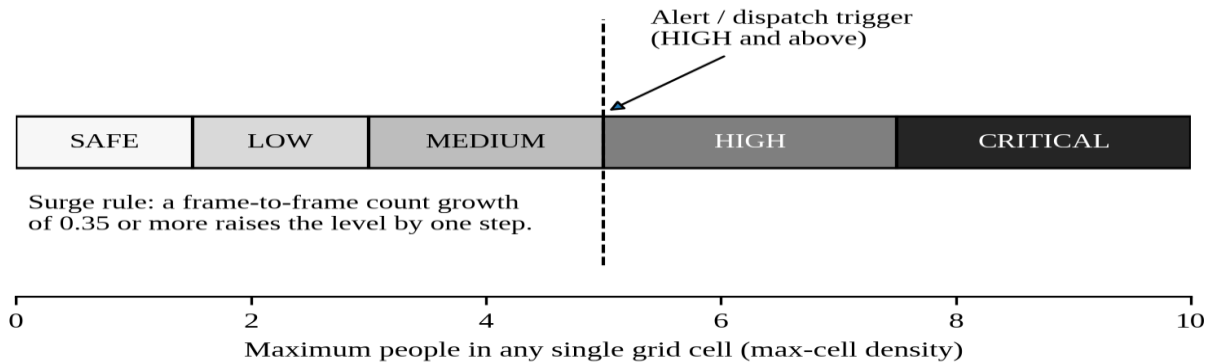


Fig. 2. Mapping from peak cell density to risk level, with the alert threshold and the surge rule.

Because cell counts are integers, the real-valued thresholds act as the integer cut-offs shown in the third column of Table I. A single person in a cell therefore never raises the level above SAFE on density alone. The rule is intentionally simple: every output can be explained to an operator in one sentence, such as “maximum cell density is six people; surge detected.”

E. Heatmap Overlay

To show where pressure is building, a heat accumulator with the same shape as the grid is updated at every inference step:

$$M_t = \lambda M_{t-1} + G_t, \quad \lambda = 0.94 \quad (5)$$

For display, M_t is normalised to 8 bits, enlarged to frame size by cubic interpolation, smoothed with a 15×15 Gaussian kernel, coloured with a JET colour map and blended with the frame at 45% opacity. The decay factor leaves a fading trail instead of a flickering per-frame picture.

F. Alert Generation

An alert is persisted when the risk level is in the trigger set {HIGH, CRITICAL}. For live streams a new alert at the same level is suppressed for 15 s so that a camera that stays critical does not write one row per frame, and e-mail to authorities is rate-limited with a 120 s cooldown. The same trigger set is reused by the dispatch module, on the principle that an event not worth an e-mail is not worth waking an officer.

G. Nearest-First Officer Dispatch

Dispatch applies only to alerts whose video or live session is tagged with a registered camera location that has coordinates, since otherwise there is nowhere to send anyone. The sequence is as follows.

Commit first. The alert is committed to the database before any dispatch work begins, so an officer is never paged about an incident that failed to save.

Deduplicate. A sustained crisis generates many alerts but is one incident. A new alert within 150 m of an

incident created in the last 600 s is folded into the existing record instead of opening another.

Rank. Eligible officers are those who are on duty, verified, not already committed to another incident, not already asked about this one, and whose last reported position is no more than 300 s old. They are ordered by great-circle distance using the haversine formula [18], which remains numerically stable at the short ranges that matter here:

$$d = 2R_E \arcsin \sqrt{(\sin^2(\Delta\phi/2) + \cos \phi_1 \cos \phi_2 \sin^2(\Delta\lambda/2))} \quad (6)$$

with $R_E = 6371.0088$ km. Officers farther than a 15 km search radius are excluded.

Offer. Only the nearest eligible officer receives the offer, with 25 s to respond. At most one offer is pending per incident, which makes a double acceptance structurally impossible rather than merely

unlikely. A decline or timeout moves the incident to the next-nearest officer, up to five offers. If nobody accepts, the incident becomes UNASSIGNED and administrators are e-mailed once; a later alert from the same crowd re-searches that same record without opening a duplicate or repeating the e-mail.

Progress. After acceptance the officer moves the incident through EN_ROUTE, ARRIVED and RESOLVED (Fig. 3). Each transition is checked on the server against an allow-list and against the identity of the accepting officer, so an incident cannot be resolved without being accepted, cannot be walked backwards, and cannot be advanced by someone else. Resolving the incident also acknowledges the parent alert, since an officer at the scene is the most reliable acknowledgement available. Transitions are serialised by an in-process lock.

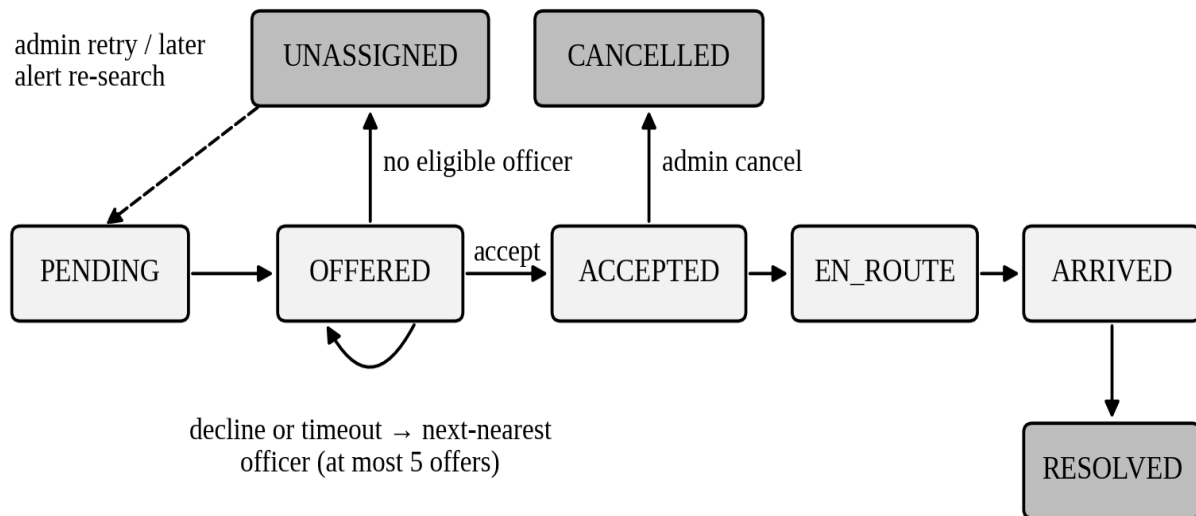


Fig. 3. Dispatch incident life cycle. Shaded states are terminal.

Officers use a progressive web application that reports position every 20 s while on duty, polls for offers while open, and receives background notifications through the Web Push protocol [19] with voluntary application-server identification [20]. On acceptance, navigation is handed to a maps application. The estimated time of arrival shown beforehand is computed from straight-line distance at an assumed

average speed of 25 km/h and is labelled as an estimate.

V. IMPLEMENTATION

The system is written in Python. The web tier uses Flask with Flask-Login for sessions, Flask-WTF for form handling and SQLAlchemy over SQLite for

persistence. Image processing relies on OpenCV [21] and NumPy; detection uses the Ultralytics package; PDF incident reports are built with ReportLab; and the evaluation module uses scikit-learn [22]. The user interface is built with Bootstrap 5 with Chart.js dashboards for recorded-video analytics, an MJPEG preview for live feeds, and a WebGL (Three.js) scene for the administrator simulation. Table II lists the parameters that govern behaviour; all are held in a single configuration module, and the dispatch parameters can be overridden through environment variables.

TABLE II. PRINCIPAL CONFIGURATION PARAMETERS

Parameter	Value	Role
Detector confidence threshold	0.35	Minimum score for a person box
Detector input size	640 px	Inference resolution
Density grid (rows × columns)	6 × 8	Spatial binning
Frames between inferences	2	CPU load reduction
Surge growth threshold	0.35	One-level escalation
Heat decay factor λ	0.94	Overlay persistence
Live alert cooldown	15 s	Suppresses repeats
E-mail cooldown	120 s	Rate-limits notifications
Offer timeout	25 s	Time to accept an offer
Maximum offers per incident	5	Escalation depth
Search radius	15 km	Eligible officer range
Incident de-duplication	150 m / 600 s	Fold repeats into one incident
Maximum location age	300 s	Rejects stale officer positions
Location report interval	20 s	Officer GPS updates

Two engineering decisions deserve mention. First, the dispatch engine is deliberately wired to the same code path whether an incident is raised by an alert or manually from the administrator simulation, so that a demonstration exercises the production logic rather than a parallel imitation of it. Simulated incidents carry a flag that excludes them from average-response-time statistics, and demonstration officers are created with unusable credentials on a reserved non-routable e-mail domain. Second, the optional Fluvio client is a compiled extension without a native Windows wheel, so it is kept out of the main dependency list and the application falls back to direct in-process streaming when it is absent.

Figs. 4–6 show the operator interface. In live mode (Fig. 4) the annotated stream is shown with the current person count, peak cell density, risk level and processing rate, and a label indicates whether frames are analysed directly in-process or through Fluvio. Fig. 5. Live CCTV Monitoring Interface of the Proposed Stampede Prediction System. The interface displays YOLO-based person detection using bounding boxes and a crowd heatmap overlay to visualize areas of high crowd concentration. The dashboard reports 48 detected people, a density value of 4, a HIGH risk level, and a processing rate of 4.1 FPS. The system operates in direct camera mode, enabling real-time crowd monitoring and risk assessment., and Fig. 6 shows the analytics page produced for a processed recording. Figs. 7 and 8 show the administrator simulation and the dispatch board, whose incidents are simulated drills rather than real events.

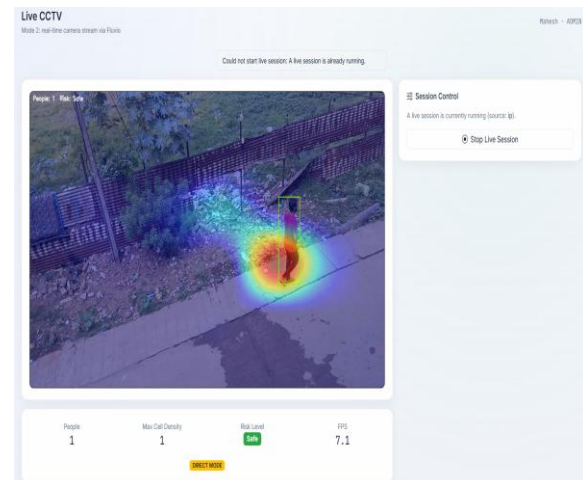


Fig. 4. Live CCTV page: annotated stream with heat overlay, people count, maximum cell density, risk level and frame rate (direct mode, no Fluvio cluster).

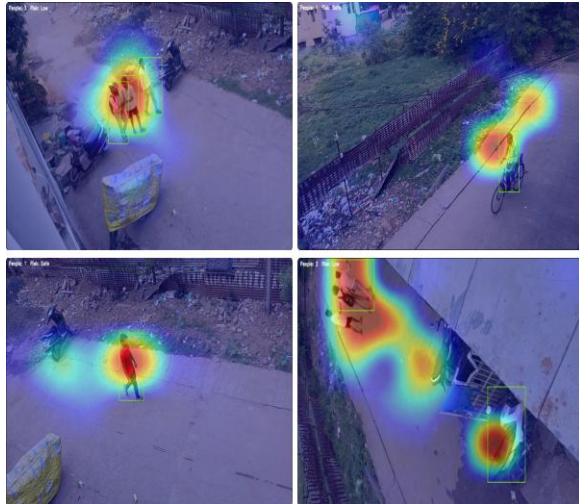


Fig. 5. Detection boxes and heatmap overlay on live frames: (a) three people close together, level LOW; (b) one person, SAFE; (c) one person, SAFE; (d) two people in a rotated camera view, level LOW ;(c) reports 48 detected people, a density value of 4, a HIGH risk level, and a processing rate of 4.1 FPS.

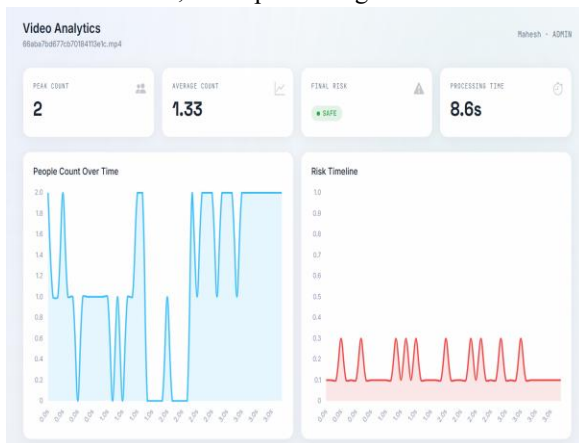


Fig. 6. Analytics page for a processed recording: peak and average count, final risk level, processing time, count over time and risk timeline.

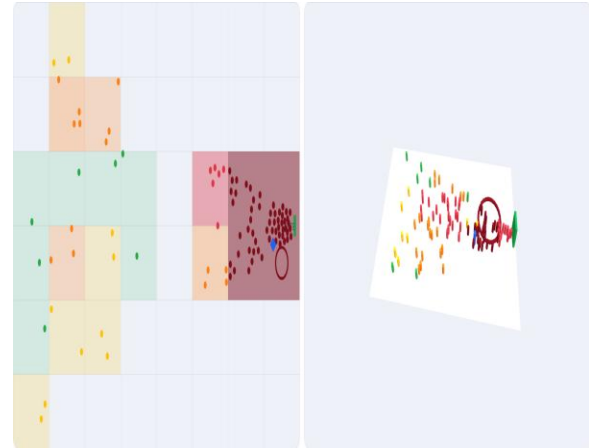


Fig. 7. Crowd-crush simulation in 2D top-down (left) and 3D (right) views with one gate: 94 people, a peak of 16 per cell and level CRITICAL in the 2D run.

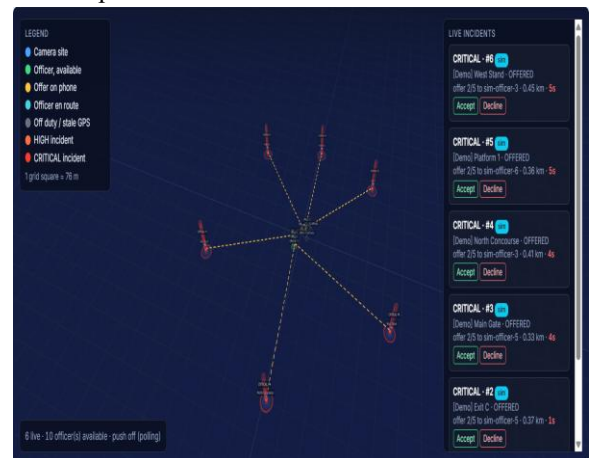


Fig. 8. Dispatch board with simulated CRITICAL incidents, each offered to the nearest available officer, shown with a countdown and Accept/Decline controls.

VI. VERIFICATION AND EVALUATION

A. Functional Verification of the Dispatch Logic

The repository includes two self-contained verification scripts. Each combines static checks on the engine source with a scripted walk through a re-implementation of the state machine that uses the project's own distance code. We ran both on the version described here; the outcomes are summarised in Table III. The scenarios cover nearest-first ordering, the one-pending-offer invariant, escalation in distance order, the five-offer ceiling, refusal of late or duplicate acceptances with distinct reasons, rejection of illegal transitions, exclusion of busy officers and officers with stale positions, de-duplication of a ten-alert surge

into a single incident with a single administrator e-mail, and, for the simulation, purging only simulated rows while keeping any real incident that references a demonstration record. Negative controls confirm that a broader purge would have destroyed real data and that omitting the cosine-of-latitude term would distort east–west distances by roughly 6% at the seeded location.

TABLE III. SUMMARY OF SCRIPTED VERIFICATION RUNS

Script	Checks passed	Checks failed	Scope
Dispatch lifecycle	54	0	State machine, ranking, de-duplication, access control
Simulation and purge scoping	50	0	Purge boundaries, seeded geometry, demo-officer limits

These checks establish that the assignment logic behaves as specified. They say nothing about whether the vision pipeline raises the right alerts, which is a separate question.

B. Protocol for Measuring Detection-to-Alert Agreement

The repository provides an evaluation module that runs the production pipeline, without modification, over human-labelled video windows. Each window is annotated as RISK or NO_RISK by observers who judge whether the footage shows dangerous crowding. A window is predicted RISK if the analyser reaches HIGH or CRITICAL on any processed frame inside it, which is exactly the condition that triggers a real alert. The module outputs per-window predictions, a classification report with precision, recall and F1 for each class, overall accuracy, and a confusion matrix. Candidate footage includes self-recorded clips, scripted clustering by volunteers, and public crowd

benchmarks such as MOT20 [12]. We recommend at least fifteen to twenty windows of both classes, and that the sample size be stated beside every reported figure. The evaluation reported in Section VI-C falls below this recommendation.

C. Experimental Evaluation

The proposed stampede risk assessment pipeline, consisting of the person detector, density estimator and risk analyser, was evaluated on 12 labelled video windows using the protocol of Section VI-B; the results are given in Table IV. The system achieved an overall accuracy of 66.67%. For the NO_RISK class, precision, recall and F1-score were 66.7%, 85.7% and 75.0%, respectively. For the RISK class, the corresponding values were 66.7%, 40.0% and 50.0%.

TABLE IV. DETECTION-TO-ALERT AGREEMENT

Class	Precision	Recall	F1-score	Windows (n)
RISK	0.667	0.400	0.500	5
NO_RISK	0.667	0.857	0.750	7
Overall accuracy	—	—	0.667	12

Note: Precision, recall and F1-score are reported per class. Overall accuracy is 0.667 (66.7%) across 12 labelled windows.

The reported figures imply two true positives, three false negatives, one false positive and six true negatives for the RISK class. The results indicate that the system performed better at identifying NO_RISK windows than RISK windows. The low recall for the RISK class indicates that some risk-labelled windows were not detected. Further evaluation using a larger, independently labelled dataset and refinement of the risk-classification rules are required. Since the current evaluation contains only 12 windows, these results should be considered preliminary rather than evidence of validated real-world performance. They describe how well a fixed rule with hand-set thresholds agrees with human judgement on the chosen clips; they do not measure the generalisation of a trained model.

VII. DISCUSSION AND LIMITATIONS

Hand-set thresholds. The values in Table I are design choices, not fitted parameters. The 12-window

evaluation in Section VI-C is far too small to tune or test them; with a larger labelled set they could be tuned on one portion of the clips and tested on another, for example by examining the trade-off between missed risky windows and false alarms.

Small evaluation set. Table IV rests on 12 windows, 5 labelled RISK and 7 NO_RISK, so a single window moves overall accuracy by 8.3 percentage points and the per-class figures by considerably more. No confidence intervals are given because they would be very wide. The RISK recall of 0.400 means that three of the five risky windows did not reach HIGH or CRITICAL, which is the failure that matters most for a warning system.

Image-space density. Cells are defined in the image, not on the ground. With an oblique camera, a cell far from the lens covers much more floor than one near it, so the same count means different physical densities in different parts of the frame. Mapping detections to a ground plane through a calibrated homography would allow density in persons per square metre and a comparison against established levels of service [7].

Detector limits. A detector pre-trained on general imagery can miss heavily occluded people, and it counts bodies rather than measuring pressure. Under-counting in the densest scenes would bias the system toward under-reporting exactly when it matters most. Density-regression models [9], [10] and point-based localisation [13] are alternatives for such scenes.

Volatile growth rate. Growth is computed between two consecutive inference steps, so with small counts a change of one person is a large fraction; moving from two to three people already exceeds the 0.35 surge threshold, and by definition a person appearing in an empty scene yields a rate of 1. Smoothing over a window, or requiring a minimum count, would reduce spurious escalation.

No motion analysis. The pipeline does not track individuals, so it cannot measure velocity, flow direction or the stop-and-go turbulence associated with crowd disasters [1]. Adding a tracker such as ByteTrack [23] would enable such features, and optical-flow classifiers have been applied to stampede detection directly [16].

Dispatch scope. Distances are straight-line, ignoring roads and traffic. The state machine is protected by a process-local lock, so it assumes one server process; running several would require moving transitions into database transactions. Browsers expose geolocation

and service workers only in secure contexts, so officer devices need HTTPS outside localhost. An incident with no registered coordinates cannot be dispatched.

Responsible use. The system counts people and does not identify them, but it processes video of public spaces. Deployments should follow local privacy law, restrict access to footage, and keep a human decision-maker responsible for acting on alerts; the system is intended as decision support, not as an autonomous authority.

VIII. CONCLUSION AND FUTURE WORK

We presented a stampede-risk system that goes from camera frames to an assigned field officer within one coherent pipeline. Its risk logic is intentionally transparent, combining peak cell density with crowd growth, and its response logic is defensive about the failure modes that matter in practice: duplicate incidents, double acceptance, stale positions and illegal status changes. The dispatch module passes 104 scripted checks, while a preliminary evaluation on 12 labelled windows gave 66.7% overall agreement with human labels, with a RISK recall of only 0.400. This sample is too small to establish real-world performance.

Planned work includes extending the labelled evaluation to a larger, independently labelled dataset, calibrating thresholds on that data, ground-plane density through camera calibration, multi-object tracking to obtain velocity and flow features, a learned classifier to cross-check the rule-based analyser, road-network routing for dispatch, and transactional state handling for multi-process deployment.

REFERENCES

- [1]. C. Cob-Parro, C. Losada-Gutiérrez, and M. Marrón-Romera, "Stampede detector based on deep learning models using dense optical flow," *Engineering Applications of Artificial Intelligence*, 2025.
- [2]. Relevance: Directly relevant to stampede detection using optical flow and deep-learning classification; useful for comparing motion-based detection with density-based rules.
Link:
<https://doi.org/10.1016/j.engappai.2024.109940>

- [3]. G. Gao, J. Gao, Q. Liu, Q. Wang, et al., “A survey of deep learning methods for density estimation and crowd counting,” *Vicinagearth*, vol. 2, article 2, 2025.
- [4]. Relevance: Reviews crowd-counting models, density-map estimation, evaluation methods, and limitations of detection-based counting in dense scenes.
Link: <https://doi.org/10.1007/s44336-024-00011-8>
- [5]. M. Alasmari, N. S. Farooqi, and Y. A. Alotaibi, “Recent trends in crowd management using deep learning techniques: A systematic literature review,” *Journal of Umm Al-Qura University for Engineering and Architecture*, 2024.
- [6]. Relevance: Potential background source for AI-based crowd monitoring, crowd management, and public-safety applications. Verify the author list, title, venue, and DOI against the publisher record.
Link: <https://doi.org/10.1007/s43995-024-00071-3>
- [7]. J. Yue and H. Wang, “Recent deep learning in crowd behaviour analysis: A brief review,” *arXiv preprint arXiv:2505.18401*, 2025.
- [8]. Relevance: Reviews deep-learning approaches for crowd-behaviour prediction and recognition. This is a preprint rather than a peer-reviewed journal article.
Link: <https://arxiv.org/abs/2505.18401>
- [9]. G. Bhola, S. Srivastava, H. R. Nidhan, et al., “Stampede detection and crowd analysis using CNN-LSTM and Farneback optical flow,” *Scientific Reports*, vol. 16, article 16155, 2026.
- [10]. Relevance: Described as combining optical-flow features with CNN–LSTM for crowd-risk classification; useful for comparison with a rule-based temporal risk analyser. Verify metadata against the journal page.
Link: <https://doi.org/10.1038/s41598-026-45262-1>