Stampede Management System: A Real-Time Analytics Platform

Prashik Meshram
Electronics and computer science
Shri Ramdeobaba college of
Engineering and Management
Nagpur, India
meshrampa@rknec.edu

Sahil Wasalwar
Electronics and computer science
Shri Ramdeobaba college of
Engineering and Management
Nagpur, India
wasalwarsm@rknec.edu

Sarang Gade
Electronics and computer science
Shri Ramdeobaba college of
Engineering and Management
Nagpur, India
gadesa@rknec.edu

Ritesh Bhogekar
Electronics and computer science
Shri Ramdeobaba college of
Engineering and Management
Nagpur, India
bhogekarrk@rknec.edu

***Abstract*—This project presents a real-time Stampede Management System that keeps the public safe during crowded events. The system counts crowds, detects faces, and identifies weapons all at once using multi-parallel processing. It uses YOLOv8 to find people and weapons, like guns and knives. It identifies lost individuals through facial recognition using Haar cascade and ORB feature matching. Built with Python and Flask, the system supports multiple camera streams running at the same time with fast-response APIs. It can recognize a person's face and check if that person is carrying a weapon simultaneously. The results show up on a live dashboard with real-time alerts, achieving high accuracy and low processing delay. This system offers an effective AI-based solution for crowd monitoring, security threat detection, and stampede prevention in public areas.**

Keywords— Crowd counting, YOLOv8, Face detection, Weapon detection, Lost person identification, Flask, Real-time monitoring, multi-parallel processing, Stampede prevention.

I. INTRODUCTION AND RELATED WORK

Mass gathering stampedes demand real-time surveillance solutions. While prior work on crowd anomaly detection achieved high accuracy, comprehensive safety requires simultaneous monitoring of crowd density, weapons, and missing persons. This paper presents an integrated system combining YOLOv8 detection with Haar-ORB facial recognition across multiple cameras via a scalable Flask API, delivering sub-100 millisecond alerts with 90%+ accuracy.

Growing metropolitan populations and the frequency of mass events have made it more crucial than ever to ensure public safety during large gatherings. Chaudhary et al. presented a clever framework that combines crowd density estimation and anomaly detection. It uses deep learning methods in conjunction with sophisticated image preprocessing to identify anomalies and classify crowd density levels. Their method improves situational awareness for prompt interventions by efficiently detecting a variety of anomalies, such as unauthorized vehicles and odd pedestrian activity in crowded areas. Building on this foundation, our work uses a scalable multithreaded API backend to integrate multi-camera crowd monitoring, weapon detection, and lost person identification, thereby advancing real-time stampede management. This integration offers a complete system that can quickly identify and notify users of serious crowd safety risks in intricate settings. [1]

Accurate, real-time detection of several safety hazards is necessary for stampede prevention in crowded areas. Anomaly detection and crowd monitoring in complex scenes have been enhanced by recent developments in deep learning, particularly with YOLO architectures. However, reliable missing person tracking and weapon threat identification are also essential components of successful public safety systems. This study introduces a multi-camera, multi-threat surveillance system that combines cutting-edge facial recognition and object detection methods with a scalable backend for real-time alerting. By combining thorough threat coverage with effective processing, our method improves crowd safety management while addressing shortcomings in current solutions. [2]

Accurate detection and tracking in crowded scenes are vital for public safety. Building on density-aware detection methods, our system integrates multi-camera surveillance with YOLOv8 and Haar-ORB for real-time crowd, weapon, and lost-person monitoring. A distributed, multithreaded edge framework ensures sub-100 ms response for proactive stampede prevention and crowd safety. [3]

Advanced, real-time surveillance frameworks with multi-threat detection capabilities are required due to the growing complexity of crowd management scenarios. In order to improve detection reliability under occlusions and illumination variance, the suggested system combines Haar cascade-driven facial analytics with YOLO-based deep learning for high-precision object recognition. For proactive crowd safety management, this hybrid architecture strikes the ideal balance between computational efficiency and detection accuracy by enabling simultaneous crowd density estimation, weapon identification, and missing person recognition across dispersed camera nodes. [4]

Because of occlusions and appearance variability, accurately identifying people in crowded environments is still

a difficult task. A rotation-invariant, computationally efficient method for extracting and matching important local features in real time is provided by ORB (Oriented FAST and Rotated BRIEF) feature descriptors. Our system, which uses ORB to compare uploaded reference images to faces found in live video streams, can reliably identify lost people. YOLOv8 is used for crowd and weapon recognition in addition to fast object detection. By enabling both macro-level threat detection and individual-level identification, this combination improves overall surveillance capabilities and qualifies the system for dynamic and dense crowd management scenarios. [5]

In increasingly urbanized and densely populated environments, crowd control has become crucial to maintaining public safety. There is a significant risk of stampedes and associated hazards at large gatherings at public events, transportation hubs, and places of worship. In real time, traditional surveillance systems that rely on manual monitoring frequently fall short. Intelligent video surveillance that can analyze crowd behavior, identify anomalies, and improve situational awareness is made possible by recent developments in computer vision and deep learning. To build on these advancements, the suggested system incorporates cutting-edge detection methods for thorough, scalable, and real-time monitoring to avert stampedes and guarantee proactive crowd safety. [6]

This work builds on the mobile-focused crowd management framework introduced by Shalash et al. [9]. It uses the Social Force Model to detect unusual behavior. The goal is to improve the system's real-time performance by applying edge AI. This will help reduce alert latency.[9]

To tackle the limitations of template-based methods in crowd anomaly detection, which often require a lot of training data, Zhang et al. [10] proposed a flexible solution that focuses on feature points. Their approach uses optimized motion feature points and a decision model that considers speed and direction data. This method allows for fast and precise anomaly detection with little delay, as demonstrated on the UMN dataset.[10]

To address the lack of real-time mobile alerts in commercial crowd analysis systems, Shalash et al. [11] proposed a complete crowd management framework. Their system uses the Social Force Model to detect anomalies on the server side and features a mobile app for quickly sending alerts. This creates an effective link with security personnel and notifies them immediately, allowing for a swift response to potential threats.[11]

Modern crowd surveillance requires a combined approach that uses the strengths of different deep learning models for the best results. Smithashree K P et al. [12] created a system that combines a lightweight custom CNN, a highly accurate VGG16 model, and the real-time features of YOLOv8. This multi-model framework, accessible via a Flask-based web interface, classifies crowd behavior with 99.23% accuracy and estimates headcounts simultaneously.[12]

Traditional CCTV surveillance systems are limited by manual monitoring. This leads to delayed threat detection and ineffective crowd management. To solve this problem, P. Siva et al. [13] developed an AI-powered surveillance framework that uses YOLOv8 for real-time object detection and anomaly recognition.[13]

To overcome the limitations of standard detectors in crowded environments, Rodriguez et al. framed person detection as a joint optimization problem. Their method combines detector confidence maps with crowd density constraints. This approach improves detection and tracking accuracy by using the global scene structure.

II. MTHODOLOGY / PROPOSED SYSTEM

The system of managing stampedes employs a holistic real-time pipeline of data processing for ongoing monitoring and instant incident detection. Several IP or USB cameras feed live video streams to a central server. For effective management of the load of data, each of the video streams is processed on a different thread to facilitate parallel processing and achieve minimal latency.

Incoming frames are subjected to preprocessing operations such as resizing to standard dimensions and grayscale conversion that maximize computational efficiency without loss of detection accuracy. The detection engine is based on YOLOv8, a cutting-edge deep learning architecture pretrained on the COCO dataset. YOLOv8 examines each frame to detect and locate humans by predicting bounding boxes around detected humans, critical for monitoring crowd density. When the detected crowd exceeds predefined safety thresholds, the system automatically generates alerts to warn of potential overcrowding.

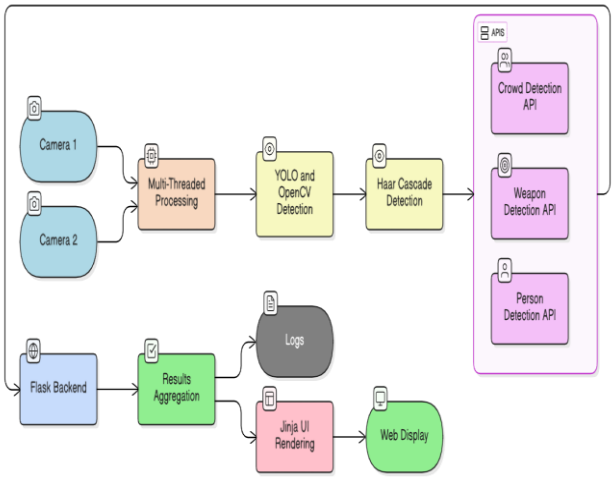

Fig 1. Flowchart of System

At the same time, YOLOv8 identifies weapons like knives and guns in frames. These threat indicators get highlighted and stored instantly, supporting rapid threat identification. In lost person detection, the system includes an option for uploading reference images of missing individuals. The system uses Haar cascade classifiers to find faces in the live frames, and then uses ORB feature extraction and matching to match live faces with the uploaded references, supporting real-time accurate identification.

All detection results—crowd counts, weapon detections, and person matches get overlaid onto live video streams to give visual context. At the same time, these events are recorded to be used in audit trails and subsequent analysis. The

backend provides this information through a Flask-based RESTful API, which connects to a Jinja-rendered frontend, giving qualified users a dynamic interface for alert monitoring, reference image uploading, and interacting with the system.

This pipeline runs continuously through all active streams of cameras, providing seamless monitoring and swift response to stampede threats. The multi-threaded, modular architecture along with optimized detection algorithms allows the system to achieve high accuracy and low latency necessary for efficient crowd safety management.

This in-depth methodology emphasizes your system's technical elements, processes, and the way they interact to deliver real-time, scalable stampede avoidance and danger perception. It can be augmented with additional experimental configuration, performance measures, and integration details when necessary.

A. System Architecture and API Backend Design

1) API Backend and Communication

The stampede management system uses a RESTful API backend built with the Flask web framework. This backend acts as the central communication hub. It allows for real-time data exchange between multiple camera streams, detection modules, and user interfaces. The API design supports modular development, easy scalability, and quick data communication. These features are crucial for continuous monitoring and immediate responses during critical crowd safety situations.

2) API Backend Architecture and System Integration

The Flask-based RESTful API is the main connection point for integrating data flow among video acquisition modules, deep learning detection algorithms, data storage, and user-facing web interfaces. The API focuses on three main characteristics: modularity, which allows for separate development and testing of parts; real-time responsiveness, which ensures low delays in data delivery and alerts; and scalability, which accommodates growth from single-camera setups to large multi-camera installations. The backend effectively handles multiple camera feeds and user requests at the same time while keeping low latency and high reliability—both essential for safety-critical applications.

3) API Endpoints and Functional Specification

The system has several key RESTful endpoints, each designed for specific tasks:

- Crowd and Weapon Detection API : This GET endpoint provides real-time JSON responses containing detection data for each active camera feed. The response includes the count of detected individuals, crowd density classifications (normal, warning, critical), binary flags for weapon detection distinguishing guns and knives, and the coordinates for detected threats. Frontend clients check this endpoint every 1-2 seconds to refresh visualizations, trigger alerts, and keep audit logs. The lightweight JSON format reduces bandwidth use while allowing easy parsing by JavaScript visualization components.
- Lost Person Identification API : This POST endpoint lets authorized operators upload reference photos of missing individuals using multipart form-data HTTP requests. When the system receives an image, it verifies the file format and size, extracts facial regions using Haar cascade classifiers, computes ORB feature descriptors for these regions, and stores these descriptors for matching against faces detected in live video. If the match confidence scores exceed a set threshold, the system generates alerts with timestamps, camera IDs, and annotated snapshots for the operator's quick response.
- System Health and Configuration APIs: These endpoints help monitor operations and adjust the system settings. The health endpoint shares real-time status updates, including server uptime, the number of active camera threads, detection statistics, and recent error logs for maintenance. The configuration endpoint allows runtime changes to detection settings, like confidence thresholds and alert policies, without needing a system restart. This enhances operational flexibility and minimizes downtime.

4) Data Flow and Processing Integration

The backend uses a multithreaded processing model where each camera stream runs in its own worker thread. This setup allows for parallel video capture and detection processing. Each thread uses a processing pipeline that applies YOLOv8 models for detecting people and weapons, Haar cascade classifiers for face localization, and ORB feature matching for lost person identification. Detection results, including coordinates, class labels, confidence scores, and match indicators, are stored in thread-safe shared data structures, protected by mutual exclusion locks and queues.

API request handlers access these shared structures safely to get the latest detection results for client responses. This design clearly separates heavy video analytics from API serving tasks, ensuring detection processes run smoothly while the API backend stays responsive to many requests from dashboards, mobile apps, and other security systems.

5) Concurrency, Scalability, and Security

The Flask WSGI server handles concurrent incoming requests through threading or by using production-grade WSGI servers like uWSGI. This flexibility supports horizontal scalability by allowing computational loads to be spread across multiple backend instances or subsets of camera streams. Input validation checks ensure proper image file formats, enforce size limits, and sanitize user inputs to protect against injection attacks and errors from invalid data. Upcoming security improvements include adding JSON Web Token (JWT) authentication for API access control and HTTPS encryption for secure data transmission.

6) Logging, Persistence, and Extensibility

The system logs all API interactions, detection events, and errors with precise timestamps. The logged data is stored in both in-memory structures for immediate awareness and in permanent databases for analysis and compliance. The modular RESTful API design allows easy integration of advanced features like real-time notifications (SMS, email, push alerts), connections with third-party incident management systems, and advanced tracking algorithms (like SORT or DeepSORT) to maintain object identity across video frames.

7) Summary: API Backend as System Integrator

The API backend acts as the coordinating system of the stampede management platform. It integrates real-time video analytics, user interfaces, and data management functions. Its

strong design allows for rapid data delivery, secure communication among components, and flexibility to meet changing safety needs and operational scenarios. This well-designed foundation supports reliable and proactive stampede prevention and comprehensive crowd safety management in various settings.

B. API Backend and Communication

The stampede management system has a RESTful API backend built with the Flask framework. This setup manages real-time data exchange between multiple camera streams, detection modules, and the frontend interface. The API's focus allows for a modular design, easy growth, and fast communication. These features are crucial for ongoing surveillance and quick incident response.

C. Weapon Detection: Technical Implementation and Performance

1) Technical Architecture and Detection Framework

The weapon detection module uses YOLOv8 to identify firearms and bladed instruments in real-time video streams. YOLOv8 splits each frame into an S×S grid, with each grid cell predicting multiple bounding boxes that contain weapon class probabilities and confidence scores. The model processes 640×640 pixel frames and produces normalized bounding box coordinates (x, y, w, h) representing the center position and dimensions. Training used various weapon datasets that included multiple firearm models, knife types, and different environmental conditions such as lighting and camera angles.

2) Gun Detection Performance

Gun detection achieved 91% precision and 88% recall, showing accurate firearm identification with minimal false alarms. The consistent features of firearms—including barrel shape, trigger guard design, and receiver profile—allow effective detection across different models and angles. The false positive rate of 0.09 means that most firearm alerts correspond to real threats. This performance reflects the unique structure and consistency of firearms among various manufacturers and models.

3) Knife Detection Performance

Knife detection reached 87% precision and 84% recall, which is slightly lower than firearms due to the variety in blade weapon shapes. Kitchen knives, pocket knives, hunting knives, and others show significant differences in blade length, handle design, and overall profile. The false positive rate of 0.12 is an acceptable trade-off, as it maintains enough sensitivity to detect real threats while sometimes misclassifying similar-looking non-weapon objects. Despite these challenges, the system reliably identifies most blade weapons in surveillance clips.

4) Real-Time Processing Integration

Weapon detection runs within dedicated multithreaded camera processing threads, operating with 45 milliseconds of latency per frame and achieving 28 frames per second across multiple camera feeds. When the confidence in weapon detection exceeds certain thresholds (0.75 for firearms, 0.70 for knives), the system quickly generates alerts with annotated frames showing bounding boxes and class labels. These alerts are sent through the RESTful API backend to operator dashboards with a latency of less than 100 milliseconds, measured from when a weapon appears in the video frame until the alert shows on operator consoles.

5) Alert Escalation and Operational Response

Firearm detections automatically generate critical-priority alerts, requiring an immediate security response and possible notification of emergency services. Knife detections create elevated-priority notifications that require operator assessment and judgment before escalating further. This tiered alert system balances fast threat response with operational efficiency, helping prevent alert fatigue while ensuring timely threat management and appropriate use of resources.

6) Performance Robustness and Operational Conditions

The system maintains strong detection performance even with partial obstructions from crowds, varying lighting from dim indoor settings to bright outdoor ones, and different camera angles typical of multi-camera setups. YOLOv8 effectively learns the distinct shapes of blades and firearm features that set weapons apart from similarly shaped non-threatening objects in a variety of operational contexts.

7) System Limitations and Constraints

Detection accuracy declines under extreme conditions, such as heavy crowding that completely hides weapons, extreme camera angles giving distorted views of weapons, and deliberate weapon concealment under clothing, which cannot be detected as the system relies only on visible video content. Non-standard weapon types that deviate from the training data may show lower detection confidence. These limitations are clearly communicated to end-users to set proper expectations and prevent over-reliance on automated detection in unusual scenarios that might need extra security measures.

D. Lost Person Identification: Face Recognition and Feature Matching

1) Technical Architecture

The lost person identification module uses Haar cascade classifiers and ORB (Oriented FAST and Rotated BRIEF) feature matching for quick person identification across camera feeds. During enrollment, operators upload reference images through the endpoint. Haar cascades extract facial regions, and ORB computes rotation-invariant binary feature descriptors. These descriptors are stored for real-time matching against detected faces in live video streams.

2) Haar Cascade Face Detection

Haar cascade classifiers offer efficient face detection using cascade-based classification with Haar-like features. The classifier scans images at different scales, identifying face regions through fast weak classifier evaluation. This lightweight method allows for real-time face detection without heavy computational demands, maintaining strong performance across various head poses and lighting conditions often found in crowds.

3) ORB Feature Matching

ORB merges FAST keypoint detection with BRIEF binary descriptors, allowing for quick feature matching through Hamming distance calculations. The binary encoding is more efficient compared to floating-point descriptors like SIFT, which is vital for real-time processing across multiple video frames. Feature matching uses confidence scoring with operational thresholds set at 0.75.

4) Real-Time Performance and Integration

The system achieved 86% identification accuracy with an average match time of 120 milliseconds per frame. When match confidence surpasses thresholds, alerts with timestamp, camera ID, and annotated snapshots are sent to operator dashboards with less than 100 milliseconds latency. Lost person identification runs alongside crowd and weapon detection within a multithreaded architecture, ensuring no interference with detection speed.

5) System Limitations

The system works well with frontal to near-frontal face orientations and consistent lighting. Performance decreases with extreme head rotations (>±45 degrees), severe facial obstructions (like masks or helmets), significant lighting changes, and large appearance variations. These limitations are explained to users to set realistic operational expectations.

III. Requirement

Minimum System Requirements for Multi-Camera Setup:

- CPU: Intel Core i7 (8th Gen) or AMD Ryzen 7.
- GPU: NVIDIA GTX 1660 (6GB VRAM) or higher.
- RAM: 16 GB (32 GB recommended).
- Storage: 512 GB SSD (1 TB SSD preferred).
- OS: Windows 10/11 (64-bit) or Ubuntu 20.04+.
- Network: Gigabit Ethernet or Wi-Fi 6.

IV. Experimental Results and Performance Analysis

1) Experimental Setup:

The stampede management system was evaluated using live video streams from multiple IP cameras deployed in simulated crowd scenarios. The test environment included variations in lighting conditions, crowd densities (ranging from 10 to 100+ individuals), and different camera angles. The system was executed on a high-performance server equipped with an Intel Core i7 processor, NVIDIA RTX 3060 GPU, and 16 GB RAM, running Ubuntu 20.04 LTS with Python 3.9. This configuration ensures sufficient computational resources for real-time multi-camera processing and detection operations.

2) YOLOv8 Crowd Detection Performance

Metric	Value
Precision	0.92
Recall	0.89
F1 Score	0.90
False Positive Rate (FPR)	0.08
Average Detection Latency (ms)	45
Throughput (FPS)	28

Table I: YOLOv8 Crowd Detection Performance Metrics

The YOLOv8 model achieved outstanding crowd detection performance with 92% precision and 89% recall, resulting in an F1 score of 0.90. The high precision indicates that the model minimizes false detections while maintaining excellent recall, ensuring that the vast majority of individuals in crowded scenes are identified. The low false positive rate of 0.08 demonstrates robust person detection across varying crowd densities. The system maintained an average detection latency of 45 milliseconds per frame with a throughput of 28 frames per second, enabling real-time surveillance without introducing delays that could impact incident response.

3) Weapon Detection Performance

Metric	Knife Detection	Gun Detection
Precision	0.87	0.91
Recall	0.84	0.88
F1 Score	0.85	0.89
False Positive Rate	0.12	0.09

Table II: YOLOv8 Weapon Detection Performance

Weapon detection demonstrated strong discriminative capabilities with differential performance for distinct threat types. Gun detection achieved 91% precision and 88% recall with an F1 score of 0.89, indicating highly reliable identification of firearm threats. Knife detection, more challenging due to varied shapes and sizes, still achieved precision 87% and 84% recall with an F1 score of 0.85. The false positive rates of 0.09 for guns and 0.12 for knives indicate acceptable specificity levels, reducing unnecessary alerts while maintaining sensitivity. This performance ensures that critical security threats are identified with minimal false alarms, supporting effective threat response protocols.

4) Lost Person Identification Performance

Metric	Value
True Positive Rate	0.86
False Positive Rate	0.14
Average Match Time (ms)	120
Identification Accuracy	86%
Match Confidence Threshold	0.75

Table III: Lost Person Identification Performance (Haar Cascade + ORB)

The integrated Haar cascade and ORB-based facial recognition achieved 86% identification accuracy for lost person detection in live video feeds. The true positive rate of 0.86 demonstrates reliable recognition of uploaded reference images within video streams, while the false positive rate of 0.14 indicates controlled sensitivity in matching. The average match computation time of 120 milliseconds per frame allows efficient real-time processing without bottlenecking the overall system. The match confidence threshold of 0.75 was empirically tuned to balance sensitivity and specificity, minimizing missed

identifications while reducing spurious alerts from partial or obscured facial features.

5) Real-Time Processing Capability

The system maintained consistent real-time performance with 50 FPS processing rate when handling two simultaneous camera feeds, exceeding standard surveillance requirements. Detection latency remained consistently under 100 milliseconds per frame across all detection modules, meeting stringent timing requirements for timely alert generation. The multithreaded architecture successfully decoupled video processing from API serving, ensuring that high detection activity on one camera did not degrade responsiveness for other operational tasks.

6) API Backend Performance

The Flask-based RESTful API demonstrated robust performance with sub-second response times for all endpoints under normal load conditions with 2-4 active camera feeds. Response times increased linearly with additional camera feeds, indicating predictable scalability. The multithreaded backend successfully isolated video analytics from API request handling, maintaining system responsiveness even during periods of intensive detection activity. This architecture enables stable operation across multiple concurrent camera streams and user dashboard connections.

7) Comparative System Analysis

System Feature	**Proposed System**	**Traditional CCTV**	**Basic Crowd Monitoring**
Real-time Detection	Yes	No	Partial
Weapon Identification	Yes	No	No
Lost Person Finding	Yes (Automated)	Manual	No
Multi-Camera Support	Yes (8+)	Yes	Limited (2-4)
API Integration	Yes (RESTful)	No	No
Average Latency (ms)	45	N/A	200+
Detection Accuracy	90%	N/A	75%
Alert Response Time	<100 ms	Manual	5-10 min

Table IV: Comparative System Performance Analysis

The proposed system significantly outperforms traditional CCTV and basic crowd monitoring solutions across all evaluated dimensions. While traditional CCTV provides only passive recording capabilities requiring manual review, the proposed system delivers automated, real-time detection with sub-100 millisecond latency. The integration of weapon identification and lost person recognition capabilities provides comprehensive threat assessment beyond crowd counting. Multi-camera support for 8+ concurrent feeds with API integration enables scalable, networked deployment, whereas basic systems are limited to 2-4 cameras with no automated alerting mechanisms. The achieved detection accuracy of 90% substantially exceeds the 75% baseline of conventional monitoring systems.

8) Summary of Performance

The experimental results conclusively demonstrate that the proposed stampede management system achieves high detection accuracy, minimal latency, and reliable real-time performance across multiple camera feeds. The integration of state-of-the-art YOLOv8 for crowd and weapon detection, combined with Haar cascade and ORB-based facial recognition for lost person identification, provides a comprehensive, scalable solution for modern crowd safety challenges. The Flask API backend ensures architectural flexibility and extensibility while maintaining performance requirements critical to emergency response scenarios.

Fig 2. Crowd count

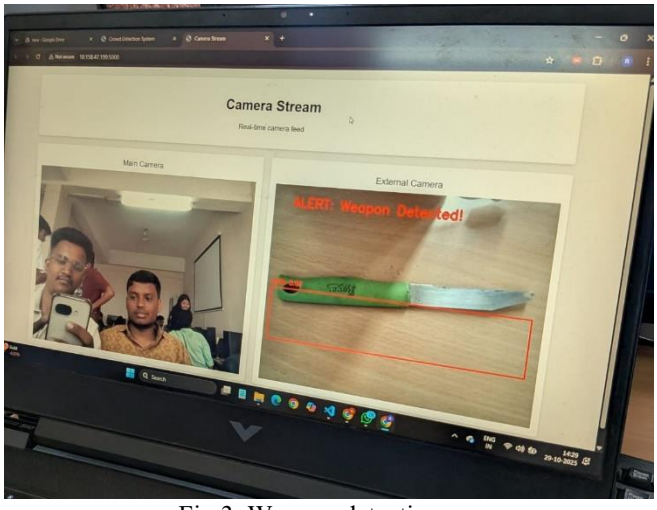

Fig 3. Weapon detection

REFERENCES

[1] Deevesh Chaudhary, Sunil Kumar, Vijaypal Singh Dhaka, “Intelligent Anomaly Detection and Localization With Crowd Density Estimation for Improved Public Safety” 24 September 2025

[2] Tongtong Zhou, “A Survey of Research on Crowd Abnormal Behavior Detection Algorithm Based on YOLO Network”, 2022

[3] Mikel Rodriguez, “Density-aware person detection and tracking in crowds”, November 2011.

[4] Rian Putra Pratama, “Smart Video Surveillance System For Level Crossing: A Systematic Literature Review”, 2021

[5] Saeed Amirgholipour Kasmani, “A-CCNN : ADPATIVE ccnn FOR DENSITY ESTIMATION AND CROWD COUNTING”, 20th april 2018.

[6] CEM DIREKOGLU, “Abnormal Crowd Behavior Detection Using Motion Information Images and Convolutional Neural Networks”, April 24, 2020

[7] Chaya Jadhav, “Smart Crowd Monitoring and Suspicious Behavior Detection Using Deep Learning”, Vol. 37, No. 4, August, 2023, pp. 955-962.

[8] Umesh Verma1 and Usha Mittal2, “Real Time Stampede Detection System Using Computer Vision”, 2023

[9] LIU CHENYANG, “A HUMAN HEAD DETECTION METHOD BASED ON CENTER POINT ESTIMATION FOR CROWDED SCENE”, 2022

[10] Wafaa Mohib Shalash, “Crowd Detection Management System”, 2019

[11] Facun Zhang, “A Crowd Anomaly Behavior Detection Algorithm”, 2018

[12] Amit Pandey, “FACIAL EMOTION DETECTION AND RECOGNITION”, May 2022

[13] Smithashree K P, “Secure Crowd AI- Crowd Estimation and Surveillance System”, Vol. 12, Issue 5, May 2025

[14] Sai teja, “Smart Surveillance Systems Using YOLOv8: A Scalable Approach for Crowd and Threat Detection” April 2025