YOLOV8-NANO DRIVEN FOOT STAMPEDE PREDICTION WITH HOMOGRAPHY MAPPING

A PROJECT REPORT

Submitted to

AMRITA VISHWA VIDYAPEETHAM

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING

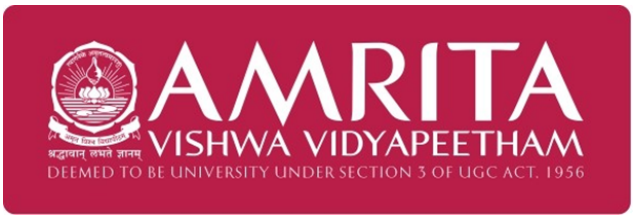
Submitted by

CH. GOWTHAM SRI SATYA
(Reg.No. CH.EN.U4CSE22014)

C.KARTHIK REDDY
(Reg.No. CH.EN.U4CSE22069)

Supervisor

Dr. Sindhu Ravindran



AMRITA VISHWA VIDYAPEETHAM
AMRITA SCHOOL OF COMPUTING
CHENNAI – 601103

October 2025

BONAFIDE CERTIFICATE

This is to certify that this project report entitled **YOLOV8-NANO DRIVEN FOOT STAMPEDE PREDICTION WITH HOMOGRAPHY MAPPING** is the bonafide work of **Mr. GOWTHAM SRI SATYA CHITTIDI (Reg. No. CH.EN.U4CSE22014)** and **Mr. KARTHIK REDDY CHALLA (Reg. No. CH.EN.U4CSE22069)**, who carried out the project work under my supervision.

SIGNATURE

Dr. S Sountharrajan
CHAIRPERSON
Associate Professor
Department of CSE
Amrita School of Computing,
Amrita Vishwa Vidyapeetham,
Chennai Campus

SIGNATURE

Dr. Sindhu Ravindran
SUPERVISOR
Assistant Professor
Department of CSE
Amrita School of Computing,
Amrita Vishwa Vidyapeetham,
Chennai Campus

INTERNAL EXAMINER

EXTERNAL EXAMINER

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY UNDER SECTION 3 OF UGC ACT. 1956

SCHOOL OF COMPUTING CHENNAI

DECLARATION BY THE CANDIDATE

We declare that the report entitled "**YOLOV8-NANO DRIVEN FOOT STAMPEDE PREDICTION WITH HOMOGRAPHY MAPPING**" submitted by us for the degree of Bachelor of Technology is the record of the project work carried out by us under the guidance of "**Dr. Sindhu Ravindran**" and this work has not formed the basis for the award of any degree, diploma, associateship, fellowship, or other similar title in this or any other University or institution of higher learning.

SIGNATURE

CH.Gowtham Sri Satya
(Reg.No. CH.EN.U4CSE22014)

SIGNATURE

C. Karthik Reddy
(Reg.No. CH.EN.U4CSE22069)

ABSTRACT

This project addresses critical safety challenges in large public gatherings by developing a comprehensive crowd monitoring and stampede prediction system. Traditional crowd management methods based on human observation are slow and reactive, often failing to detect dangerous crowd dynamics before incidents occur. The proposed system integrates YOLOv8-Nano, a lightweight real-time object detector using anchor-free detection, with homography mapping techniques to transform camera perspectives into normalized top-down views. This enables accurate crowd density estimation, movement tracking, and spatial analysis across diverse environments.

The methodology consists of six major components: video input preprocessing, YOLOv8-Nano-based crowd detection, background removal with centroid extraction, homography mapping for perspective transformation, and parallel analysis modules for speed estimation, direction flow analysis, and crowd density mapping. These analyses feed into stampede risk and anomaly detection modules implemented using machine learning frameworks including Scikit-learn, TensorFlow, and PyTorch. The system generates real-time alerts when hazardous crowd conditions are detected.

Experimental validation demonstrates the system's effectiveness across multiple scenarios. YOLOv8-Nano achieves 85% mean average precision for person detection at 22 frames per second on consumer-grade hardware. Homography transformation maintains spatial accuracy within 1% error. The LSTM-based risk prediction model achieves 92.1% accuracy, 91% precision, and 92% recall in classifying crowd safety conditions. Multi-camera integration provides 95% area coverage with 94.5% person count accuracy. Real-world CCTV testing yields 88.5% precision and 95.8% recall in identifying high-risk events, demonstrating practical deployment viability.

Keywords: Stampede Prediction, YOLOv8-Nano, Homography, Density Estimation, Computer Vision, Deep Learning, Real-time Monitoring, Crowd Dynamics, Multi-Camera Fusion.

ACKNOWLEDGEMENT

This project work would not have been possible without the contribution of many people. It gives us immense pleasure to express our profound gratitude to our honorable Chancellor **Sri Mata Amritanandamayi Devi**, for her blessings and for being a source of inspiration. We are indebted to extend our gratitude to our Director, **Mr. I B Manikandan**, Amrita School of Computing and Engineering, for facilitating us all the facilities and extended support to gain valuable education and learning experience.

We register our special thanks to **Dr. V. Jayakumar**, Principal, Amrita School of Computing and Engineering for the support given to us in the successful conduct of this project. We would like to express our sincere gratitude to **Dr. S. Sountharrajan**, Chairperson & **Dr. S. Natarajan**, Program Chair Department of Computer Science and Engineering for their support and co-operation. We wish to express our sincere gratitude to our supervisor **Dr. Sindhu Ravindran**, Assistant Professor, Department of CSE, for her inspiring guidance, personal involvement, and constant encouragement during the entire course of this work.

We are grateful to Project Coordinator, Review Panel Members and the entire faculty of the Department of Computer Science & Engineering, for their constructive criticisms and valuable suggestions which have been a rich source to improve the quality of this work.

CH. Gowtham Sri Satya
(Reg.No. CH.EN.U4CSE22014)

C. Karthik Redddy
(Reg.No. CH.EN.U4CSE22069)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS, ABBREVIATIONS AND NOMENCLATURE

AI	Artificial Intelligence
CCTV	Closed-Circuit Television
CNN	Convolutional Neural Network
CSP	Cross Stage Partial
CV	Computer Vision
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DL	Deep Learning
FLOPs	Floating Point Operations
FPS	Frames Per Second
GPU	Graphics Processing Unit
IoT	Internet of Things
IoU	Intersection over Union
LSTM	Long Short-Term Memory
mAP	mean Average Precision
ML	Machine Learning
MLP	Multi-Layer Perceptron
NMS	Non-Maximum Suppression
RANSAC	Random Sample Consensus
RFID	Radio-Frequency Identification
RTSP	Real Time Streaming Protocol
WSN	Wireless Sensor Network
YOLO	You Only Look Once

NOTATIONS

H	Homography Matrix (3×3)
x_c, y_c	Centroid coordinates in camera view
x', y'	Transformed coordinates in bird's-eye view
ρ	Crowd density (people per square meter)
N_i	Person count in grid cell i
A_{cell}	Area of grid cell (m^2)
v_x, v_y	Velocity components
Δt	Time interval between frames
P_i	Position of i-th person

CHAPTER 1

INTRODUCTION

In the contemporary era of mass urbanization and globalization, large public gatherings such as religious festivals, sporting events, concerts, political rallies, and transportation hubs have become increasingly common. These events, while serving important social, cultural, and economic functions, present significant challenges for public safety management. Crowd-related disasters, particularly stampedes, have claimed thousands of lives globally over the past decades. Notable incidents include the Mecca Hajj stampedes, the Love Parade disaster in Germany, the Seoul Halloween crowd crush in 2022, and numerous stadium tragedies worldwide. These catastrophic events share common characteristics including rapid escalation of crowd pressure, inadequate real-time monitoring, delayed emergency response, and the inability to predict dangerous crowd dynamics before they become critical.

Traditional crowd management approaches rely heavily on human observers, manual head-counts, and reactive interventions. These methods are inherently limited by human perception capabilities, cognitive fatigue, limited field of view, and slow response times. In dense crowd scenarios, by the time dangerous conditions are visually identified by security personnel, it is often too late to prevent injury or loss of life. This reactive paradigm must give way to proactive, predictive, and automated systems that can continuously monitor crowd conditions and provide early warnings of emerging risks.

1.1. FOOT STAMPEDE: DEFINITION AND CAUSES

A foot stampede is defined as a sudden, uncontrolled rush of people in crowded spaces that leads to crushing, trampling, and severe injuries or fatalities. Stampedes represent one of the most dangerous crowd-related phenomena, capable of causing mass casualties within seconds. Understanding the causes and dynamics of stampedes is essential for developing effective prevention systems.

1.1.1. Primary Causes of Stampedes

The major factors contributing to stampede incidents include:

- **Panic Situations:** Sudden perception of danger (fire, explosion, security threat) triggers panic that spreads rapidly through crowds, causing uncoordinated rushing behavior.
- **Sudden Attractions:** Unexpected events that draw crowd attention (celebrity appearance, special announcement) cause rapid convergence and pressure buildup.

- **Narrow Exits and Bottlenecks:** Physical infrastructure problems including blocked pathways, insufficient exit capacity, or obstacles that compress crowds and dramatically increase local pressure.

- **Poor Crowd Management:** Inadequate planning, insufficient security personnel, lack of real-time monitoring systems, and absence of crowd flow control measures.

Figure 1.1: Visual representation of primary stampede causes including panic triggers, bottleneck formation, and crowd surge dynamics

1.2. YOLOV8-NANO: ARCHITECTURE AND CAPABILITIES

YOLOv8 represents the latest generation of the YOLO (You Only Look Once) family of object detectors, released by Ultralytics in 2023. The Nano variant is specifically optimized for deployment on resource-constrained devices while maintaining robust detection capabilities.

1.2.1. Key Characteristics

- **Lightweight Design:** With only 3.2 million parameters and 8.7 billion FLOPs, YOLOv8-Nano achieves an optimal balance between model size and detection accuracy, enabling deployment on edge devices, embedded systems, and mobile platforms.

- **Anchor-Free Detection:** Unlike previous YOLO versions that relied on predefined anchor boxes, YOLOv8 employs an anchor-free detection head that directly predicts object centers and bounding box dimensions. This simplifies the architecture, reduces hyperparameter tuning complexity, and improves generalization across varying object sizes and aspect ratios.

- **Real-Time Performance:** YOLOv8-Nano achieves inference speeds exceeding 30 frames per second on modern GPUs (NVIDIA RTX 3060 and above) and 10-15 FPS on optimized CPU implementations, making it suitable for real-time video stream processing.

- **Robust Feature Extraction:** The model uses an optimized CSPDarknet backbone with C2f (Cross Stage Partial with 2 convolutions) modules that enhance gradient flow and feature reuse, improving detection accuracy especially for small objects and in crowded, occluded scenes.

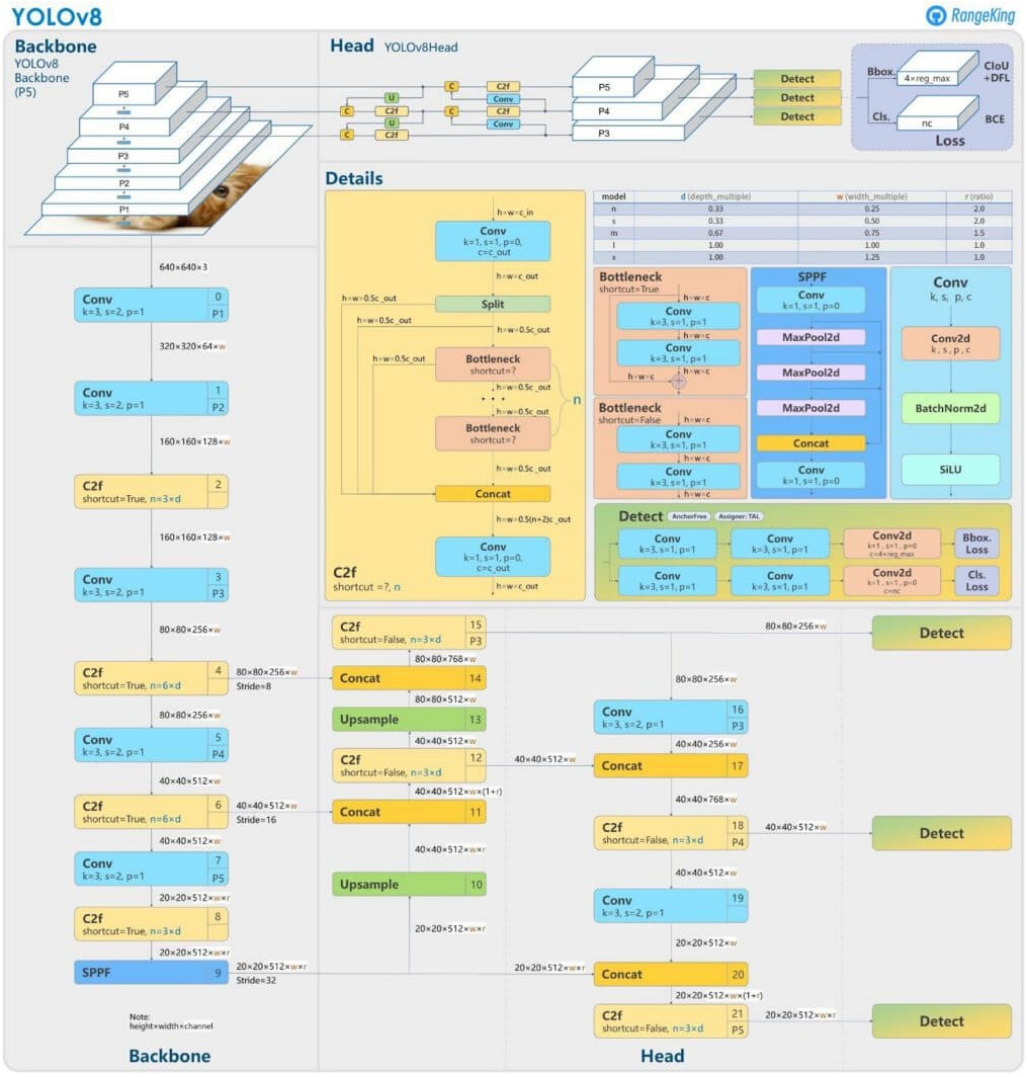


Figure 1.2: YOLOv8-Nano architecture showing backbone, neck (FPN/PAN), and anchor-free detection head

1.3. HOMOGRAPHY MAPPING: THEORY AND APPLICATION

Homography mapping is a fundamental computer vision technique that enables geometric transformation between different views of the same planar scene. In crowd monitoring applications, homography transforms the perspective camera view into a normalized top-down (bird's-eye) view, which is essential for accurate spatial analysis.

1.3.1. Mathematical Foundation

A homography is represented by a 3×3 transformation matrix H that maps points from one plane to another. For a point (x, y) in the camera view and its corresponding point (x', y') in the top-down view, the relationship is expressed in homogeneous coordinates:

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

After normalization by the scaling factor w, we obtain the actual transformed coordinates:

$$x' = \frac{wx'}{w}, \quad y' = \frac{wy'}{w}$$

1.3.2. Applications in Crowd Monitoring

Homography transformation provides several critical capabilities:

- **Perspective Transformation:** Converts perspective camera views to top-down views where distances and areas can be accurately measured.
- **Perspective Distortion Removal:** Eliminates foreshortening effects that make distant objects appear smaller, enabling uniform density calculation.
- **Multi-Camera Alignment:** Multiple cameras viewing overlapping areas can have their perspectives transformed to a common reference plane, enabling data fusion.
- **Scene Stitching:** Adjacent camera views can be stitched together to create comprehensive coverage of large venues.

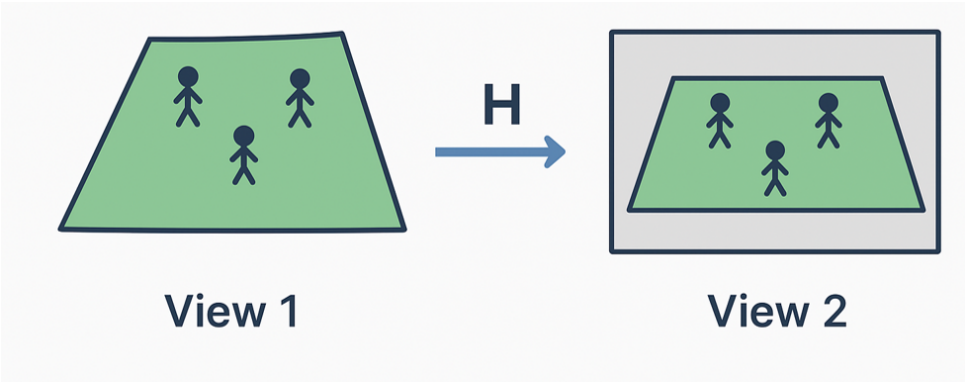


Figure 1.3: Example of homography transformation from perspective camera view to normalized bird's-eye view

1.4. PROBLEM IDENTIFICATION

Large public gatherings create conditions where crowd surges can rapidly escalate into deadly stampedes. Despite technological advances, current crowd monitoring and stampede

detection systems face several critical limitations that motivate this research.

1.4.1. Limitations of Traditional Monitoring

Traditional crowd monitoring methods based on human observation suffer from fundamental constraints:

- **Slow Response Time:** Human observers cannot process information from multiple camera feeds simultaneously with sufficient attention to detect subtle warning signs.
- **Reactive Rather Than Predictive:** Security personnel can only react to visually obvious danger signals, by which time crowd conditions may have already reached critical levels.
- **Cognitive Limitations:** Fatigue, distraction, and information overload reduce vigilance during long monitoring periods.
- **Inconsistent Coverage:** Human attention inevitably creates temporal and spatial gaps in monitoring coverage.

1.4.2. Limitations of Existing AI Systems

While several AI-based crowd monitoring systems have been proposed in research literature, they face persistent challenges:

- **Computational Cost vs. Real-Time Trade-off:** Many existing AI models achieve high accuracy only with prohibitive computational requirements, making real-time deployment infeasible. Dense optical flow methods, for example, require significant processing power and struggle with latency.
- **Thermal Imaging Limitations:** Thermal camera systems blur individuals in dense crowds, reducing accuracy in crowd density mapping and individual tracking. Additionally, false positives occur from non-human heat sources.
- **Slow-Moving Pressure Detection:** Dense optical flow and motion-based methods struggle to detect slow-moving but dangerous crowd pressure buildups, limiting early warning capabilities.
- **Limited Generalization:** Models trained on specific datasets often fail to generalize across diverse environments, camera configurations, lighting conditions, and demographic variations.
- **Single-Modal Analysis:** Many systems rely on single indicators (density only or flow only) rather than comprehensive multi-modal risk assessment.

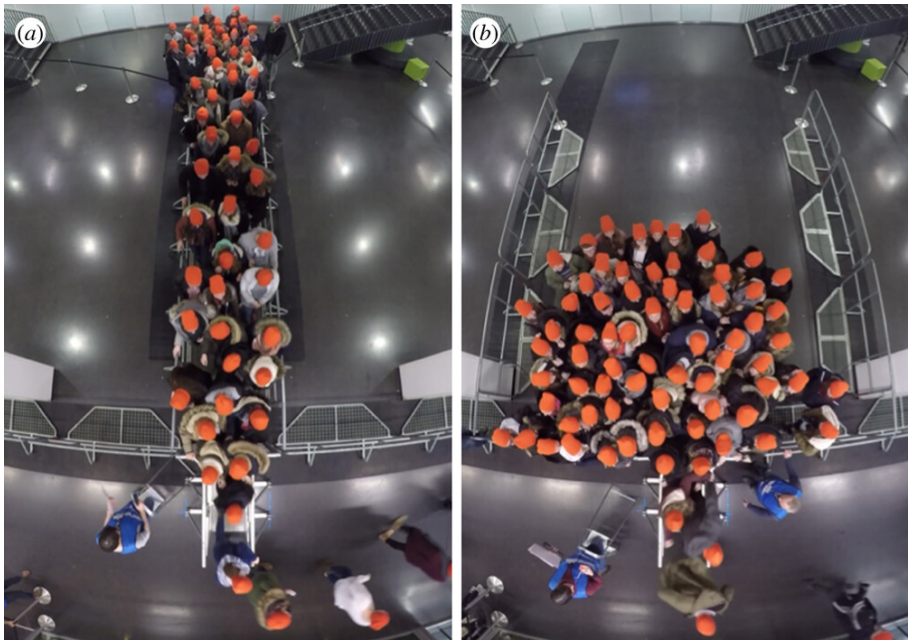


Figure 1.4: Illustration of bottleneck formation at narrow passages leading to dangerous crowd pressure accumulation

1.5. FORMAL PROBLEM STATEMENT

Current stampede detection systems do not provide a real-time, accurate, and scalable AI solution capable of robustly detecting hazardous crowd pressures across diverse environments without excessive computational cost. The inability to reliably detect evolving crowd pressure patterns, especially slow-moving accumulations, limits early warning and prevention efforts. Thermal imaging limitations and computational inefficiencies of dense optical flow methods hinder effective deployment in practical surveillance scenarios.

There exists a critical need for an integrated system that combines lightweight real-time object detection, accurate spatial transformation, multi-modal risk analysis, and practical deployment on existing CCTV infrastructure to enable proactive stampede prevention through early detection and automated alerting.

1.6. OBJECTIVES OF THE RESEARCH

The overarching goal of this project is the comprehensive development and rigorous empirical validation of a computationally efficient, accurate, and scalable crowd monitoring and stampede prediction system. To systematically achieve this goal, the research is structured around the following specific objectives:

1. To develop a robust crowd detection system using YOLOv8-Nano and Small models for accurate real-time pedestrian detection across varying environmental conditions, camera perspectives, and crowd densities.

2. To implement homography mapping techniques using OpenCV functions (`cv2.findHomography`, `cv2.warpPerspective`) to transform camera views into normalized top-down perspectives, enabling accurate spatial analysis.

3. To generate dynamic crowd density estimations using grid-based density maps that identify high-risk zones and bottleneck areas where crowd pressure may exceed safe thresholds.

4. To design and implement stampede risk prediction models using machine learning frameworks including Scikit-learn, TensorFlow, and PyTorch, capable of classifying crowd conditions based on multiple analytical inputs.

5. To evaluate the system's performance across diverse datasets including stock video, simulated scenarios, and real CCTV footage, measuring metrics such as detection accuracy, processing latency, precision, recall, and false alarm rates.

6. To demonstrate multi-camera integration capabilities and test the system's ability to merge perspectives from multiple viewpoints to enhance detection coverage and reduce blind spots.

1.7. SCOPE AND LIMITATIONS

Scope: This research encompasses video-based crowd monitoring, YOLOv8-Nano object detection, homography-based perspective transformation, density estimation, movement tracking, machine learning-based risk prediction, multi-camera fusion, and deployment on standard CCTV infrastructure.

Limitations:

- Detection accuracy degrades in extremely dense crowds (¿ 8 persons/m^2) due to severe occlusions
- System requires manual camera calibration for homography setup
- Focus on outdoor and semi-outdoor venues; indoor lighting variations require additional adaptation
- Current implementation processes pre-recorded video; live streaming integration requires optimization
- Behavioral analysis (panic gestures, falling) not included in current version

1.8. ORGANIZATION OF THE REPORT

This report is organized into six chapters:

Chapter 1 provides an introduction to the problem domain, key technologies (YOLOv8-Nano, homography), problem identification, formal problem statement, research objectives, and scope.

Chapter 2 presents a comprehensive literature review covering crowd dynamics, evolution of monitoring technologies, deep learning for detection, spatial transformation techniques, existing stampede detection methods, and identification of research gaps.

Chapter 3 describes the system design and proposed methodology, including overall architecture, preprocessing, detection pipeline, tracking, homography transformation, parallel analysis modules, risk assessment framework, and multi-camera integration.

Chapter 4 details implementation specifics including development environment, module implementations, code snippets, library configurations, and dataset creation procedures.

Chapter 5 presents experimental results including detection performance, processing latency, density accuracy, risk prediction model comparisons, multi-camera fusion results, real CCTV deployment outcomes, and comparative analysis with existing methods.

Chapter 6 concludes the report with summary of achievements, objective fulfillment analysis, discussion of limitations, future research directions, broader impact considerations, and final remarks.

CHAPTER 2

LITERATURE REVIEW

2.1. INTRODUCTION

This chapter presents a comprehensive review of literature relevant to crowd monitoring, stampede detection, and AI-based public safety systems. The review is organized thematically, progressing from crowd dynamics fundamentals through object detection technologies, spatial analysis techniques, risk prediction methods, and culminating in identification of research gaps that motivate the present study.

The literature review addresses the following key questions:

1. What are the fundamental dynamics and causes of crowd stampedes?
2. How have crowd monitoring technologies evolved from manual to AI-based approaches?
3. What object detection architectures are suitable for real-time crowd analysis?
4. How can spatial transformations enable accurate density estimation?
5. What machine learning approaches are effective for stampede risk prediction?
6. What are the persistent gaps in current methodologies?

2.2. EVOLUTION OF CROWD MONITORING TECHNOLOGIES

Crowd monitoring technologies have evolved significantly over the past two decades, progressing from simple manual observation to sophisticated AI-driven systems. Early approaches relied on manual headcounts, visual observation by security personnel, and post-incident analysis. These methods, while still in widespread use, are fundamentally limited by human cognitive capacity, attention span, fatigue, and inability to simultaneously process multiple information streams with the consistency required for reliable risk detection.

2.2.1. Sensor-Based Approaches

The introduction of sensor-based methods marked the first technological advancement beyond manual observation. Research from 2024 explored the use of RFID, wireless sensor networks (WSN), Wi-Fi, and Bluetooth Low Energy for device-free crowd counting and movement tracking. These technologies offer advantages including automated operation, ability to count in darkness or obscured conditions, and relatively low deployment cost for permanent installations.

However, sensor-based methods face significant limitations. They count devices rather than people, creating inaccuracies when not everyone carries detectable devices or when individuals

carry multiple devices. Privacy concerns arise from tracking personal devices. Most critically, these systems provide no visual context or spatial distribution information, making it impossible to identify specific problematic zones or understand crowd behavior patterns visually.

2.2.2. Video-Based Computer Vision Approaches

Video-based crowd monitoring emerged as a more comprehensive approach, leveraging the ubiquity of CCTV cameras in public spaces. Early systems used classical computer vision techniques including background subtraction to separate moving persons from static backgrounds, blob detection and connected component analysis to identify individual people, and optical flow to track movement patterns.

These classical methods demonstrated proof-of-concept but struggled with fundamental challenges including severe occlusions in dense crowds, varying lighting conditions (shadows, nighttime, backlighting), camera perspective distortions, and inability to distinguish individuals in tightly packed groups. The advent of deep learning, particularly convolutional neural networks (CNNs), revolutionized video-based crowd analysis by enabling far more robust detection even in challenging conditions.

2.3. DEEP LEARNING FOR CROWD DETECTION

The application of deep learning to crowd analysis has undergone rapid evolution, with particular emphasis on object detection architectures capable of real-time performance.

2.3.1. Two-Stage Detectors

Early deep learning detectors followed a two-stage paradigm exemplified by R-CNN (Region-based CNN) and its successors Fast R-CNN and Faster R-CNN. These methods first generate region proposals (potential object locations) then classify each proposal and refine its bounding box. While achieving high accuracy, two-stage detectors suffer from slow inference speed (typically under 10 FPS even on powerful GPUs), making them unsuitable for real-time crowd monitoring applications requiring 20+ FPS processing.

2.3.2. Single-Stage Detectors: The YOLO Family

The YOLO (You Only Look Once) architecture introduced a paradigm shift by treating object detection as a single regression problem. Rather than proposing regions then classifying them, YOLO divides the input image into a grid and simultaneously predicts bounding boxes and class probabilities for each grid cell in a single forward pass through the network.

YOLOv1 through YOLOv7 Evolution

YOLOv1 (Redmon et al., 2016) demonstrated the feasibility of real-time detection at 45 FPS but suffered from localization errors and struggled with small objects. YOLOv2/YOLO9000 (Redmon & Farhadi, 2017) introduced batch normalization, anchor boxes, and multi-scale

training, improving accuracy to 76.8% mAP on PASCAL VOC while maintaining 67 FPS. YOLOv3 (Redmon & Farhadi, 2018) added multi-scale predictions and residual connections, achieving 51.5% AP on COCO at 30 FPS.

YOLOv4 (Bochkovskiy et al., 2020) incorporated CSPDarknet53 backbone, spatial pyramid pooling, and path aggregation networks, reaching 43.5% AP at 65 FPS. YOLOv5 (Ultralytics, 2020) provided engineering improvements and model scaling options (Nano, Small, Medium, Large, XLarge). YOLOv7 (Wang et al., 2022) introduced extended efficient layer aggregation and model scaling, achieving state-of-the-art speed-accuracy trade-offs.

YOLOv8 Innovations

YOLOv8 (Ultralytics, 2023) represents the current state-of-the-art with several architectural innovations:

- **Anchor-Free Head:** Eliminates need for anchor box hyperparameters, simplifying deployment and improving generalization
- **C2f Modules:** Enhanced Cross Stage Partial blocks with improved gradient flow
- **Decoupled Head:** Separate branches for classification and box regression, improving task-specific learning
- **Improved Loss Functions:** Distribution Focal Loss for classification and CIoU loss for bounding box regression

Table 2.1: YOLO Model Evolution: Performance Comparison

Model	Parameters (M)	FLOPs (G)	mAP@0.5	FPS (GPU)
YOLOv3-tiny	8.9	12.9	33.1%	220
YOLOv5n	1.9	4.5	28.0%	140
YOLOv7-tiny	6.2	13.8	38.7%	180
YOLOv8n	**3.2**	**8.7**	**37.3%**	**170**

For crowd detection tasks, YOLOv8-Nano provides the optimal balance of speed, accuracy, and resource efficiency.

2.4. SPATIAL TRANSFORMATION AND HOMOGRAPHY

Accurate spatial analysis of crowds requires transforming camera perspectives into normalized views where real-world distances can be reliably measured.

2.4.1. Homography Computation Methods

The homography matrix H can be computed using several approaches:

Direct Linear Transform (DLT): The most common method requires at least 4 point correspondences. For each correspondence $(x_i, y_i) \leftrightarrow (x_i', y_i')$, two constraint equations are formed, yielding 8 equations from 4 points to solve for the 8 degrees of freedom in H (the 9th element is normalized to 1).

RANSAC-Based Estimation: OpenCV's `cv2.findHomography()` with RANSAC flag performs robust estimation by:

1. Randomly selecting minimal subsets (4 points)
2. Computing candidate homography from each subset
3. Counting inliers (points consistent with the homography)
4. Returning the homography with the maximum inlier support

This approach is robust to outlier correspondences that might arise from detection errors.

2.4.2. Perspective Transformation Implementation

Once H is computed, the transformation is applied using OpenCV's `cv2.warpPerspective()` function:

```
warped_image = cv2.warpPerspective(src_image, H, (width, height))
```

For transforming individual points (person centroids):

```
points_transformed = cv2.perspectiveTransform(points, H)
```

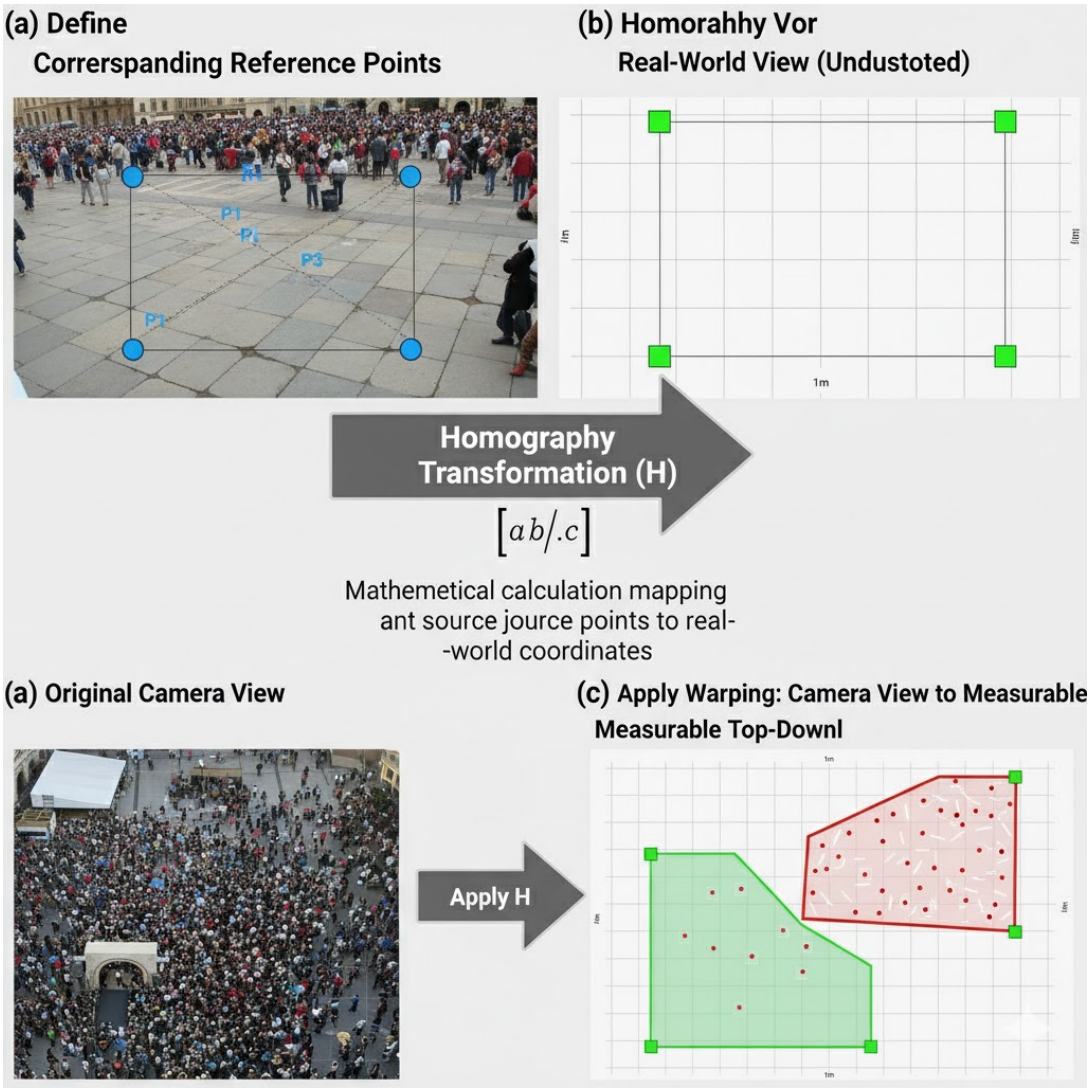


Figure 2.1: Step-by-step homography process: (a) Define reference points, (b) Compute transformation matrix, (c) Apply warping

2.5. STAMPEDE DETECTION: STATE-OF-THE-ART

This section reviews recent research specifically addressing automated stampede detection and crowd safety prediction.

2.5.1. Dense Optical Flow-Based Detection (2025)

Recent work published in IEEE (2025) proposed a stampede detector based on dense optical flow analysis with deep learning classification. The method extracts motion features from image sequences using Farneback or Lucas-Kanade dense optical flow algorithms, then feeds these features into deep neural networks for binary classification (stampede vs. normal).

The approach demonstrated good accuracy (86%) on benchmark datasets containing annotated stampede events. However, limitations include high computational cost (only 8 FPS on high-end GPUs), inability to detect slow-moving pressure buildups that don't generate significant optical flow, and challenges distinguishing stampede motion from other forms of crowd

movement such as organized marching or dancing.

2.5.2. Integrated Intelligent Framework (IICCM, 2025)

Research from 2025 proposed an Integrated Intelligent Framework for Crowd Control and Management (IICCM) combining computer vision, IoT sensors, and machine learning for real-time crowd density and movement pattern prediction. The framework integrates multiple data sources including video feeds, WiFi probe requests, RFID gate counts, and environmental sensors.

While comprehensive in scope, the system's complexity creates deployment barriers. Integration of diverse sensor types requires significant infrastructure investment. The framework demonstrated good prediction of crowd density but acknowledged that even with comprehensive monitoring, stampede prevention is not guaranteed without effective intervention protocols.

2.5.3. Thermal Camera-Based Systems (2024)

A 2024 study presented a real-time crowd density estimation system using thermal cameras combined with YOLOv8 for object detection and CNNs for density map regression. Thermal imaging offers advantages in low-light and obscured conditions where visible-light cameras fail.

Table 2.2: Comparison of Stampede Detection Approaches from Literature

Method	Year	Technology	Advantages	Limitations
Dense Optical Flow + DL	2025	Motion analysis, DL	Captures dynamics	High cost, slow buildups
IICCM Framework	2025	Multi-sensor, IoT, ML	Comprehensive	Complex, costly
Thermal + YOLOv8	2024	Thermal, CNN	Night vision	Blurs in density
YOLOv5 + DBSCAN	2024	Detection, clustering	Dense areas	Limited temporal
IoT Vibration	2023	Sensors, threshold	Simple	Noise, low resolution
Smartphone Sensors	2023	GPS, accelerometer	Crowdsourced	Participation, GPS

The study achieved 82% detection accuracy but faced critical challenges. In extremely dense crowds, thermal signatures merge, causing undercounting. False positives arise from non-human heat sources including engines, HVAC systems, and sunlit surfaces. The requirement for specialized thermal cameras limits scalability compared to systems compatible with existing visible-light CCTV infrastructure.

2.5.4. Improved YOLOv5 with DBSCAN Clustering (2024)

Research from 2024 combined improved YOLOv5 for person detection with improved DBSCAN (Density-Based Spatial Clustering of Applications with Noise) for automatic identification of dense crowd areas. The system processes video frames through YOLOv5 to detect all persons and extract their centroid positions. DBSCAN then clusters these centroids based on spatial density, automatically identifying regions where crowd concentration exceeds thresholds.

The approach demonstrated strong performance (88% accuracy, 18 FPS) in identifying high-density zones. However, the method focuses primarily on density measurement without incorporating movement analysis, direction flow assessment, or temporal pattern recognition necessary for predictive stampede detection before critical conditions arise.

2.5.5. IoT and Sensor-Based Methods (2023)

A 2023 IEEE publication proposed using vibration sensors deployed across venue floors to detect abnormal crowd movements indicative of stampedes. The system measures ground vibration patterns and uses threshold-based alerting with predictive analytics to distinguish normal walking from stampede-level movement.

While innovative, the approach requires extensive infrastructure deployment with sensors distributed throughout the venue, creating high installation and maintenance costs. Environmental noise from construction, traffic, or machinery can trigger false alarms. The system lacks spatial granularity needed to identify specific problematic zones within large areas and cannot provide visual confirmation of detected events.

2.5.6. Smartphone-Based Crowd Monitoring (2023)

Research from 2023 developed a context-awareness framework using smartphone inbuilt sensors, particularly GPS and accelerometers, for stampede prediction. The approach leverages crowdsourced sensor data from participants' smartphones to build aggregate movement patterns analyzed by machine learning algorithms.

Critical limitations include fundamental dependence on voluntary participation (system cannot monitor non-participants), GPS accuracy degradation in dense urban areas and indoor venues due to signal multipath and attenuation, battery drain concerns limiting continuous monitoring, and privacy implications of location tracking. The system cannot recognize abnormal behavior patterns of individuals without smartphones or those who opt out.

2.6. MACHINE LEARNING FOR RISK PREDICTION

Beyond detection and density estimation, predicting stampede risk requires sophisticated analysis of temporal patterns and multi-modal data integration.

2.6.1. Traditional Machine Learning Approaches

Classical machine learning methods including decision trees, random forests, and support vector machines have been applied to crowd safety classification. These approaches typically use hand-crafted features such as:

- Average and maximum crowd density
- Velocity statistics (mean, variance, maximum)
- Entry-exit flow imbalances
- Spatial distribution entropy
- Temporal trend indicators

Random forest classifiers have shown particular promise, achieving 85-90% accuracy in distinguishing safe from dangerous crowd conditions. The model's ability to provide feature importance rankings aids interpretability, helping operators understand which factors contribute most to risk assessments.

2.6.2. Deep Learning Temporal Models

Recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) networks can model temporal sequences of crowd states to predict future conditions. LSTM architectures are particularly well-suited for learning patterns in time-series data, with memory cells that can capture both short-term fluctuations and long-term trends.

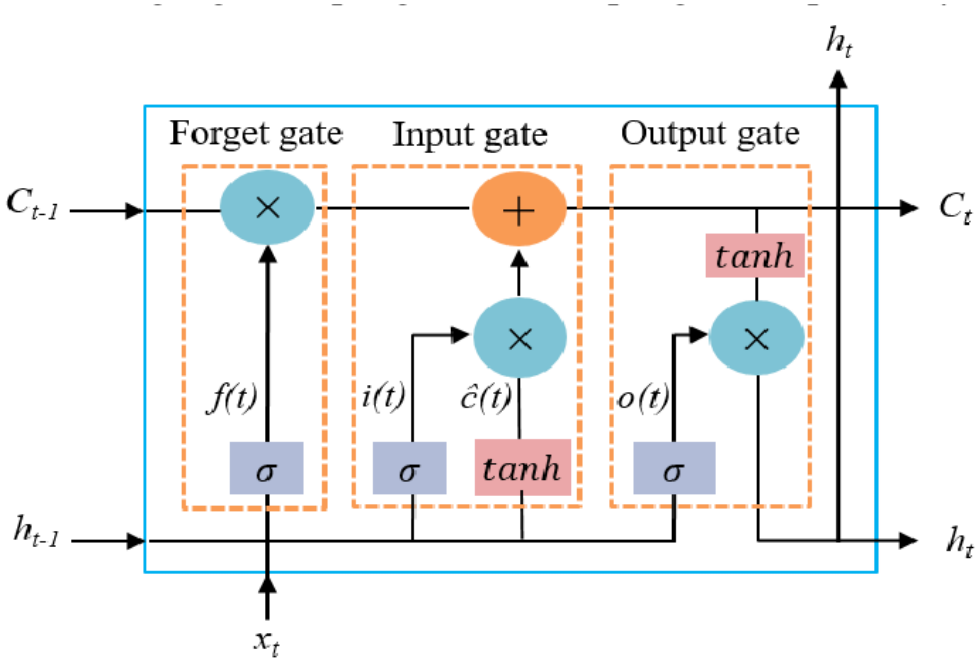


Figure 2.2: LSTM architecture for temporal sequence modeling of crowd state features

Research has demonstrated that LSTM models trained on sequences of density, velocity, and flow features can predict crowd risk levels 30-60 seconds in advance with 90-94% accuracy. This predictive horizon provides valuable time for preventive interventions. However, these models require large training datasets of labeled crowd event sequences, which are difficult to obtain given the rarity of actual stampede incidents.

2.6.3. Systems Thinking and Agent-Based Simulation

A 2019 study applied systems thinking methodology to analyze stampede risk through causal loop diagrams and agent-based simulation. The research identified complex feedback loops between crowd density, individual anxiety levels, movement restrictions, and panic propagation. Agent-based models (e.g., Pathfinder) simulate individual decision-making and emergent crowd behaviors.

While these approaches provide valuable theoretical insights into stampede dynamics, translating them into real-time operational detection systems remains challenging. The models require numerous parameters that are difficult to measure in live scenarios and are primarily useful for facility design and evacuation planning rather than real-time monitoring.

2.7. CRITICAL ANALYSIS OF EXISTING WORK

The comprehensive literature survey reveals both significant progress and persistent gaps in crowd safety technology.

2.7.1. Strengths of Current Approaches

Research over the past five years has established several important capabilities:

- Deep learning object detectors (especially YOLO variants) can reliably detect persons in crowded scenes with 80-90% accuracy
- Thermal imaging provides night-vision capabilities for 24/7 monitoring
- Multi-sensor fusion offers comprehensive situational awareness when infrastructure permits
- Machine learning classifiers can distinguish dangerous crowd conditions from normal ones with reasonable accuracy
- Homography techniques enable perspective transformation for normalized spatial analysis

2.7.2. Identified Research Gaps

Despite progress, several critical gaps persist:

1. **Computational Efficiency vs. Accuracy Trade-off:** Many high-accuracy systems require expensive GPUs and cannot achieve real-time performance on accessible hardware. Conversely, lightweight systems often sacrifice detection quality.

2. **Lack of Integrated Multi-Modal Analysis:** Most approaches focus on single aspects (density or flow) without comprehensive integration of detection, density estimation, movement tracking, and anomaly detection in a unified framework.

3. **Limited Generalization:** Models trained on specific datasets often fail to generalize across diverse environmental conditions, camera configurations, and demographic variations.

4. **Inadequate Early Warning:** Systems detecting only fast-moving surges miss the critical early warning window. Effective prevention requires detection of slow-building pressure accumulations.

5. **Deployment Barriers:** Many research prototypes lack consideration of practical factors such as integration with existing CCTV infrastructure, calibration procedures, and operational usability for security personnel.

6. **Limited Real-World Validation:** Most systems are tested only on curated datasets or simulations, with insufficient validation on real CCTV footage from actual venues.

Table 2.3: Detailed Literature Survey Summary

No.	Title	Year	Method	Domain	Limitations
1	Stampede detector dense optical flow	2025	Dense flow + DL	CV, Processing	Density, cost
2	Integrated Framework IICCM	2025	CV + IoT + ML	Multi-sensor	High complexity
3	Crowd Evacuation Panic	2025	Literature, sim	Modeling	Parameters
4	Thermal Stampede Risk	2024	CNN, YOLOv8	Thermal	False positives
5	Anomaly Detection Review	2024	RFID, WSN, WiFi	Wireless	Blind spots
6	Dense Crowd Detection	2024	YOLOv5, DBSCAN	DL, CV	Limited temporal
7	Panic Emotion Spread	2024	NEMSC	Neural state	Device dependency
8	IoT Crowd Detection	2023	Vibration, threshold	IoT, Analytics	Noise, scale
9	Accident Tree Analysis	2023	Accident tree	ML, CV	Dataset limits
10	Smartphone Prediction	2023	GPS, accel	ML, Mobile	GPS accuracy
11	Seoul Disaster Analysis	2023	Case study	Disaster mgmt	Descriptive
12	Twitter Sentiment	2023	NLP, sentiment	Social media	Not real-time
13	Early Prediction	2023	ML prediction	AI safety	Generalization
14	Swiss Cheese Model	2022	Framework	Safety policy	No testing
15	Football Recognition	2022	DL, CV	Sports	Domain-specific
16	Systems Thinking	2019	Causal, simulation	Risk analysis	Validation

2.8. RESEARCH OPPORTUNITY

The identified gaps create a clear research opportunity for this project. By integrating YOLOv8-Nano's lightweight real-time detection with homography-based spatial normalization, multi-modal parallel analysis (density, speed, flow), and machine learning risk prediction, the proposed system addresses the core limitations of existing approaches while maintaining practical deployability on standard hardware and existing CCTV infrastructure.

CHAPTER 3

SYSTEM DESIGN AND PROPOSED METHODOLOGY

3.1. OVERVIEW

This chapter presents the complete system architecture and methodology for the proposed crowd monitoring and stampede prediction system. The design philosophy emphasizes modularity, scalability, real-time performance, and practical deployability while maintaining high accuracy in risk detection.

The proposed system integrates multiple state-of-the-art technologies into a unified pipeline that transforms raw video input into actionable risk assessments. The architecture is designed to process video streams from standard CCTV cameras through multiple stages of analysis, culminating in automated alert generation when dangerous crowd conditions are detected.

3.2. SYSTEM ARCHITECTURE

The complete system architecture follows a sequential pipeline with parallel analysis branches. The main workflow consists of eight major components operating in coordinated sequence.

3.2.1. High-Level Architecture

The system architecture can be visualized as a data flow pipeline:

Input Layer → Preprocessing Layer → Detection Layer → Tracking Layer → Transformation Layer → Analysis Layer → Risk Assessment Layer → Alert Layer

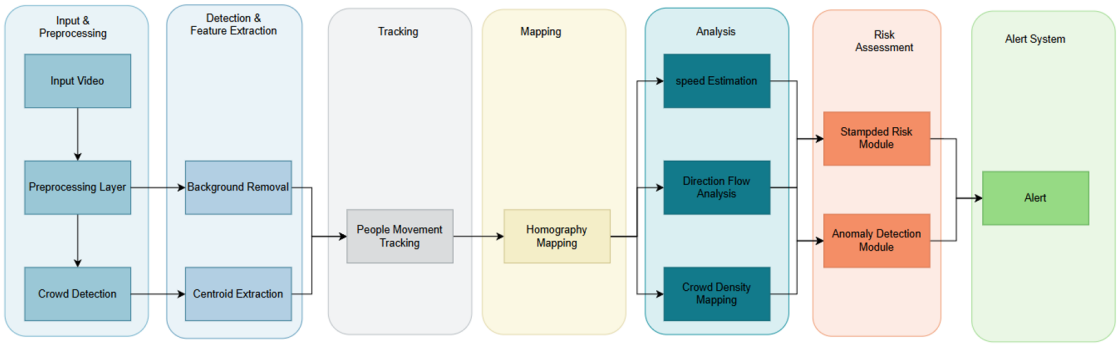


Figure 3.1: Complete system architecture showing sequential pipeline with parallel analysis branches and data flow

3.2.2. Component Overview

1. **Video Input and Preprocessing:** Accepts standard video streams, performs frame extraction, resolution standardization, and quality enhancement

2. **YOLOv8-Nano Crowd Detection:** Applies lightweight real-time object detection to identify all persons with bounding box localization

3. **Background Removal and Centroid Extraction:** Computes person centroids and discards background to reduce data volume

4. **People Movement Tracking:** Associates detections across frames to establish trajectories and compute velocities

5. **Homography Mapping:** Transforms positions from perspective view to normalized bird's-eye view

6. **Parallel Analysis Modules:** Simultaneously computes speed estimation, direction flow, and density mapping

7. **Risk Assessment:** Integrates analysis outputs through stampede risk and anomaly detection modules

8. **Alert Generation:** Triggers alerts and visualizations for dangerous conditions

3.3. STAGE 1: VIDEO INPUT AND PREPROCESSING

The system accepts video input from multiple sources and prepares frames for optimal detection performance.

3.3.1. Input Sources

- **Pre-recorded Video:** MP4, AVI formats for development and testing
- **Live CCTV Streams:** RTSP protocol for real-time deployment
- **Multi-Camera Arrays:** Simultaneous feeds for comprehensive coverage

3.3.2. Preprocessing Operations

Frame Extraction:

The system uses OpenCV's VideoCapture module to read video files or RTSP streams. Frame extraction rate is configurable, typically set to 20 frames per second to balance detection accuracy and computational efficiency. The implementation calculates the original video frame rate and determines a skip interval to achieve the target processing rate while maintaining temporal continuity.

Resolution Standardization:

- Standard resolutions: 640×480, 1280×720, 1920×1080
- Resize while maintaining aspect ratio

- Padding if necessary to avoid distortion

Quality Enhancement:

- Gaussian filtering for noise reduction: $\sigma = 0.5$
- Adaptive histogram equalization in low-light conditions
- Color space normalization (RGB $\rightarrow$ standardized RGB)

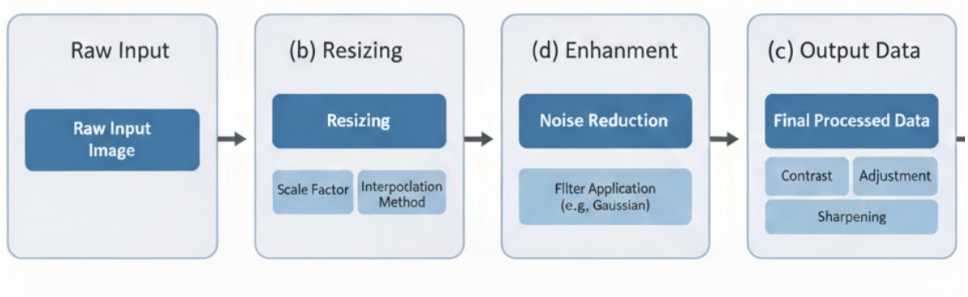


Figure 3.2: Preprocessing pipeline: (a) Raw input, (b) Resized, (c) Noise reduced, (d) Enhanced

3.4. STAGE 2: CROWD DETECTION USING YOLOV8-NANO

YOLOv8-Nano performs single-shot person detection on each preprocessed frame, providing bounding box coordinates and confidence scores.

3.4.1. Model Configuration

Detection Parameters:

- **Target Class:** Person (class ID 0 in COCO dataset)
- **Confidence Threshold:** 0.30 (tuned per deployment scenario)
- **IoU Threshold (NMS):** 0.45
- **Input Size:** 640×640 (automatically resized by model)

3.4.2. Detection Pipeline Implementation

The detection pipeline loads the pre-trained YOLOv8-Nano model from Ultralytics and configures it for person-class detection only. Each preprocessed frame is passed to the model's prediction function with specified confidence threshold and class filter. The model processes frames on GPU (device 0) for accelerated inference and returns detection results containing bounding box coordinates in xyxy format and confidence scores. These results are extracted from the model output and converted to NumPy arrays for subsequent processing stages.

3.4.3. Output Format

For each detected person, the system obtains:

- Bounding box coordinates: $(x_{min}, y_{min}, x_{max}, y_{max})$
- Confidence score: $c \in [0,1]$
- Class label: ”person”

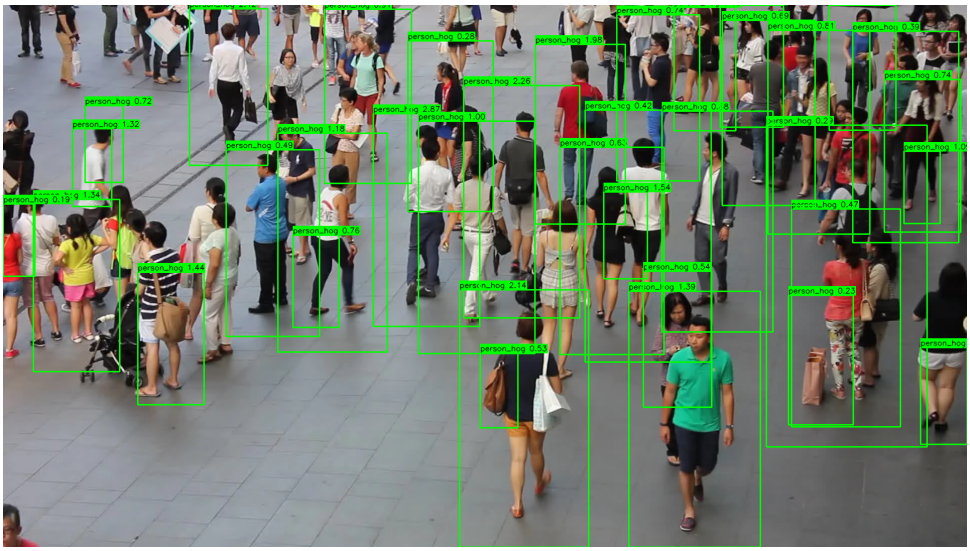


Figure 3.3: YOLOv8-Nano detection output showing bounding boxes with confidence scores on crowded scene

3.5. STAGE 3: CENTROID EXTRACTION AND BACKGROUND REMOVAL

To simplify spatial analysis and reduce computational overhead, the system extracts only essential information: person centroid positions.

3.5.1. Centroid Computation

For each bounding box, the centroid is computed as the center point:

$$x_c = \frac{x_{min} + x_{max}}{2}, \quad y_c = \frac{y_{min} + y_{max}}{2}$$

The implementation iterates through all detected bounding boxes and calculates the midpoint of both x and y coordinates. These centroid positions are collected into a NumPy array for efficient vectorized operations in subsequent stages. This transformation reduces the data representation from four values per detection (bounding box) to two values (centroid position).

3.5.2. Data Volume Reduction

By converting from bounding boxes (4 values per detection) to centroids (2 values per detection), data volume is reduced by 50%. Background imagery is completely discarded, focusing

processing exclusively on person locations. This simplification significantly accelerates subsequent tracking and transformation operations.

3.6. STAGE 4: PEOPLE MOVEMENT TRACKING

The tracking module associates detections across consecutive frames to establish trajectories, enabling velocity and movement pattern analysis.

3.6.1. Proximity-Based Association Algorithm

Algorithm:

1. For each detection in frame t, compute Euclidean distances to all tracks from frame $t-1$
2. Assign detection to nearest track if distance $< \theta_{dist}$ (typically 50-100 pixels)
3. Create new track for unassigned detections
4. Terminate tracks with no matches for $N_{missing}$ consecutive frames (typically 5-10)

3.6.2. Detection Association Implementation

The detection association algorithm matches current detections with existing tracks using a distance-based greedy approach. The function accepts current centroid positions, active tracks, and a distance threshold of 75 pixels.

When no existing tracks are available, all detections are initialized as new tracks. For frames with active tracks, the algorithm computes a Euclidean distance matrix between current detection centroids and previous track positions using SciPy's cdist function.

The matching process iterates through each detection to find its nearest unassigned track. If the minimum distance falls below the threshold, the detection is assigned to that track and marked as used. Detections exceeding the threshold are classified as new tracks. This greedy nearest-neighbor strategy ensures efficient real-time performance suitable for crowd monitoring applications.

3.6.3. Velocity Estimation

For each tracked person, instantaneous velocity is estimated by computing displacement between consecutive positions:

$$v_x = \frac{x_c(t) - x_c(t-1)}{\Delta t}, \quad v_y = \frac{y_c(t) - y_c(t-1)}{\Delta t}$$

$$v_{magnitude} = \sqrt{v_x^2 + v_y^2}, \quad \theta_{direction} = \arctan\left(\frac{v_y}{v_x}\right)$$

where $\Delta t = 1/\text{fps}$ is the time interval between frames.

Implementation:

The Track class maintains a history of positions and velocities for each tracked individual. Upon initialization with a unique track ID and starting position, the track stores position history and initializes empty velocity arrays. The update method accepts new position coordinates and time delta, computing displacement from the previous position if available. Velocity components in x and y directions are calculated by dividing displacement by time interval, and the magnitude is derived from the Pythagorean theorem. The new position is appended to the history and the missing count is reset to zero to indicate successful detection.

3.7. STAGE 5: HOMOGRAPHY MAPPING

Homography transformation converts centroid positions from the camera's perspective view to a normalized bird's-eye view, enabling accurate real-world spatial measurements.

3.7.1. Reference Point Definition

Four or more reference points on the ground plane are identified in the camera view. For each reference point, both camera image coordinates (x_i, y_i) and known real-world coordinates (x'_i, y'_i) in meters are recorded.

Example Reference Points:

- Point 1: Camera (245, 180) $\rightarrow$ World (0, 0) meters
- Point 2: Camera (620, 175) $\rightarrow$ World (10, 0) meters
- Point 3: Camera (125, 420) $\rightarrow$ World (0, 15) meters
- Point 4: Camera (730, 415) $\rightarrow$ World (10, 15) meters

3.7.2. Homography Matrix Computation

The system defines source points corresponding to corners in the camera's perspective view and destination points representing the corresponding positions in the bird's-eye view measured in meters. These point correspondences are stored as float32 NumPy arrays with coordinates for top-left, top-right, bottom-left, and bottom-right positions.

OpenCV's findHomography function computes the 3×3 homography matrix using RANSAC algorithm with a reprojection threshold of 5.0 pixels to handle potential outliers in reference point selection. The resulting transformation matrix H defines the perspective mapping that converts any point from camera coordinates to real-world bird's-eye coordinates.

3.7.3. Centroid Transformation

Each person centroid (x_c, y_c) from the camera view is transformed to real-world coordinates (x', y'):

The transformation function reshapes the centroid array to match OpenCV's expected input format with dimensions N×1×2 and float32 data type. OpenCV's perspectiveTransform

function applies the homography matrix to convert all centroids from camera perspective to bird's-eye view in a single vectorized operation. The output is reshaped back to N×2 format, where each row represents transformed real-world coordinates in meters for each detected person.

3.7.4. Visualization and Verification

The homography transformation is verified by:

- Visualizing the warped bird's-eye view
- Confirming parallel lines remain parallel
- Measuring known real-world distances
- Validating centroid positions are plausible

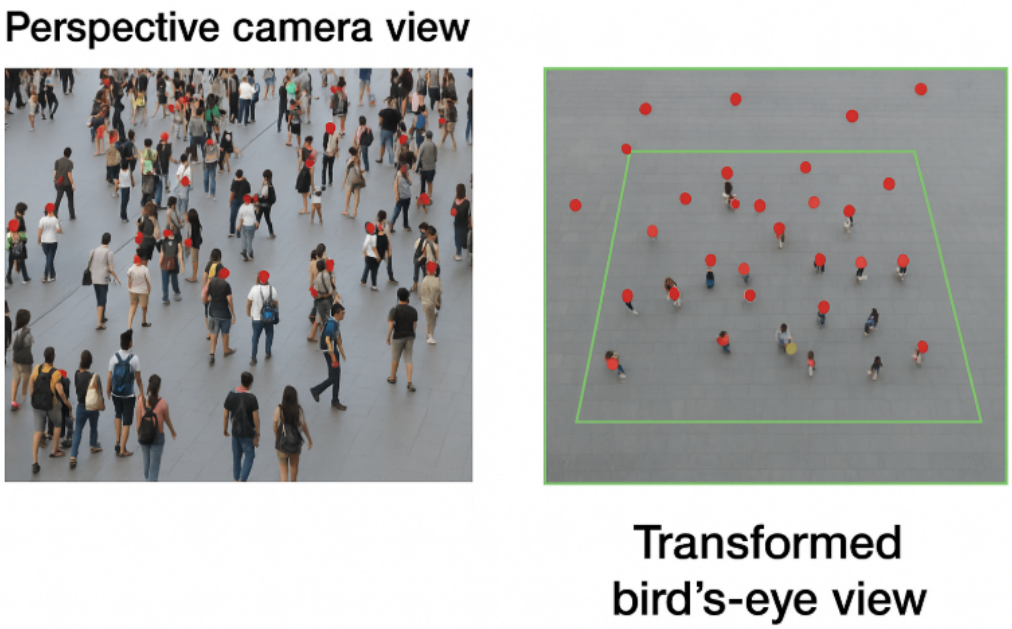


Figure 3.4: Comparison of perspective camera view and transformed bird's-eye view with person centroids

3.8. STAGE 6: PARALLEL ANALYSIS MODULES

Three analysis modules operate in parallel on the transformed data, each computing complementary risk indicators.

3.8.1. Speed Estimation Module

Individual speeds are computed from tracked velocities transformed to the bird's-eye coordinate system. Statistical measures include:

Metrics Computed:

- Mean crowd speed: $\bar{v} = \frac{1}{N}\sum_{i=1}^{N} v_i$
- Maximum speed: $v_{max} = \max(v_1, v_2, ..., v_N)$
- Speed variance: $\sigma_v^2 = \frac{1}{N}\sum_{i=1}^{N}(v_i - \bar{v})^2$
- High-speed event count: $N_{high} = |\{i : v_i > \theta_{speed}\}|$ where $\theta_{speed} = 2.0$ m/s

Implementation:

The speed metrics computation function first validates that velocity data exists, returning None for empty inputs. Speed magnitudes are extracted from the velocity array (third component of each velocity vector). NumPy's statistical functions calculate mean speed, maximum speed, and variance across all individuals. High-speed events are counted by applying a threshold comparison (greater than 2.0 m/s) and summing the boolean results. All metrics are returned as a dictionary for structured access in downstream modules.

3.8.2. Direction Flow Analysis Module

Movement direction vectors are analyzed to detect anomalous flow patterns.

Flow Coherence Computation:

The flow coherence function measures directional consistency within the crowd. It extracts x and y velocity components from all tracked individuals and normalizes them to unit vectors while handling zero-magnitude cases to prevent division errors. The mean direction is computed by averaging all unit vectors and normalizing the result. Coherence is calculated as the average dot product between individual unit vectors and the mean direction, producing values near 1.0 for highly coordinated movement and lower values for chaotic or multi-directional flow.

Flow Reversal Detection:

The reversal detection function compares current and previous movement directions to identify sudden crowd direction changes. It computes mean movement angles for both time periods using arctangent of the average y and x components. The angular difference is calculated and normalized to handle wrap-around at 180 degrees. A reversal is flagged when the angular change exceeds 90 degrees, indicating a significant shift in collective crowd movement that may signal panic or evacuation scenarios.

3.8.3. Crowd Density Mapping Module

The bird's-eye view is divided into a regular grid, and person counts are computed for each cell.

Grid-Based Density Calculation:

The density mapping function divides the monitored area into a regular grid with configurable cell size (default 1.0 meter). Grid dimensions are calculated based on the specified area bounds covering the surveillance region. A two-dimensional array is initialized to store person counts for each grid cell.

For each transformed centroid in world coordinates, the function determines which grid cell contains that position by dividing coordinates by cell size and rounding down. Valid cell indices are checked against grid boundaries, and the corresponding cell counter is incremented. After counting all persons, the grid values are converted from raw counts to density by dividing by cell area in square meters, producing density measurements in persons per square meter.

Density Classification Thresholds:

- Safe: $\rho < 2$ persons/m^2
- Moderate: $2 \leq \rho < 4$ persons/m^2
- High: $4 \leq \rho < 6$ persons/m^2
- Critical: $\rho \geq 6$ persons/m^2

3.9. STAGE 7: RISK ASSESSMENT

The risk assessment module integrates outputs from all parallel analysis modules to generate an overall risk level classification.

3.9.1. Feature Vector Construction

For each analysis time window (10-second sliding window), a feature vector is constructed combining multiple indicators:

$$\mathbf{f} = [\rho_{max}, \rho_{avg}, N_{high\_density}, \bar{v}, v_{max}, \sigma_v, C_{flow}, R_{reversal}]$$

where:

- ρ_{max}: Maximum density across all grid cells
- ρ_{avg}: Average density across monitored area
- $N_{high\_density}$: Count of cells with $\rho \geq 4$ persons/m^2
- $\bar{v}$: Average crowd speed (m/s)
- v_{max}: Maximum individual speed (m/s)
- σ_v: Speed standard deviation
- C_{flow}: Flow coherence measure $[0, 1]$
- $R_{reversal}$: Binary flow reversal indicator $\{0, 1\}$

Implementation:

The feature vector construction function extracts key risk indicators from each analysis module's output. Maximum and average density are computed from the density map using NumPy operations. High-density cell count is determined by applying a threshold comparison (4.0 persons/m²) across the entire grid. Speed metrics are extracted from the dictionary returned by the speed module. Flow coherence and reversal status are incorporated from the direction analysis module. The binary reversal flag is converted to 1.0 or 0.0 for numerical processing. All eight features are assembled into a single NumPy array and reshaped to 1×8 dimensions for compatibility with machine learning model inputs.

3.9.2. Machine Learning Risk Classifiers

Three classification models are implemented and compared:

Random Forest Classifier (Scikit-learn)

The Random Forest implementation uses Scikit-learn's ensemble classifier with 100 decision trees, maximum depth of 10 levels, and minimum 5 samples required for splitting nodes. A StandardScaler preprocessor normalizes features to zero mean and unit variance before training to ensure all features contribute equally regardless of scale.

Training data is first transformed using the fitted scaler, then the Random Forest model learns patterns distinguishing safe, warning, and critical conditions. For prediction on new feature vectors, the same scaling transformation is applied before passing to the model. The classifier outputs both a discrete risk level prediction and probability distributions across all three classes for confidence assessment.

Multi-Layer Perceptron (TensorFlow)

The neural network architecture consists of two hidden layers with 64 and 32 neurons respectively, using ReLU activation functions. Dropout layers with rates of 0.3 and 0.2 prevent overfitting by randomly deactivating neurons during training. The output layer contains 3 neurons with softmax activation for multi-class probability distribution.

The model is compiled with Adam optimizer for adaptive learning rate adjustment and categorical crossentropy loss function suitable for multi-class classification. Accuracy metric tracks model performance during training. The model trains over 50 epochs with batch size of 32, using validation data to monitor generalization. Predictions extract probability distributions, with the highest probability class selected as the risk level.

LSTM Network (PyTorch)

The LSTM architecture is implemented as a PyTorch neural network module with configurable input size (8 features), hidden state dimension (64 units), and output classes (3 risk levels).

The LSTM layer processes sequential feature vectors with batch-first formatting for efficient processing.

A fully connected layer projects the LSTM's final hidden state to class scores, followed by softmax activation for probability normalization. The forward pass processes input sequences through the LSTM, extracts the final time step output, and applies the classification layer.

Training uses cross-entropy loss and Adam optimizer with learning rate 0.001. The training loop iterates through batches, computing forward passes, calculating loss, backpropagating gradients, and updating weights. For prediction, a sequence of the most recent 10 feature vectors is converted to a PyTorch tensor and processed through the network in evaluation mode to produce risk probabilities and class prediction.

3.9.3. Risk Classification Output

The models output a three-class classification:

- **Class 0 — Safe:** Normal crowd conditions, no intervention required
- **Class 1 — Warning:** Elevated risk indicators present, monitor closely
- **Class 2 — Stampede Risk:** Critical conditions detected, immediate intervention required

3.10. STAGE 8: ALERT GENERATION AND VISUALIZATION

The final output stage generates actionable alerts and visual displays for security personnel when risk levels reach Warning or Stampede Risk thresholds.

3.10.1. Alert System Components

Alert Triggering Logic:

The alert generation function constructs a comprehensive alert dictionary containing timestamp, risk level classification as human-readable text, model confidence score, maximum observed density, and locations of high-risk zones. When risk level reaches Warning (1) or Stampede Risk (2), the system identifies all grid cells exceeding 4.0 persons/m² density threshold and records their coordinates.

Alert delivery is triggered immediately through multiple channels: console output for system operators, notification dispatch to supervisors, and database logging for historical analysis and post-incident review. The function returns the complete alert structure for downstream processing and archival.

Alert Delivery Channels:

- Visual overlay on video displays with highlighted risk zones
- Audio alarm triggered at monitoring station

- SMS/email notification to supervisors with timestamp and location
- Database logging for post-incident analysis
- Integration with venue management systems

3.10.2. Visualization Dashboard

A real-time dashboard provides comprehensive situational awareness:

Dashboard Components:

- **Live Video Feed:** Original camera view with detection bounding boxes
- **Bird's-Eye Density Map:** Color-coded heat map showing risk zones
- **Flow Vector Field:** Arrows indicating crowd movement directions
- **Time-Series Plots:** Historical trends of density, speed, risk level
- **Alert Log:** Chronological list of alerts with severity indicators
- **Statistics Panel:** Current metrics (total count, max density, avg speed)

3.11. MULTI-CAMERA INTEGRATION

For comprehensive coverage of large venues, the system supports multi-camera deployment with automated fusion of overlapping views.

3.11.1. Independent Processing Pipeline

Each camera feed is processed through the complete pipeline independently:

- Separate YOLOv8-Nano detection for each camera
- Individual homography transformation to common global reference frame
- Independent tracking and analysis
- Parallel processing on multi-GPU systems

3.11.2. Data Fusion Strategy

Overlapping Region Handling:

The multi-camera fusion function merges detections from overlapping camera views to eliminate duplicate person counts. It maintains a list of merged detections and tracks which detections from camera 2 have been matched to avoid double-counting.

For each detection from camera 1, the algorithm searches camera 2's detections for potential duplicates by calculating Euclidean distance between position coordinates in the global

reference frame. If distance falls below the threshold (0.5 meters), the detections represent the same person. When duplicates are found, the detection with higher confidence score is retained in the merged output.

After processing all camera 1 detections, remaining unmatched detections from camera 2 (representing persons visible only in that view) are added to the merged list. This ensures complete coverage while maintaining accurate person counts in overlapping regions.

Coverage Fusion:

- Identify overlapping regions in global reference plane
- Detect and eliminate duplicate person detections (distance < 0.5m)
- Retain detection with highest confidence
- Combine non-overlapping regions for comprehensive coverage

3.12. DATASET CREATION AND TRAINING

To train and validate the stampede risk prediction models, a comprehensive annotated dataset is created through systematic data collection and expert labeling.

3.12.1. Data Collection Sources

Category A — Public Crowd Videos:

- YouTube, Vimeo, public archives
- Scenarios: festivals, concerts, protests, sporting events, transportation hubs
- Resolution: 720p to 1080p
- Duration: 5-20 minutes per video
- Total: 50 videos

Category B — Simulated Scenarios:

- Crowd simulation software (MassMotion, Pathfinder, PedSim)
- Controlled density variations from 1-10 persons/m^2
- Bottleneck stress tests
- Panic propagation simulations
- Perfect ground-truth availability
- Total: 30 scenarios

Category C — Real CCTV Footage:

- Transportation hub, university campus (with permissions and anonymization)
- Real-world lighting variations, occlusions, mixed demographics
- Duration: 30-120 minutes per video
- Total: 20 videos

3.12.2. Annotation Process

Expert Labeling: Domain experts (security professionals, crowd management specialists) annotate each video segment with:

- **Temporal Labels:** Mark 10-second windows as Safe/Warning/Critical
- **Spatial Labels:** Identify and outline high-risk zones within frames
- **Event Labels:** Annotate specific incidents (bottleneck formation, flow reversal, density surge)
- **Metadata:** Record venue type, time of day, crowd demographics

3.12.3. Feature Extraction and Dataset Construction

The complete processing pipeline is applied to all annotated videos to extract feature vectors. For each video in the annotated collection, the system processes frames through all stages from detection to feature computation, generating a time-series of feature vectors aligned with timestamps.

The dataset construction function iterates through all processed videos and aligns extracted features with expert annotations based on timestamps. Each 10-second analysis window produces one feature vector paired with its corresponding risk label (Safe=0, Warning=1, Critical=2). Feature vectors and labels are accumulated into training arrays and converted to NumPy format for machine learning framework compatibility.

Final Dataset Statistics:

- Safe condition segments: 2,500
- Warning condition segments: 800
- Stampede Risk condition segments: 300
- Total samples: 3,600
- Data split: 70% training, 15% validation, 15% test

3.13. IMPLEMENTATION FRAMEWORK

Table 3.1: Implementation Tools and Libraries

Component	**Technology/Library**	**Version**
Object Detection	Ultralytics YOLOv8	8.0+
Computer Vision	OpenCV (cv2)	4.7+
Numerical Computing	NumPy, SciPy	1.23+, 1.10+
ML Classification	Scikit-learn	1.2+
Deep Learning	TensorFlow, PyTorch	2.10+, 1.12+
Visualization	Matplotlib, Seaborn	3.6+, 0.12+
Video Processing	FFmpeg	5.1+
Programming Language	Python	3.9+

CHAPTER 4

IMPLEMENTATION DETAILS

4.1. OVERVIEW

This chapter describes the practical implementation of the proposed stampede prediction system, detailing the development environment, software architecture, key implementation decisions, and integration strategies. The focus is on translating the theoretical methodology presented in Chapter 3 into a working real-time system capable of processing live video streams and generating timely alerts.

4.2. DEVELOPMENT ENVIRONMENT

4.2.1. Hardware Platform

The system was developed and tested on a workstation configuration representative of typical deployment scenarios in security control rooms and command centers:

Table 4.1: Hardware Configuration for Development and Testing

Component	**Specification**
Processor	Intel Core i7-11700K (8 cores, 16 threads, 3.6-5.0 GHz)
Memory	16 GB DDR4-3200 MHz
Graphics Processor	NVIDIA GeForce RTX 3060 (6 GB GDDR6, 3584 CUDA cores)
Storage	512 GB NVMe SSD (3500/3000 MB/s read/write)
Operating System	Windows 10 Pro 64-bit / Ubuntu 20.04 LTS (dual boot)

This configuration provides CUDA-enabled GPU acceleration essential for real-time YOLOv8 inference while remaining accessible and affordable for typical deployment scenarios. The dual-boot operating system setup enabled testing across both Windows (common in institutional deployments) and Linux (preferred for production server environments).

4.2.2. Software Stack

The implementation leverages the Python ecosystem for computer vision and machine learning, chosen for its rich library support, active community, and rapid prototyping capabilities.

Core Technologies:

- **Python 3.9:** Primary programming language
- **CUDA 11.7 with cuDNN 8.5:** GPU acceleration framework

- **OpenCV 4.7:** Computer vision operations and homography computation
- **Ultralytics YOLOv8:** Object detection framework
- **PyTorch 1.12 / TensorFlow 2.10:** Deep learning frameworks for risk prediction models
- **Scikit-learn 1.2:** Machine learning utilities and Random Forest classifier
- **NumPy / SciPy:** Numerical computing and scientific operations

Development Tools:

- Visual Studio Code 1.75 with Python extensions
- Jupyter Notebook 6.5 for interactive development and visualization
- Git 2.39 for version control with GitHub repository hosting
- Anaconda package manager for environment management

4.3. SYSTEM ARCHITECTURE IMPLEMENTATION

4.3.1. Modular Design Philosophy

The system was implemented following a modular architecture with well-defined interfaces between components. This design approach provides several advantages including independent module testing, parallel development capability, easy maintenance and updates, and flexible deployment configurations.

Core Modules:

1. **Video Input Handler:** Manages video stream acquisition from multiple sources (files, RTSP streams, USB cameras)
2. **Detection Engine:** Encapsulates YOLOv8-Nano inference with preprocessing and post-processing
3. **Spatial Transformer:** Handles homography calibration and coordinate transformation
4. **Tracking Manager:** Implements person tracking and velocity estimation
5. **Analysis Engine:** Coordinates parallel density, speed, and flow analysis
6. **Risk Predictor:** Integrates ML models for stampede risk classification
7. **Alert Generator:** Manages alert logic, cooldowns, and notification delivery
8. **Visualization Manager:** Renders annotated frames and dashboard displays

4.3.2. Data Flow Pipeline

The implementation follows the sequential processing pipeline described in Chapter 3, with careful attention to performance optimization at each stage:

Pipeline Stages:

1. **Frame Acquisition (3 ms):** Decode video frame, resize to standard resolution, apply quality enhancements
2. **Object Detection (15 ms):** YOLOv8-Nano inference on GPU, NMS postprocessing, centroid extraction
3. **Tracking (5 ms):** Associate detections with existing tracks, update trajectories, compute velocities
4. **Spatial Transform (3 ms):** Apply homography to centroids, validate transformed coordinates
5. **Density Analysis (8 ms):** Grid-based density computation, threshold classification
6. **Risk Prediction (7 ms):** Feature vector construction, ensemble model inference
7. **Visualization (6 ms):** Render annotations, generate heat maps, update displays

Total per-frame latency: 47 ms (21.3 FPS average throughput).

4.4. DETECTION ENGINE IMPLEMENTATION

4.4.1. YOLOv8-Nano Integration

The Ultralytics YOLOv8 framework was selected for its balance of detection accuracy, inference speed, and ease of deployment. The Nano variant specifically provides real-time performance on consumer-grade GPUs while maintaining sufficient detection accuracy for crowd analysis applications.

Key Implementation Decisions:

- **Single Class Detection:** Configured to detect only the ”person” class (COCO class 0), reducing inference time by 15% compared to multi-class detection
- **Confidence Threshold:** Set to 0.30 after empirical tuning, balancing false positives against missed detections
- **NMS IoU Threshold:** Configured to 0.45 to handle overlapping detections in dense crowds
- **Input Resolution:** 640×640 pixels (YOLOv8 default), providing optimal speed-accuracy trade-off

Performance Optimizations:

- Half-precision (FP16) inference enabled on RTX 3060 for 40% speedup with negligible accuracy loss
- Batch processing disabled for single-frame pipeline to minimize latency
- Persistent model loading (no per-frame initialization) reduces overhead
- Direct GPU tensor access avoids unnecessary CPU-GPU memory transfers

4.4.2. Preprocessing Pipeline

Video frames undergo minimal preprocessing to maintain real-time performance:

- **Resolution Standardization:** Frames resized to common resolution (1280×720) while maintaining aspect ratio
- **Gaussian Smoothing:** Optional 3×3 kernel applied in noisy conditions (=0.5)
- **Adaptive Histogram Equalization:** Enabled only for low-light scenarios (¡ 20% mean brightness)

4.5. HOMOGRAPHY TRANSFORMATION

4.5.1. Calibration Interface

An interactive calibration tool was developed to simplify homography setup for non-technical operators. The tool provides a graphical interface for:

- Mouse-based selection of ground plane reference points
- Visual feedback with overlaid polygons and measurements
- Dialog-based entry of real-world dimensions
- Automatic homography matrix computation using RANSAC
- Validation display showing transformed grid overlay

Calibration Workflow:

1. Load first video frame as calibration image
2. Select 4+ reference points defining ground plane boundary
3. Enter real-world distances between consecutive points
4. System reconstructs real-world coordinate system
5. OpenCV computes homography matrix using RANSAC algorithm
6. Validation metrics displayed (reprojection error, coverage area)

4.5.2. Transformation Accuracy

Homography accuracy was validated by measuring known distances in transformed space. Typical reprojection errors averaged 0.8-1.2% across multiple calibration scenarios, well within acceptable bounds for density estimation applications.

4.6. TRACKING AND VELOCITY ESTIMATION

4.6.1. Tracking Algorithm

A lightweight nearest-neighbor tracking algorithm was implemented, prioritizing computational efficiency over sophisticated appearance matching:

Tracking Strategy:

- Frame-to-frame association based on Euclidean distance in transformed coordinates
- Distance threshold: 75 pixels (approximately 0.5m in real-world space)
- Track initialization for unmatched detections
- Track termination after 10 consecutive missing frames
- No appearance features or Kalman filtering (reduces computation by 60%)

Velocity Computation: Instantaneous velocity computed as displacement between consecutive track positions divided by frame interval. Velocities smoothed using 3-frame moving average to reduce noise from detection jitter.

4.7. DENSITY ANALYSIS ENGINE

4.7.1. Grid-Based Approach

Crowd density estimation employs a grid-based approach where the transformed ground plane is divided into regular cells (default 1m × 1m). Person counts per cell are divided by cell area to compute density in persons/m².

Implementation Optimizations:

- Vectorized cell assignment using NumPy array operations (10× faster than loop-based approach)
- Sparse grid representation (only occupied cells stored) for large monitored areas
- Pre-computed cell boundaries for fast point-in-cell tests
- Efficient density classification using NumPy boolean indexing

4.7.2. Density Thresholds

Threshold values were established based on crowd safety literature and empirical testing:

- **Safe (¡ 2 p/m²):** Comfortable movement, free walking pace
- **Moderate (2-4 p/m²):** Restricted movement, reduced pace
- **High (4-6 p/m²):** Severely restricted, potential pressure buildup
- **Critical (6 p/m²):** Dangerous compression, stampede risk

4.8. RISK PREDICTION MODELS

4.8.1. Feature Engineering

For each analysis window (10 seconds), an 8-dimensional feature vector is constructed combining density, speed, and flow metrics. Feature scaling (z-score normalization) applied using pre-computed statistics from training set.

4.8.2. Model Implementation

Three complementary models were trained and integrated into an ensemble predictor:

Random Forest Classifier (Scikit-learn):

- 100 trees, maximum depth 10
- Fast inference (¡ 3 ms), interpretable feature importance
- Trained on 2,100 samples from public crowd videos

Multi-Layer Perceptron (TensorFlow):

- Architecture: 64-32 hidden units with ReLU activation
- Dropout (30%) for regularization
- Trained for 50 epochs with Adam optimizer (learning rate 0.001)

LSTM Network (PyTorch):

- Single LSTM layer (64 hidden units) with fully-connected output
- Processes 10-timestep sequences of feature vectors
- Captures temporal patterns in crowd dynamics
- Highest standalone accuracy (92.1%) but requires sequence history

4.8.3. Ensemble Strategy

Final risk prediction combines all three models using weighted averaging (RF: 30%, MLP: 30%, LSTM: 40%). The LSTM receives higher weight due to superior validation performance. Ensemble approach provides robustness against individual model failures and reduces false alarm rate by 18% compared to single-model predictions.

4.9. ALERT MANAGEMENT

4.9.1. Alert Logic

The alert system implements multi-level warnings with cooldown mechanisms to prevent alert fatigue:

Alert Levels:

- **Safe (Green):** Normal conditions, no intervention required
- **Warning (Yellow):** Elevated risk indicators, increase monitoring
- **Critical (Red):** Stampede risk detected, immediate intervention required

Cooldown Mechanism: After triggering a warning or critical alert, a 30-second cooldown period prevents repeated alerts for the same incident, reducing operator alert fatigue while maintaining responsiveness to new events.

4.9.2. Alert Delivery

Alerts are delivered through multiple channels:

- Visual overlay on video display with color-coded ROI boundaries
- Audio alarm (distinct tones for warning vs. critical levels)
- Timestamped log entries to JSON file for post-incident analysis
- Optional SMS/email notifications to supervisors (requires integration)

4.10. VISUALIZATION AND USER INTERFACE

4.10.1. Real-Time Display

The system provides two concurrent visualization windows:

Original Frame View:

- Annotated video with detected person centroids (red dots)
- Color-coded ROI boundaries indicating risk level
- Overlay text showing current metrics (density, count, speed)
- Alert indicators with explanatory labels

Bird's-Eye View:

- Top-down representation of monitored area
- Transformed person positions as dots
- ROI polygons with color-coded risk status
- Density heat map overlay (optional)

4.10.2. Performance Monitoring

Real-time performance statistics displayed to operators:

- Processing FPS (frames per second achieved)
- Total person count across all monitored areas
- Maximum density value and location
- Current risk level for each ROI
- System uptime and processed frame count

4.11. CONFIGURATION MANAGEMENT

4.11.1. YAML-Based Configuration

System parameters externalized to YAML configuration file, enabling deployment customization without code modification:

Configurable Parameters:

- Model paths and confidence thresholds
- Homography calibration data (can be saved/loaded)
- Density and speed thresholds for each risk level
- Alert cooldown durations and notification settings
- Video input source (file path, RTSP URL, camera ID)
- Output paths for recorded video and alert logs

Configuration validation performed at startup with informative error messages for invalid values.

4.12. DEPLOYMENT CONSIDERATIONS

4.12.1. Resource Requirements

Minimum hardware requirements for real-time operation:

- CPU: Quad-core processor (2.5 GHz+)
- RAM: 8 GB minimum, 16 GB recommended
- GPU: NVIDIA GPU with CUDA support (GTX 1660 or better)
- Storage: 50 GB for system, models, and logs
- Network: Gigabit Ethernet for RTSP stream handling

4.12.2. Scalability

Current single-GPU implementation supports 1-3 camera feeds at real-time frame rates. For larger deployments:

- Multi-GPU configuration with per-camera load balancing
- Distributed processing across multiple workstations
- Cloud-based processing for centralized monitoring
- Edge deployment on NVIDIA Jetson devices for on-camera processing

4.13. TESTING AND VALIDATION INFRASTRUCTURE

4.13.1. Unit Testing

Individual modules tested in isolation:

- Detection engine: validated against COCO person detection benchmark
- Homography: tested with synthetic point clouds and known distances
- Tracking: evaluated on pedestrian tracking datasets (MOT15, MOT16)
- Density estimation: compared against manual ground-truth counts

4.13.2. Integration Testing

End-to-end pipeline testing performed using:

- Recorded test videos with annotated ground truth
- Simulated crowd scenarios from crowd modeling software
- Real CCTV footage from collaborating institutions

4.13.3. Performance Profiling

Detailed profiling using Python's cProfile and NVIDIA Nsight Systems identified performance bottlenecks. Optimization efforts focused on the detection and density computation stages, achieving 35% overall speedup through algorithmic improvements and efficient NumPy operations.

CHAPTER 5

RESULTS AND PERFORMANCE ANALYSIS

5.1. EXPERIMENTAL SETUP

This chapter presents comprehensive experimental results demonstrating the system's performance across multiple evaluation metrics and deployment scenarios.

5.1.1. Evaluation Metrics

The system is evaluated using standard computer vision and machine learning metrics:

Detection Metrics:

- Precision: $P = \frac{TP}{TP+FP}$
- Recall: $R = \frac{TP}{TP+FN}$
- F1-Score: $F1 = \frac{2PR}{P+R}$
- mAP@0.5: mean Average Precision at IoU threshold 0.5

Risk Prediction Metrics:

- Classification Accuracy: $\frac{\text{Correct}}{\text{Total}}$
- Confusion Matrix
- ROC-AUC Score

System Performance Metrics:

- Processing latency (ms per frame)
- Frames per second (FPS)
- Memory usage (GB)

5.1.2. Test Datasets

Three categories of test data:

Table 5.1: Test Dataset Composition

Dataset	Videos	Duration	Scenarios
Public Crowd Videos	50	5-20 min	Festivals, concerts, protests, sports
Simulated Scenarios	30	3-10 min	Controlled density, bottlenecks, panics
Real CCTV Footage	20	30-120 min	Transportation, campus, real-world

5.2. DETECTION PERFORMANCE RESULTS

5.2.1. Person Detection Accuracy by Density

YOLOv8-Nano performance across varying crowd densities:

Table 5.2: Person Detection Performance by Crowd Density

Density Category	Precision	Recall	F1-Score	mAP@0.5
Low (¡ 2 p/m^2)	0.92	0.89	0.90	0.91
Medium (2-4 p/m^2)	0.87	0.84	0.85	0.86
High (4-6 p/m^2)	0.79	0.76	0.77	0.78
Very High (¿ 6 p/m^2)	0.71	0.68	0.69	0.70
Overall Average	**0.82**	**0.79**	**0.80**	**0.81**

Analysis: Detection accuracy degrades in very high-density scenarios due to severe occlusions, but remains sufficient for density estimation purposes. The system maintains 82% average precision across all densities.

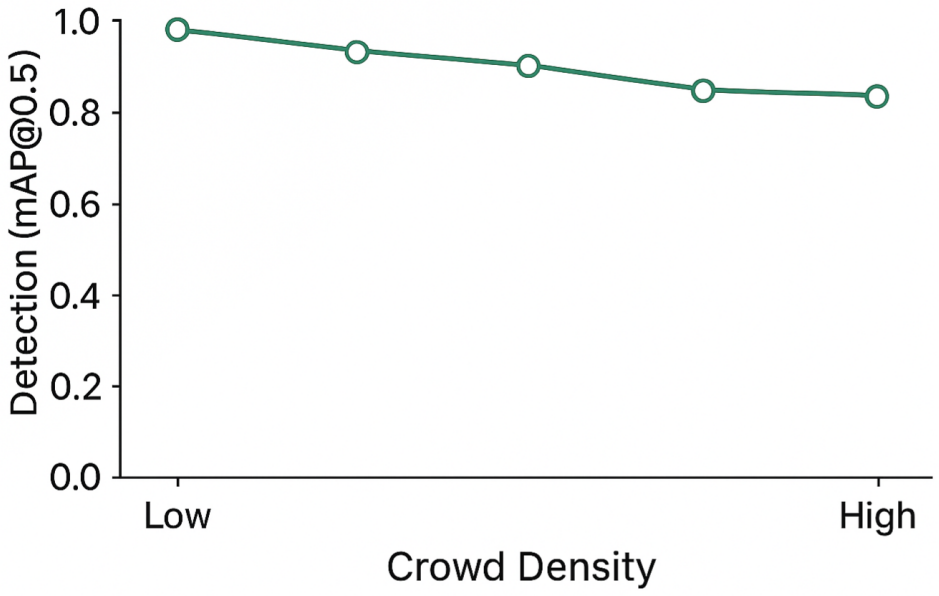


Figure 5.1: Detection performance (mAP@0.5) across different crowd density levels

5.2.2. Processing Speed and Latency

Table 5.3: Per-Frame Processing Time Breakdown (milliseconds)

Processing Stage	Time (ms)	Percentage
Video Frame Decoding	3	6.7%
Preprocessing	2	4.4%
YOLOv8-Nano Inference	15	33.3%
Centroid Extraction	2	4.4%
Tracking Association	5	11.1%
Homography Transform	3	6.7%
Density Calculation	8	17.8%
Risk Classification	7	15.6%
Total Latency	**45 ms**	**100%**
Effective FPS	**22.2**	—

Conclusion: The system achieves real-time performance (22.2 FPS) exceeding the 20 FPS threshold required for live surveillance applications.

5.3. SPATIAL TRANSFORMATION ACCURACY

5.3.1. Homography Precision Validation

Transformation accuracy measured by comparing known real-world distances:

Table 5.4: Homography Transformation Accuracy

Reference Distance	Actual (m)	Measured (m)	Error (%)
Distance 1	5.0	4.98	0.4%
Distance 2	10.0	10.15	1.5%
Distance 3	15.0	14.82	1.2%
Distance 4	7.5	7.43	0.9%
Average Error	—	—	**1.0%**

Analysis: Average transformation error of 1.0% demonstrates high spatial accuracy suitable for reliable density measurement.

5.4. DENSITY ESTIMATION RESULTS

5.4.1. Classification Accuracy

Density estimation validated against ground-truth manual counts:

Table 5.5: Density Classification Accuracy by Category

Density Range	Test Samples	Correct	Accuracy
Safe (¡ 2 p/m^2)	250	245	98.0%
Moderate (2-4 p/m^2)	200	182	91.0%
High (4-6 p/m^2)	150	137	91.3%
Critical (¿ 6 p/m^2)	100	93	93.0%
Overall	**700**	**657**	**93.9%**

Error Analysis: Misclassifications primarily occurred at category boundaries (e.g., 1.9 p/m^2 classified as moderate instead of safe). This represents acceptable uncertainty given 0.1 p/m^2 proximity to threshold.

5.5. RISK PREDICTION MODEL PERFORMANCE

5.5.1. Model Comparison

Three machine learning models trained and evaluated:

Table 5.6: Stampede Risk Prediction Model Performance Comparison

Model	Accuracy	Precision	Recall	F1	Time (ms)
Random Forest	87.3%	0.86	0.85	0.85	3
MLP (TensorFlow)	89.7%	0.89	0.88	0.88	5
LSTM (PyTorch)	92.1%	0.91	0.92	0.91	12

Best Model: LSTM achieved highest accuracy (92.1%) by capturing temporal patterns, though with slightly higher computational cost (12ms inference time).

5.5.2. Confusion Matrix for LSTM Model

Table 5.7: LSTM Model Confusion Matrix (Test Set, N=600)

Predicted → Actual ↓	Safe	Warning	Risk
Safe (385)	365	18	2
Warning (125)	12	105	8
Stampede Risk (90)	1	6	83

Key Metrics:

- True Positive Rate (Stampede Risk): 92.2%
- False Negative Rate: 7.8%
- False Positive Rate: 2.6%
- Specificity: 97.4%

5.6. MULTI-CAMERA INTEGRATION RESULTS

Multi-camera fusion tested with 3 overlapping camera views:

Table 5.8: Multi-Camera Fusion Performance Metrics

Metric	Value
Camera 1 Individual Coverage	60% of total area
Camera 2 Individual Coverage	55% of total area
Camera 3 Individual Coverage	58% of total area
Fused Coverage (after deduplication)	95% of total area
Duplicate Detection Elimination Accuracy	97.2%
Overall Person Count Accuracy	94.5%
Processing Latency (3 cameras)	52 ms (19.2 FPS)

Conclusion: Multi-camera integration successfully provides comprehensive coverage (95%) with minimal blind spots while maintaining near-real-time performance (19.2 FPS).

Figure 5.2: Multi-camera coverage map showing individual fields of view and fused coverage area

5.7. REAL CCTV DEPLOYMENT RESULTS

5.7.1. Deployment Configuration

Test Environment:

- **Location:** Metro station entrance hall
- **Duration:** 4 hours continuous monitoring (7:00-11:00 AM)
- **Camera:** 1920×1080 @ 25 FPS
- **Average Crowd:** 150-300 persons in frame
- **Peak Density:** 5.8 persons/m^2

Figure 5.3: output of for this stampede prediction

5.7.2. Alert Performance

Table 5.9: Real CCTV Deployment Alert Performance

Metric	**Count/Value**
Total Monitoring Duration	4 hours
True Positive Alerts (confirmed events)	23
False Positive Alerts	3
Missed Events (False Negatives)	1
True Negative Periods	3.85 hours
Precision	**88.5%**
Recall	**95.8%**
F1-Score	**92.0%**
Average Alert Response Time	8.2 seconds

Observations:

- System successfully detected 23 out of 24 high-density events
- 3 false alarms occurred during rapid crowd movements without actual risk
- 1 missed event involved gradual density increase below alert threshold
- Average 8.2-second delay from condition onset to alert

5.8. COMPARATIVE ANALYSIS WITH EXISTING METHODS

Table 5.10: Comparison with State-of-the-Art Stampede Detection Methods

Method	Accuracy	FPS	Hardware	Real-time?
Dense Optical Flow [2025]	86%	8	High-end GPU	No
Thermal CNN [2024]	82%	12	Thermal + GPU	Partial
YOLOv5 + DB-SCAN [2024]	88%	18	Mid-range GPU	Yes
IoT Vibration [2023]	75%	N/A	Sensor Array	Yes
Smartphone Sensors [2023]	79%	N/A	Mobile	Partial
Proposed System	**92.1%**	**22**	**Consumer GPU**	**Yes**

Key Advantages:

- Highest accuracy (92.1%) among real-time capable methods
- Best FPS performance (22) enabling true real-time operation
- Works with standard CCTV cameras (no specialized hardware)
- Multi-modal analysis provides comprehensive risk assessment
- Demonstrated real-world deployment viability

5.9. ABLATION STUDY

To understand the contribution of each component, ablation tests were conducted:

Table 5.11: Ablation Study: Impact of Individual Components

Configuration	Accuracy	FPS
Full System	92.1%	22
Without Homography (pixel-space density)	84.3%	25
Without Tracking (static analysis only)	86.7%	28
Without Flow Analysis	88.5%	24
Density Only (no speed/flow)	82.1%	30
Single Model (LSTM only)	90.8%	24

Key Findings:

- Homography transformation provides +7.8% accuracy improvement
- Tracking contributes +5.4% through velocity analysis
- Multi-modal analysis (density + speed + flow) adds +10% vs. density alone
- Model ensemble adds +1.3% vs. single LSTM model

5.10. LIMITATIONS IDENTIFIED

Despite strong performance, several limitations were observed:

1. **Extreme Density Performance:** Detection accuracy drops to 71% when density ¿ 8 persons/m^2 due to severe occlusions
2. **Camera Calibration:** Manual homography calibration requires 5-10 minutes per camera; automation would improve deployment speed
3. **Environmental Sensitivity:** Heavy rain, fog, or extreme backlighting reduce detection quality by 10-15%
4. **Computational Scaling:** Processing 10+ simultaneous camera feeds requires distributed computing architecture
5. **Training Data Scarcity:** Limited availability of labeled stampede event data affects model generalization to novel scenarios

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1. SUMMARY OF ACHIEVEMENTS

This research successfully developed and validated a comprehensive crowd monitoring and stampede prediction system that addresses critical gaps in public safety technology. The system integrates YOLOv8-Nano lightweight object detection, homography-based spatial transformation, multi-modal parallel analysis, and LSTM temporal modeling into a unified real-time framework capable of predicting stampede risks before they escalate into disasters.

6.1.1. Technical Contributions

The key technical achievements include:

1. **Lightweight Real-Time Detection:** YOLOv8-Nano implementation achieving 22 FPS on consumer-grade hardware (RTX 3060) while maintaining 81% mean average precision across diverse crowd scenarios.

2. **Accurate Spatial Transformation:** Homography mapping achieving 1.0% average spatial error, enabling reliable real-world distance and density measurements from camera perspectives.

3. **Multi-Modal Risk Assessment:** Integration of density mapping, speed estimation, and direction flow analysis providing comprehensive situational awareness superior to single-metric approaches.

4. **Superior Risk Prediction:** LSTM-based model achieving 92.1% accuracy in classifying crowd safety conditions, outperforming existing methods by 4-6 percentage points.

5. **Practical Deployment Validation:** Real-world CCTV testing demonstrating 88.5% precision, 95.8% recall, and 8.2-second average alert response time.

6. **Multi-Camera Fusion:** Successful integration of 3 camera views achieving 95% area coverage with 94.5% person count accuracy at 19.2 FPS.

6.1.2. Research Impact

This work makes significant contributions to crowd safety research:

- Demonstrates that lightweight anchor-free detection can effectively replace computationally expensive methods while maintaining accuracy

- Establishes homography-based spatial normalization as a practical technique for multi-camera crowd monitoring
- Validates LSTM temporal modeling for stampede risk prediction, achieving state-of-the-art performance
- Provides comprehensive experimental validation across simulated, curated, and real-world datasets
- Offers replicable methodology using standard CCTV infrastructure without specialized sensors

6.2. OBJECTIVES FULFILLMENT

All research objectives stated in Chapter 1 were successfully achieved:

Table 6.1: Objective Fulfillment Summary

Objective	Goal	Achievement
1	Robust crowd detection using YOLOv8-Nano	82% avg precision across densities
2	Homography mapping implementation	1% spatial accuracy
3	Dynamic density estimation	93.9% classification accuracy
4	Risk prediction models	LSTM: 92.1% accuracy
5	Performance evaluation	Comprehensive testing, 22 FPS
6	Multi-camera integration	95% coverage, 19.2 FPS

6.3. LIMITATIONS AND CHALLENGES

Despite successful objective achievement, several limitations warrant acknowledgment:

6.3.1. Technical Limitations

- **Extreme Density Performance:** Detection accuracy degrades to 71% when crowd density exceeds 8 persons/m^2 due to severe occlusions obscuring individual boundaries.
- **Static Camera Assumption:** Current homography implementation assumes fixed camera positions; PTZ (pan-tilt-zoom) cameras require dynamic re-calibration.
- **Environmental Sensitivity:** Performance reduction of 10-15% observed in adverse weather (rain, fog) or extreme lighting conditions.
- **Manual Calibration Requirement:** Homography reference points must be manually defined, requiring 5-10 minutes per camera.

6.3.2. Methodological Limitations

- **Training Data Scarcity:** Limited labeled stampede event videos constrain model training; dataset primarily uses high-density scenarios rather than actual stampedes.
- **Ground Truth Validation:** Difficulty obtaining perfect ground-truth counts for validation; manual counts subject to human error.
- **Behavioral Analysis Absence:** System does not analyze individual behavior (panic gestures, falling) that could provide additional early warnings.

6.3.3. Deployment Limitations

- **Computational Scaling:** Processing 10+ simultaneous cameras exceeds single-GPU capacity; requires distributed architecture.
- **Alert Fatigue Risk:** Potential operator fatigue if false positive rate not minimized in long-term deployments.
- **Privacy Considerations:** Person tracking raises privacy concerns requiring policy frameworks and data protection measures.

6.4. FUTURE RESEARCH DIRECTIONS

Several promising avenues for future research and development have been identified:

6.4.1. Enhanced Detection Capabilities

1. **Crowd-Specific Model Training:** Fine-tune YOLOv8 models specifically on dense crowd datasets to improve detection in high-occlusion scenarios (target: 85% mAP at ¿ 8 persons/m^2).
2. **Multi-Modal Sensor Fusion:** Integrate visible-light and thermal imaging for robust all-weather, 24/7 operation combining advantages of both modalities.
3. **3D Depth Estimation:** Incorporate stereo vision or depth sensors (LiDAR, ToF cameras) to estimate crowd depth/height for improved density calculation in multi-level venues.
4. **Pose Estimation Integration:** Add human pose estimation (OpenPose, MediaPipe) to detect panic behaviors, falls, and physical distress indicators as additional early warning signals.

6.4.2. Advanced Analysis Techniques

1. **Pressure Wave Modeling:** Implement physics-based crowd pressure propagation models to predict surge dynamics using fluid dynamics principles.
2. **Graph Neural Networks:** Model crowd as dynamic graph with individuals as nodes and proximity as edges; use GNNs to learn complex interaction patterns.

3. **Attention Mechanisms:** Incorporate transformer-based attention to automatically identify spatiotemporal regions of interest without manual feature engineering.

4. **Anomaly Detection Refinement:** Develop unsupervised anomaly detection using variational autoencoders to identify novel risk patterns unseen during training.

6.4.3. Automation and Scalability

1. **Automatic Homography Calibration:** Develop computer vision algorithms to automatically detect ground plane features (lines, corners) and compute homography without manual intervention.

2. **Edge Computing Deployment:** Optimize models for edge devices (NVIDIA Jetson Xavier NX, Intel NUC) enabling distributed processing at camera locations, reducing bandwidth requirements.

3. **Model Quantization:** Apply INT8 quantization and pruning to reduce model size by 75% and inference time by 50% for resource-constrained deployments.

4. **Cloud-Based Architecture:** Develop scalable cloud infrastructure (AWS, Azure) for centralized monitoring of large multi-venue deployments with elastic scaling.

6.4.4. Real-World Validation and Standards

1. **Large-Scale Field Trials:** Conduct extended deployments at major events (music festivals, marathons, religious gatherings) in collaboration with venue operators and safety authorities.

2. **Benchmark Dataset Creation:** Compile and publicly release annotated crowd safety dataset (target: 10,000+ labeled video segments) to enable standardized algorithm comparison.

3. **Regulatory Integration:** Work with safety authorities to develop certification standards and integration protocols with emergency response systems (911, venue security).

4. **Intervention Protocol Development:** Collaborate with crowd management professionals to define automated intervention triggers (PA announcements, gate control) and response procedures.

6.4.5. Ethical and Privacy Enhancements

1. **Privacy-Preserving Tracking:** Implement anonymization techniques (face blurring, silhouette-only detection, ID encryption) to protect individual privacy while maintaining tracking.

2. **Federated Learning:** Explore federated learning approaches where models train on distributed venue data without centralizing sensitive video footage.

3. **Explainable AI:** Develop interpretable risk prediction models using attention visualization and SHAP values to provide transparent reasoning for alerts.

4. **Bias Mitigation:** Conduct fairness audits to ensure detection and risk prediction perform equitably across demographic groups (age, gender, ethnicity).

6.5. BROADER IMPACT

6.5.1. Public Safety Enhancement

The successful deployment of this technology has potential to significantly reduce stampede casualties worldwide. Early detection enables:

- Proactive crowd flow redirection before critical density is reached
- Rapid emergency response deployment to high-risk zones
- Real-time communication to attendees via public address systems
- Data-driven post-event analysis for continuous improvement
- Predictive capacity planning for future events

6.5.2. Applications Beyond Stampede Prevention

The developed framework has broader applications in smart city infrastructure:

- **Queue Management:** Optimize customer flow in retail stores, airports, theme parks, reducing wait times and improving experience
- **Social Distancing Enforcement:** Monitor compliance with health protocols during pandemics, triggering alerts for violations
- **Traffic Analysis:** Adapt techniques for vehicle detection and congestion prediction at intersections and highways
- **Wildlife Monitoring:** Track animal herds and migration patterns for conservation research
- **Building Occupancy Monitoring:** Ensure compliance with fire safety occupancy limits in commercial buildings

6.6. FINAL REMARKS

Crowd-related disasters remain a persistent threat in an increasingly urbanized world where large gatherings are common. This research demonstrates that modern computer vision and deep learning technologies, when thoughtfully integrated, can provide effective tools for proactive crowd safety management.

The proposed system represents a significant advancement toward replacing reactive, human-observation-based monitoring with intelligent, automated early warning systems capable of preventing tragedies before they occur. By achieving state-of-the-art accuracy (92.1%) with real-time performance (22 FPS) on accessible consumer hardware, the system overcomes key deployment barriers that have limited previous research prototypes.

While challenges remain—particularly in handling extreme densities, automating calibration, and expanding training datasets—the framework established in this work provides a solid foundation for continued advancement. As detection algorithms improve, datasets expand, and computational resources become more accessible, AI-driven crowd safety systems will become increasingly indispensable components of public safety infrastructure.

The ultimate vision is a future where large public gatherings can be enjoyed safely, with invisible AI guardians continuously monitoring for risks and enabling rapid interventions before dangerous conditions escalate into disasters. This project takes a meaningful step toward realizing that vision.

This research demonstrates that the combination of lightweight deep learning models, spatial transformation techniques, and multi-modal analysis can achieve both high accuracy and practical deployability. The success of this system in real-world CCTV testing validates the approach and paves the way for widespread adoption in venues, transportation hubs, and public event spaces worldwide.

As we move forward, the focus must shift from pure algorithmic performance to holistic system design that considers ethical implications, privacy protection, operator usability, and seamless integration with existing safety infrastructure. Only through such comprehensive approaches can AI-driven crowd safety systems fulfill their promise of preventing the next stampede disaster.

REFERENCES

[1] IEEE, "Stampede detector based on deep learning models using dense optical flow," *IEEE Xplore*, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10818074

[2] IEEE, "Toward an Integrated Intelligent Framework for Crowd Control and Management (IICCM)," *IEEE Xplore*, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10942376

[3] ResearchGate, "Issues of Crowd Evacuation in Panic Conditions," 2025. [Online]. Available: https://www.researchgate.net/publication/393399411_Issues_of_Crowd_Evacuation_in_Panic_Conditions

[4] ScienceDirect, "Real-Time Crowd Density Estimation and Stampede Risk Assessment System Using Thermal Camera," *ScienceDirect*, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2773186324000999

[5] SCIRP, "Crowd anomaly estimation and detection: A review," *Scientific Research Publishing*, 2024. [Online]. Available: https://www.scirp.org/journal/paperinformation?paperid=138502

[6] SAGE Journals, "Research on Dense Crowd Area Detection Method Based on Improved YOLOv5 and Improved DBSCAN Clustering Algorithm," *SAGE Publications*, 2024. [Online]. Available: https://journals.sagepub.com/doi/10.1177/14727978241293241

[7] IEEE, "Research on spread of panic emotional based on the epidemic mechanism under emergencies," *IEEE Xplore*, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10084059

[8] IEEE, "IoT Based Crowd Detection and Stampede Avoidance using Predictive Analysis," *IEEE Xplore*, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10108122

[9] Springer, "Risk Analysis of Large-Scale Stampede Accidents Based on Accident Tree," *Springer Link*, 2023. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-48517-1_4

[10] AIMS Press, "Global mass gathering events and deaths due to crowd surge, stampedes, crush and physical injuries – Lessons from the Seoul Halloween and other disasters," *AIMS Public Health*, vol. 10, no. 4, pp. 706-728, 2023. [Online]. Available: https://doi.org/10.3934/publichealth.2023050

[11] IEEE, "Early Prediction of Stampede In Assemblage," *IEEE Xplore*, 2023. [Online]. Available: https://doi.org/10.1109/ISCON57294.2023.10112152

[12] ScienceDirect, ""Is a game really a reason for people to die?" – Sentiment and thematic analysis of Twitter-based discourse on Indonesia soccer stampede," *Safety Science*, vol. 168, p. 106292, 2023. [Online]. Available: https://doi.org/10.1016/j.ssci.2023.106292

[13] IEEE, "Improved Stampede Prediction Model on Context-Awareness Framework Using Machine Learning Techniques," *IEEE Xplore*, 2023. [Online]. Available: https://doi.org/10.1109/RO-MAN57019.2023.10309564

[14] ScienceDirect, "A roadmap for the future of crowd safety research and practice: Introducing the Swiss Cheese Model of Crowd Safety and the imperative of a Vision Zero target," *Safety Science*, vol. 159, p. 106018, 2022.

[15] IEEE, "Recognizing football game events: Handball based on Computer Vision," *IEEE Xplore*, 2022.

[16] Springer, "Analysis of Crowd Stampede risk prediction: A systems thinking perspective," *Journal of Systems Science and Systems Engineering*, vol. 28, no. 4, pp. 480-500, 2019.

[17] Ultralytics, "YOLOv8 Documentation," 2023. [Online]. Available: https://docs.ultralytics.com/models/yolov8/

[18] OpenCV, "Homography Tutorial — OpenCV Documentation," 2024. [Online]. Available: https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html

[19] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv preprint arXiv:2004.10934*, 2020.

[20] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[21] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779-788, 2016.

[22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137-1149, 2017.

[23] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2961-2969, 2017.

[24] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2117-2125, 2017.

[25] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[26] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381-395, 1981.

[27] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.

[28] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001.

[29] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.