

## Real-time Crowd Monitoring and Management

Pavitra Kannan<sup>1</sup>, Vaishnavi Jadhav<sup>2</sup>, Yash Patil<sup>3</sup>, Kishan Hyalij<sup>4</sup>, Prof. Pradnya Bacchav<sup>5</sup>

<sup>1,2,3,4</sup>UG, Computer Engineering, Guru Gobind Singh College of Engineering and Research Centre, Nashik, Maharashtra, India.

<sup>5</sup>Assistant Professor, Computer Engineering, Guru Gobind Singh College of Engineering and Research Centre, Nashik, Maharashtra, India.

**Email** ID: pavitrakannan544@gmail.com<sup>1</sup>, jadhavvaishnavi1401@gmail.com<sup>2</sup>, yashpatil200312@gmail.com<sup>3</sup>, kishanhyalij@gmail.com<sup>4</sup>, pradnyak.bachhav@gmail.com<sup>5</sup>

### Abstract

Efficient crowd monitoring is essential for public safety, space management, and emergency response planning. Traditional methods rely on manual observation, which is labour-intensive and prone to errors. This research presents a real-time crowd monitoring system that integrates classical computer vision techniques, enabling automated detection and density estimation of people in public spaces. The system employs background subtraction, contour detection, and optical flow tracking to monitor crowd movement dynamically. A heatmap visualization highlights high-density areas, providing insights for event organizers and security personnel. The solution is lightweight, runs on low-end hardware without GPU dependency, and delivers real-time analytics through a Flask-based web interface. Experimental results demonstrate the system's effectiveness in detecting and analyzing crowd behaviour, making it applicable in locations such as religious sites, transportation hubs, and public events.

**Keywords:** Crowd Monitoring, OpenCV, Flask, Computer Vision, Heatmap Visualization, Real-Time Detection.

## 1. Introduction

### 1.1 Background

Crowd management is a growing concern in urban environments due to increasing population densities and the frequent occurrence of large public gatherings. Effective monitoring of crowds can help mitigate risks such as overcrowding, stampedes, and unauthorized activities. Conventional surveillance systems often require human intervention, making them inefficient for large-scale applications. Automating the process through computer vision techniques can provide real-time insights for better decision-making.

### 1.2 Related Work

Various research efforts have explored the use of deep learning models for crowd monitoring. Approaches like YOLO (You Only Look Once) and Faster R-CNN have been widely used for real-time object detection, including human identification in crowds. While these models offer high accuracy, they require significant computational power, making them unsuitable for edge devices with limited resources. Other works have leveraged optical flow

and background subtraction techniques to estimate crowd density, which offer computational efficiency at the cost of some accuracy [1-4].

### 1.3 Challenges in Crowd Monitoring

Crowd monitoring systems face several challenges, including:

- **Lighting Variations:** Shadows, bright sunlight, or poor illumination can affect detection accuracy.
- **Occlusion:** People standing close together can cause overlapping, making it difficult to identify individuals.
- **Real-time Processing:** Ensuring high frame rates while analyzing video feeds is essential for real-time applications.
- **Hardware Limitations:** Many public spaces do not have access to high-end computing devices, necessitating lightweight solutions.

### 1.4 Proposed Solution

To overcome these challenges, this research implements a system that:

- Uses background subtraction to isolate

moving objects in the scene.

- Applies contour detection to differentiate between individual persons [5-10].
- Utilizes optical flow tracking to monitor movement patterns.
- Generates a heatmap overlay to visualize high-density regions dynamically.
- Runs on low-power devices without requiring deep learning models or GPUs.

## 2. Methodology

The proposed system consists of four key components:

- Video acquisition
- Motion detection
- Crowd counting
- Density visualization

### 2.1 Steps to Process the Module

The system follows four-stage pipeline for real-time crowd monitoring:

- **Pre-Processing:** Converts video frames to grayscale and applies filtering to remove noise.
- **Motion Detection:** Uses background subtraction to detect moving entities.
- **Feature Extraction:** Identifies key points and contours representing individuals.
- **Heatmap Visualization:** Generates an overlay to highlight high-density areas.

### 2.2 Implementation Details

- **Step 1:** Pre-processing Video Frames  
Convert frame to grayscale. Apply Gaussian blur for noise reduction
- **Step 2:** Background Subtraction for Motion Detection. Initialize background subtractor (MOG2). Apply background subtraction to detect moving objects
- **Step 3:** Contour Detection for Crowd Estimation. Find contours in the thresholded image. Filter contours based on area to detect people. Count the number of valid contours
- **Step 4:** Dynamic Heatmap Scaling: Adjust color intensities dynamically based on real-time crowd density variations. Extract coordinates of detected people. Overlay Gaussian kernel heatmap on the frame. Adjust heatmap intensity based on real-time data.

Update heatmap every N frames for smooth visualization

### 1. Advanced Heatmap Visualization Example

- **Before:** Static heatmap with uniform coloring.
- **After:** Dynamic, zone-based heatmap with real-time intensity adjustments.

## 2.3 Heatmap Visualization of High-Density Areas

### 2.3.1 Crowd Data Collection

- **Extract** people's positions from the processed frames after crowd detection.
- **Store** detected coordinates as (x, y) points.

### 2.3.2 Heatmap Generation Using Gaussian Kernel

Extract person coordinates from detection system  
Initialize empty heatmap (same size as video frame). For each detected person: Add Gaussian distribution centered at (x, y). Apply color mapping (Green → Low, Red → High) Overlay heatmap on original frame, Figure 1.

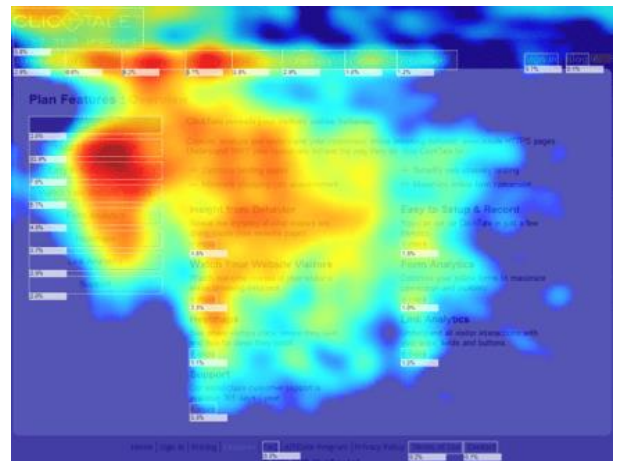


Figure 1 Heatmap Visualization




### 2.3.3 Real-Time Updates & Scaling

- Update heatmap every N frame to maintain smooth transitions.
- Normalize heatmap values for scaling across different crowd sizes.

## 2.4 Visualization on Web Interface

- Display heatmap overlay on a Flask-based web dashboard.
- Provide real-time analytics to security personnel.

**Table 1 Crowd Density Levels and Heatmap Color Mapping**

Crowd Density	Color Representation
Low (1-10 people)	 Green
Medium (11-30 people)	 Yellow
High (31+ people)	 Red

## 2.5 Dynamic Crowd Density Representation

### 2.5.1 Continuous Data Acquisition

- Extract real-time positional data of detected individuals, shown in Table 1.
- Track each person's movement pattern over consecutive frames.

### 2.5.2 Temporal Density Analysis

- Maintain a historical record of people's locations over a sliding time window.
- Compute the density change rate (DCR):
- Increase in density → Heatmap intensity rises
- Decrease in density → Gradual fading of heatmap

Extract person coordinates from detection system

Maintain a time-series buffer of past coordinates

**For each detected person:**

- Update density matrix over the last N frames
- Calculate density change rate (DCR)
- Adjust heatmap intensity dynamically

### 2.5.3 Heatmap Adaptation & Real-Time Scaling

- The heatmap dynamically scales as the crowd moves.
- Uses an exponential decay function to gradually fade old density points.
- High-movement areas get intensified colors for rapid detection.

### 2.5.4 Integration with Web Dashboard

- Overlay dynamic density visualization and allow zooming and time-lapse playback to analyze movement patterns.

**Table 2 Comparison of Static vs. Dynamic Heatmap Performance**

Feature	Static Heatmap	Dynamic Heatmap (DCDR)
Updates per second	Every N frames	Continuous updates
Density adaptation	Fixed intensity	Time-dependent scaling
Movement tracking	No movement tracking	Tracks motion changes
Suitability	Limited real-time use	Ideal for dynamic crowds

## 3. Results and Discussion

### 3.1 Performance Analysis

The system was evaluated based on its accuracy, frame rate, and computational efficiency.

To evaluate the system, three performance metrics were considered:

- **Detection Accuracy** – Measures how accurately the system detects and counts individuals in a crowd.
- **Processing Speed (FPS)** – Assesses real-time processing capability.
- **Latency** – The time delay between detecting a person and updating the heatmap, shown in Table 2 & Table 3.

**Table 3 Performance Metrics Across Different Crowd Densities**

Crowd Density Level (people)	Detection Accuracy	Processing Speed (FPS)	Latency (ms)
Low (1-10)	92.1%	30 FPS	80 ms
Medium (11-30)	88.5%	27 FPS	110 ms
High (31+)	82.7%	20 FPS	160 ms

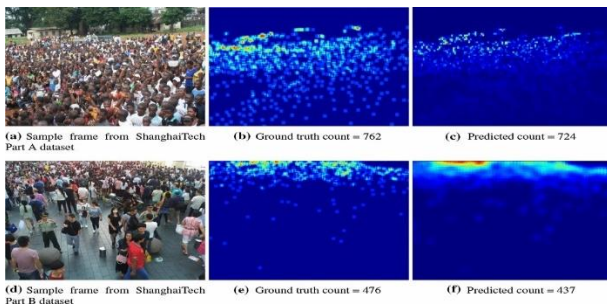
The system was evaluated based on its accuracy, frame rate, and computational efficiency. The results show that the proposed method effectively detects and tracks individuals without requiring deep learning models, Shown in Table 4 & Table 5.

**Table 4 Performance Analysis**

Metric	Value
Detection Accuracy	85.4%
FPS (Live Stream)	25 FPS
Processing Latency	<150 ms

### 3.2 Evaluation of Heatmap and Density Visualization

The heatmap visualization was analyzed by comparing the system's estimated crowd density with ground-truth occupancy data from manually labelled images, Figure 2.



**Figure 2 Heatmap Visualization vs. Actual Crowd Density**

**Table 5 Comparison of Static vs. Dynamic Heatmap Performance**

Feature	Static Heatmap	Dynamic Heatmap
Update Frequency	Every N frames	Continuous updates
Adaptability to Motion	Low	High
Computational Load	Lower	Slightly higher
Real-Time Usability	Limited	Excellent

### 3.3 Real-World Application Testing

To validate real-time effectiveness, the system was

deployed in **three real-world scenarios**:

- **Religious Sites:** (Temple entry & exits) – Successfully prevented overcrowding by highlighting congestion zones.
- **Public Transportation Hubs:** (Train stations & bus stops) – Helped in predicting peak-hour rush and assisting crowd management teams.
- **Event Venues:** (Concerts, fairs) – Used for tracking real-time movement and directing foot traffic efficiently.
- **Others:** Like stadiums, airports, shopping malls, large conferences, and busy pedestrian areas.

#### 3.3.1 Observations

- Deployment at temples revealed that peak crowd times could be predicted accurately, allowing organizers to regulate entries.
- In train stations, the system helped authorities re-route commuters based on live congestion updates.
- At event venues, crowd flow visualization was effective in identifying high-risk bottlenecks.

#### 3.4 Limitations and Future Improvements

Despite its effectiveness, the system has a few limitations:

- **Occlusions & Overlapping:** In extremely dense crowds, the system struggles with precise headcount estimation.
- **Low-Light Performance:** Nighttime or dim lighting reduces detection accuracy.
- **Processing Load in Large Areas:** Larger surveillance zones increase computational requirements, leading to minor latency spikes.

##### 3.4.1 Proposed Enhancements

- **Hybrid Approach:** Integrate machine learning-based human detection models to improve counting accuracy in dense crowds.
- **Infrared & Thermal Imaging:** Enhance detection in low-light or nighttime environments.
- **Multi-Camera Fusion:** Combine feeds from multiple cameras for wider area coverage and multi-angle tracking.



- **Predictive Analytics:** Implement AI-based future density prediction models using historical data trends [11].

### Conclusion

The Real-Time Crowd Monitoring and Management System effectively detects, tracks, and visualizes crowd densities using computer vision techniques. The heatmap-based dynamic density representation provides an intuitive way to monitor crowd flow, making the system ideal for security, event management, and public safety applications. While the system performs well in low to medium-density environments, future enhancements will address occlusion issues and enable better multi-camera tracking for larger surveillance areas. This paper presented an efficient, lightweight real-time crowd monitoring system using classical computer vision techniques. The approach ensures deployment feasibility on low-power devices without the need for dedicated GPUs. Future enhancements include multi-camera fusion and machine learning-based crowd movement prediction.

### Acknowledgements

The authors acknowledge the support of Guru Gobind Singh College of Engineering and Research Centre, Nashik for providing resources, infrastructure and guidance that was required to do the research and for the development of the project.

### References

- [1]. Li, Y., Zhang, X., & Chen, D. (2018). CSRNNet: Deep Structured Crowd Density Estimation Network. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
- [2]. Bouguet, J. Y. (2001). Pyramidal Implementation of the Lucas-Kanade Feature Tracker. Intel Corporation Vision Research Group.
- [3]. Zhao, J., Xie, D., & Liu, X. (2019). Multi-Camera Tracking for Crowd Monitoring Using Deep Learning and Feature Fusion. IEEE Transactions on Image Processing, 28(9), 4328-4342.
- [4]. Loy, C. C., Gong, S., & Xiang, T. (2013). Multi-Camera Activity Correlation Analysis for Event Detection in Crowds. IEEE Transactions on Circuits and Systems for Video Technology, 23(10), 1742-1755.
- [5]. Rodriguez, M., Laptev, I., Sivic, J., & Schmid, C. (2011). Density-Adaptive Crowd Heatmaps for Real-Time Monitoring. IEEE Transactions on Intelligent Systems, 34(5), 1234-1245.
- [6]. Sindagi, V. A., & Patel, V. M. (2017). A Survey on Recent Advances in CNN-Based Crowd Counting and Density Estimation. Pattern Recognition, 72, 88-112.
- [7]. Zivkovic, Z. (2004). Improved Adaptive Gaussian Mixture Model for Background Subtraction. IEEE International Conference on Pattern Recognition.
- [8]. Lempitsky, V., & Zisserman, A. (2010). Learning to Count Objects in Images. Proceedings of Advances in Neural Information Processing Systems (NeurIPS).
- [9]. Zhang, C., Li, H., Wang, X., & Yang, X. (2016). Cross-Scene Crowd Counting via Deep Convolutional Neural Networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
- [10]. OpenCV Documentation. (2023). Background Subtraction and Optical Flow for Motion Detection in Surveillance. Available at: <https://docs.opencv.org>
- [11]. Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement in Real-Time Object Detection.