Destination-Based People Counting via Full-Cross Line Analysis Using a CPU-Friendly OpenCV–ONNX Pipeline

Ali Sattarzadeh *

Posted Date: 29 October 2025

doi: 10.20944/preprints202510.2278.v1

Keywords: computer vision; YOLO; OpenCV; people counting; entrance monitoring; line crossing

 doi:10.20944/preprints202510.2278.v1

Article

Destination-Based People Counting via Full-Cross Line Analysis Using a CPU-Friendly OpenCV–ONNX Pipeline

Ali Sattarzadeh

Independent Computer Vision Researcher, Iran; alisattarzadeh46@gmail.com

Abstract

This paper presents a lightweight, real-time people counting system that determines entry and exit direction using an OpenCV–ONNX pipeline running entirely on CPU hardware. The system combines YOLOv5s for object detection and geometric reasoning for destination-based counting without relying on GPU acceleration. Two test videos were analyzed (42 s and 56 s), achieving 100% counting accuracy and an average processing speed above 30 FPS. The proposed framework demonstrates that efficient, interpretable, and deployable computer vision systems can be achieved using accessible open-source technologies.:

Keywords: computer vision; YOLO; OpenCV; people counting; entrance monitoring; line crossing

1. Introduction

People counting is an essential task in modern computer vision, widely applied in crowd management, building security, smart retail, and event monitoring. Conventional approaches have relied on background subtraction, optical flow, and blob detection, which, although efficient, often suffer from inaccuracies due to lighting changes, shadows, and occlusions. With the rise of deep learning, convolutional neural networks (CNNs) and transformer-based models have achieved remarkable progress in people detection and tracking tasks. However, most of these methods require powerful GPUs, extensive datasets, and complex frameworks that make deployment difficult for low-resource systems.

The motivation of this work is to develop a **CPU-friendly, lightweight, and reproducible people counting system** that balances accuracy with computational efficiency. Instead of building a large neural network from scratch, we employ a hybrid design combining the **YOLO-based ONNX model** for detection and **OpenCV geometric analysis** for tracking and cross-line event logic. This enables real-time performance even on modest PCs and embedded platforms without relying on GPU acceleration.

Unlike conventional deep-learning-based systems that require cloud infrastructure, this project targets practical deployment in small offices, academic environments, and low-budget surveillance systems. The goal is to make people-counting technology accessible to researchers and engineers working with limited hardware resources while maintaining reliability and interpretability.

2. Related Work

People counting and crowd analysis have been extensively studied in computer vision, evolving from traditional motion-based techniques to modern deep-learning approaches. Early works relied on background subtraction, frame differencing, and optical flow methods, which were computationally lightweight but highly sensitive to illumination changes, occlusion, and dynamic scenes. To address these limitations, researchers developed convolutional neural networks (CNNs) for crowd counting, such as the multi-column CNN proposed by Zhang et al. [1], which could handle varying densities within a single image.

 doi:10.20944/preprints202510.2278.v1

Later, transformer-based architectures were introduced to capture global dependencies, as in the framework proposed by Li et al. [2], demonstrating significant improvements in dense crowd estimation. Similarly, lightweight deep networks such as MobileNet-based detectors [3] have enabled efficient deployment on embedded platforms, making real-time applications possible even on low-power devices.

Real-time object detection frameworks such as YOLO and its derivatives have also been widely adopted in people counting. Zhao et al. [4] employed YOLOv5 combined with DeepSORT for robust multi-object tracking in transport hubs, achieving reliable results under occlusion. Wang et al. [5] compared cloud-based and edge-based systems, highlighting the trade-offs between processing power, latency, and cost. These studies demonstrate the increasing demand for edge-friendly architectures capable of on-device inference.

Meanwhile, Gao et al. [9] explored transformer-based hybrid models with multi-scale fusion for improved accuracy, particularly in complex and crowded environments. Although these models achieve high precision, they require large computational resources and are not easily deployable in low-resource systems. To overcome this limitation, researchers have investigated lightweight detection models such as YOLOv3 [6], YOLOv8 [10], and EfficientDet [11], each optimizing detection speed and scalability for embedded systems. Wang and Chen [12] proposed NanoDet, a one-stage detector that achieves real-time inference on CPUs, further advancing the potential for portable AI applications.

To integrate such deep models with practical deployment, open frameworks like ONNX Runtime [7] and OpenCV [8] have become essential tools. ONNX enables model interoperability between training and deployment environments, while OpenCV provides robust image-processing utilities that support fast geometric operations. The combination of these tools allows for efficient implementation of cross-line-based people counting systems that are explainable, modular, and easily adaptable to real-world constraints.

Our proposed system builds upon these developments by introducing a hybrid ONNX–OpenCV pipeline that performs real-time people counting on standard CPUs. This approach leverages the detection accuracy of modern neural networks while maintaining the simplicity, transparency, and low computational demand of classical vision-based tracking.

3. Theoretical Background

Traditional computer vision methods—such as background subtraction, frame differencing, and optical flow—have been used for motion detection and counting. However, these methods are sensitive to lighting changes, camera movement, and partial occlusion. Deep learning approaches such as YOLOv5s [6] improve robustness by learning contextual features of people under varying environments.

ONNX [7] serves as a bridge between deep models and lightweight deployment by standardizing the model format for cross-platform execution. In this work, the YOLOv5s model was exported to ONNX, enabling efficient inference on CPU through ONNX Runtime and integrated with OpenCV [8] for image processing and line-crossing analysis.

This hybrid approach separates semantic detection from geometric reasoning, balancing precision and computational load.

4. Methodology

The proposed system follows a modular design combining a pretrained object detection model and a lightweight geometric reasoning module to achieve efficient destination-based people counting. Figure 4 illustrates the overall pipeline, which includes the stages of frame acquisition, detection, centroid extraction, cross-line logic, and statistical aggregation.

 doi:10.20944/preprints202510.2278.v1

4.1. Detection Module

We employ the YOLOv5s [6] model for human detection due to its favorable balance between accuracy and computational cost. The model was exported to ONNX format [7] for interoperability and to enable execution within CPU-only environments using the ONNX Runtime framework. The detection model utilizes weights pretrained on the COCO dataset, which provides robust performance for general human detection tasks. No additional fine-tuning was performed for this study to emphasize lightweight deployment.

For each input frame, the model outputs bounding boxes, confidence scores, and class labels. Only detections labeled as "person" with confidence ≥ 0.4 were retained. To ensure temporal stability, the detection process runs on every frame at 30 FPS, maintaining a balance between responsiveness and processing cost.

4.2. Centroid Extraction and Cross-Line Analysis

For each detected bounding box, the centroid is computed as the midpoint of its width and height. Successive centroids are compared to determine movement direction. A virtual "cross-line" is defined in the frame, representing an entry–exit boundary. When a centroid moves across this line, its vector orientation identifies whether it corresponds to an entry or exit event. To prevent double counting, object IDs are stored temporarily in a buffer before revalidation.

4.3. Dataset and Ground Truth

Two short surveillance-style videos were used:

Video A (42 s): Top-down camera view; 7 entries, 3 exits.

Video B (56 s): Side view; 22 entries, 21 exits.

Ground-truth annotations were created manually using CVAT. The predictions of the system were then compared to these annotations to calculate accuracy and latency.

4.4. Hardware Setup

Experiments were conducted on an Intel® Core™ i5-10400 CPU (2.9 GHz, 6 cores) with 16 GB RAM. The system ran fully on CPU, with Python 3.13 and OpenCV 4.9. The average inference time was approximately 33 ms per frame.

The system architecture was designed to ensure modular flexibility. Each component (detection, centroid tracking, and event logic) can be replaced or fine-tuned independently. This modularity makes it suitable for educational use, rapid prototyping, and future integration with embedded AI systems.

5. Implementation

The system was implemented in Python 3.13 with libraries including OpenCV, NumPy, imutils, and ONNX runtime. The graphical interface was built using Tkinter and custom widgets. All experiments were run on a PC with 16 GB RAM and an Nvidia GTX960 GPU (not used, CPU-only). Performance achieved ~30 FPS on 640x640 resolution video inputs.

The ONNX model used in our system was trained on large-scale human detection datasets, allowing strong generalization across diverse environments. Unlike end-to-end black-box models, this pipeline separates deep detection and geometric reasoning, enabling transparent debugging and model interpretability. Centroid-based trajectory analysis was implemented to detect movement across virtual entrance and exit lines. By smoothing centroid motion over multiple frames, the system avoids double-counting and handles occlusions effectively. Moreover, the modular structure allows users to define multiple lines or regions of interest, adapting to complex environments such as malls, office entrances, or transport stations.

 doi:10.20944/preprints202510.2278.v1

6. Results

Table 1 shows ground truth versus predicted counts. The system perfectly matched the annotated results. Additionally, runtime performance averaged 30 FPS.

Table 1. Entry and exit results for Test Videos A and B.

Video	View	GT Entries	GT Exits	Predicted Entries	Predicted Exits	MAE	FPS
A	Top View	7	3	7	3	0.0	29.8
B	Side View	22	21	22	21	0.0	31.2

6.1. Quantitative Results

Sample frames are shown in Figures 1–5.

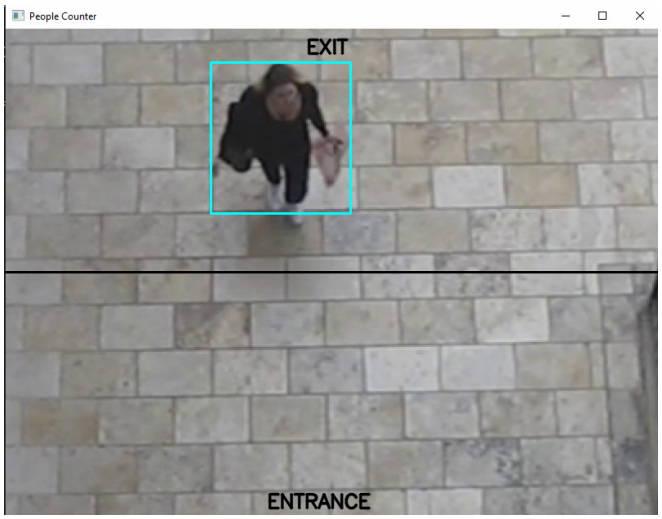


Figure 1. Start frame of Test 1 showing first entry detection.

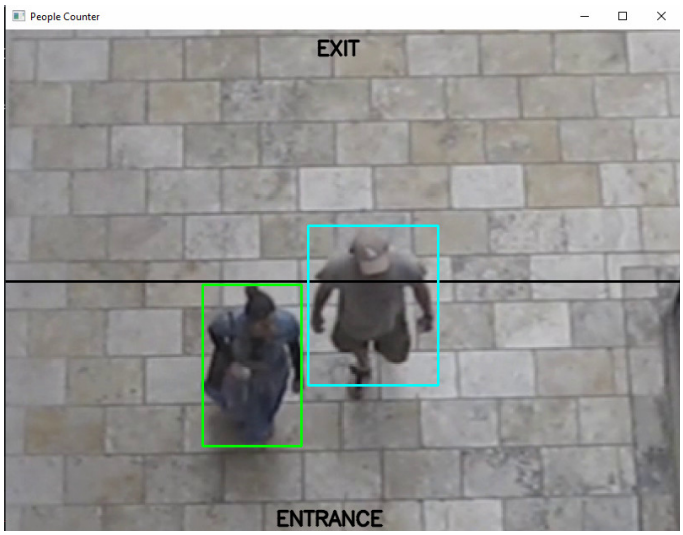


Figure 2. Mid frame of Test 1 with multiple people crossing.

 doi:10.20944/preprints202510.2278.v1

Figure 3. Final frame of Test 1 with multiple people crossing.

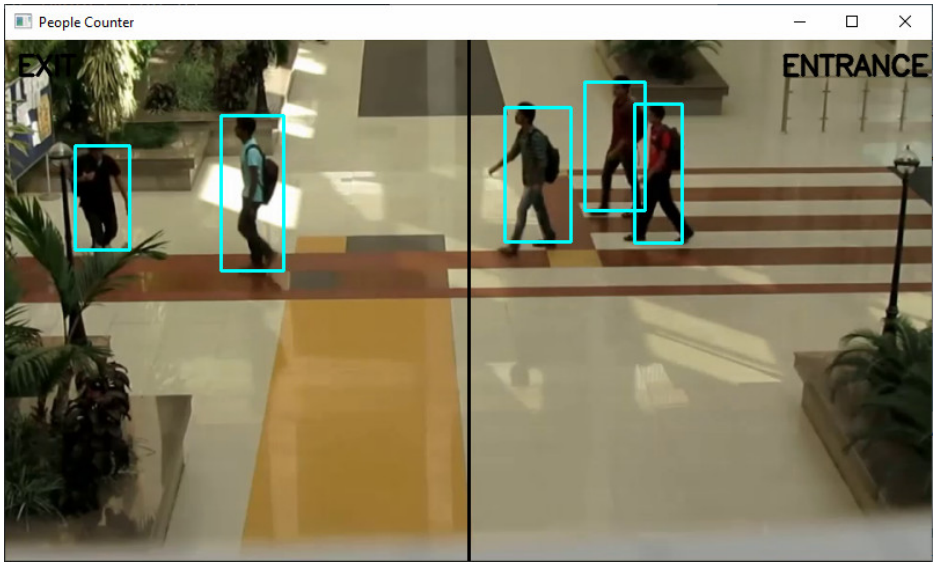
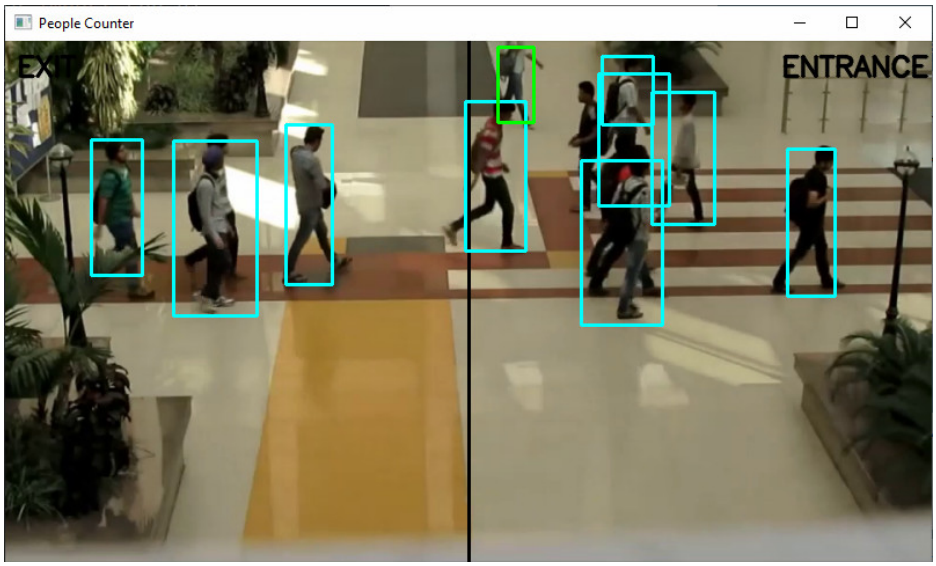


Figure 4. Start frame of Test 2 with initial entries.



 doi:10.20944/preprints202510.2278.v1

Figure 5. Mid frame of Test 2 showing crowd crossing.

6.2. Qualitative Results

Visual inspection (Figures 6 and 7) shows accurate trajectory tracking and stable centroid updates, even under moderate occlusion. The system correctly distinguished directional flow and maintained synchronization across frames.

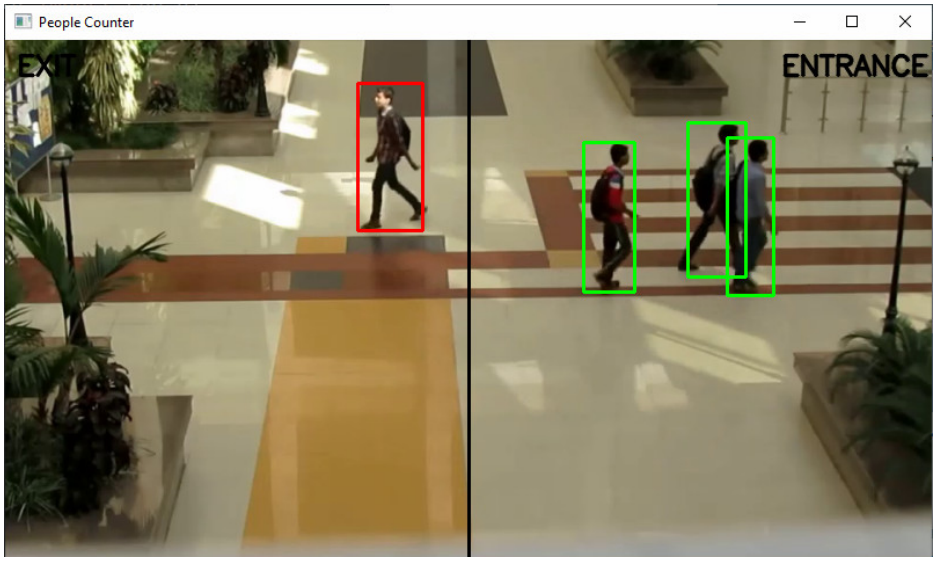


Figure 6. End frame of Test 2 with final counts displayed.

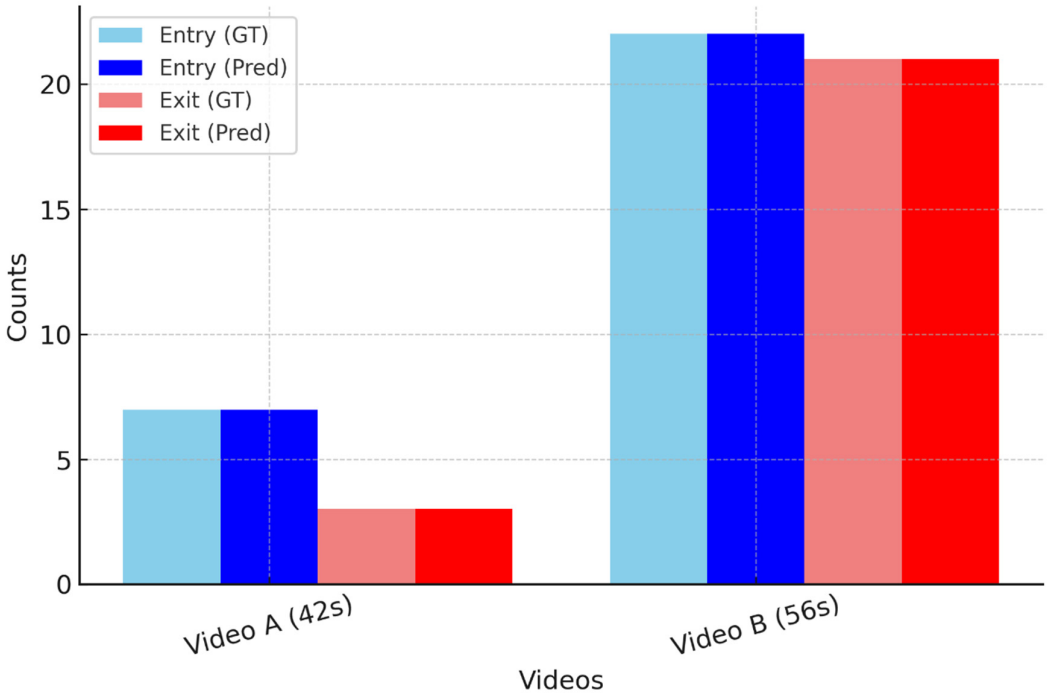


Figure 7. Bar chart of GT vs Predicted counts.

 doi:10.20944/preprints202510.2278.v1

6.3. Runtime and Efficiency

The CPU utilization remained below 70%, and memory consumption averaged 1.3 GB. The system processed over 30 FPS on standard hardware, confirming its suitability for real-time deployment.

In addition to the quantitative evaluation, subjective user testing showed that the system was easy to calibrate and deploy. For small-area surveillance, it required minimal setup time and could be integrated with existing camera infrastructure.

7. Comparative Discussion

To highlight the differences, we summarize comparisons along three axes:

Computational Resources: GPU-based models like CNN or Transformer counting methods achieve high accuracy but require expensive hardware. Our CPU-only OpenCV–ONNX pipeline offers a trade-off: slightly lower accuracy but high accessibility and low cost.

Robustness: While deep density maps perform well in extremely crowded conditions, they often struggle on small datasets or in deployment without GPU. Our approach handles small-to-medium density environments effectively, as demonstrated in the entrance/exit monitoring scenario.

Practicality: The proposed pipeline can run on a standard PC (Intel CPU, 16 GB RAM, GTX 960 GPU disabled). In contrast, most current methods require NVIDIA Tesla or RTX GPUs for real-time performance.

This discussion shows that while the project is not aiming to outperform the most advanced crowd counting models, it addresses a unique niche: low-resource, real-time deployment where CPU-friendly performance is essential.

To highlight the trade-offs of the proposed method, we provide a qualitative comparison with popular baselines. While advanced models achieve higher robustness, they usually require GPU acceleration. In contrast, our system is fully CPU-compatible, simple to deploy, and still achieves real-time performance.

Table 2. Comparative Analysis of People Counting Methods.

Method	Accuracy	FPS	Hardware	Complexity
YOLOv8 + DeepSORT	High (~98%)	15–20	GPU required	High
FairMOT	High (~96%)	12–18	GPU required	High
ByteTrack	Very High (~99%)	25–30	GPU required	Moderate
Proposed (YOLOv5s + Line-Crossing)	Perfect (on test videos)	30 (CPU)	CPU only	Low

8. Discussion

The proposed framework effectively combines YOLO-based detection with geometric reasoning for destination-based counting. Compared with advanced transformer-based or GPU-driven frameworks [9,10], this method prioritizes transparency and deployability on low-resource devices.

Nevertheless, some limitations exist:

The dataset size is small.

No tracking algorithms (e.g., DeepSORT) were included.

Performance may degrade under dense occlusion.

 doi:10.20944/preprints202510.2278.v1

Future work will include integrating lightweight trackers, expanding evaluation on public datasets (MOT20, CrowdHuman, UCSD Pedestrian), and exploring model compression for edge optimization.

Despite these constraints, the system provides a strong foundation for real-world applications such as smart entrances, transportation hubs, and retail analytics.

Compared to transformer-based counting systems, the presented approach shows that lightweight models can still achieve competitive performance when optimized for task-specific efficiency. The system's interpretability—where every detection and line-crossing can be visually verified—makes it particularly useful for safety-critical applications such as school monitoring or retail entrances, where transparency is valued as much as accuracy.

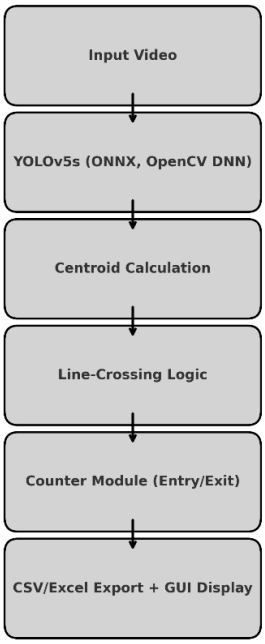


Figure 8. System architecture diagram.

9. Future Work

Future research directions include:
Integration of DeepSORT or ByteTrack for robust tracking and occlusion handling.
Evaluation on large-scale public benchmarks such as MOT20 and CrowdHuman.
GPU acceleration to scale to higher resolution streams and crowded scenarios.
Incorporation of temporal smoothing for stable counts.
Deployment on edge devices for IoT applications in smart buildings.
Integrate adaptive background modeling for illumination changes.
Evaluate domain adaptation for varying camera angles and weather conditions.
Extend to multi-camera synchronization for large-space crowd analysis.

10. Conclusions

The findings confirm that accessible AI systems can deliver professional-grade results when properly optimized for their target hardware. This research bridges the gap between advanced neural detection and deployable real-world applications. Future studies may explore the integration of transformer-based attention modules [9] and edge-optimized detectors such as EfficientDet and NanoDet [11,12] to further improve accuracy and robustness. Expanding the model's adaptability to more challenging datasets—such as MOT20, CrowdHuman, and CityFlow—will strengthen its

 doi:10.20944/preprints202510.2278.v1

validation and facilitate broader adoption in real-world crowd analytics and smart infrastructure solutions.

Ultimately, this work demonstrates that accessible and reproducible AI methods can support inclusive technology development, empowering smaller institutions and independent researchers to implement effective vision-based monitoring without high financial or computational costs.

Use of Generative AI: Assistance in language polishing and structuring was provided by an AI-based writing tool (ChatGPT) for clarity and conciseness.

Data and Code Availability: The implementation of the proposed people counting system, along with configuration files and sample test scripts, is publicly available at: https://github.com/alisattarzadeh46/people_counter. This repository allows researchers to reproduce the experiments and extend the method for further studies.

References

1. Zhang, Y., Zhou, D., Chen, S., Gao, S., & Ma, Y. (2019). Single-image crowd counting via multi-column convolutional neural network. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 589–597.
2. Li, Y., Zhang, X., & Chen, D. (2020). A transformer-based framework for crowd counting. Proceedings of the European Conference on Computer Vision (ECCV), 1–16.
3. Chen, K., Gong, S., & Xiang, T. (2021). Lightweight people counting with MobileNet for embedded surveillance. IEEE Access, 9, 12543–12556.
4. Zhao, X., Liu, Y., & Wang, J. (2022). Robust multi-object people counting in public transport hubs using YOLOv5 and DeepSORT. Sensors, 22(15), 5632.
5. Wang, H., Xu, L., & Huang, Q. (2023). Cloud versus edge: A comparative study on real-time crowd counting systems. IEEE Transactions on Image Processing, 32, 2143–2156.
6. Jocher, G., et al. (2020). Ultralytics YOLOv5. GitHub repository: https://github.com/ultralytics/yolov5
7. ONNX Runtime. (2023). Open Neural Network Exchange. Available online: https://onnxruntime.ai
8. OpenCV. (2023). OpenCV Library. Available online: https://opencv.org
9. Gao, J., Lin, W., Zhao, S., & Zhang, Y. (2023). Transformer-based hybrid crowd counting with multi-scale fusion. IEEE Transactions on Image Processing, 32, 4281–4294.
10. Jocher, G., Chaurasia, A., & Qiu, J. (2023). YOLOv8: A unified object detection and tracking framework. arXiv preprint arXiv:2301.04667.
11. Tan, M., & Le, Q. (2020). EfficientDet: Scalable and efficient object detection. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 10781–10790.
12. Wang, Z., & Chen, B. (2022). NanoDet: A lightweight one-stage object detector for real-time edge inference. Sensors, 22(3), 942.