Crowd Surveillance using YOLOv8 and BoT-SORT Tracking

Harsh Vardhan Sharma (B23373)
Akshat Jha (B23336)

IIT Mandi
AR–522 Robot Vision Project

December 9, 2025

Outline

Why Crowd Surveillance?

- Dense public spaces increasingly rely on automated monitoring.
- Manual surveillance is not scalable or reliable.
- Applications:
  - Public safety and behavioural analysis.
  - Robotics and autonomous navigation.
  - Transportation and crowd-flow analytics.
- Key technical challenges:
  - Heavy occlusion.
  - High density and small targets.
  - Real-time inference constraints.

Mid-Term Progress: Head Detection

- **Previous Approach**:
  - Model: **YOLOv8n** (Nano).
  - Dataset: **JHU-Crowd++** (Crowd counting dataset).
  - Target: Single class `head`.
- **Why we switched to Body Detection (MOT20)?**
  - **Tracking Stability (IoU)**: Head boxes are too small; slight movements cause near-zero Intersection-over-Union (IoU) between frames, breaking motion-based trackers like BoT-SORT.
  - **Occlusion Robustness**: A head is easily completely hidden. A full body offers more surface area; if the head is blocked, the torso often remains visible.
  - **Context**: Full-body bounding boxes are more intuitive for human surveillance and crowd flow visualization than floating heads.

Complete Training Pipeline

1. Input video / MOT20 frames.
2. Frame extraction.
3. YOLOv8s inference using fine-tuned weights.
4. YOLO $\rightarrow$ MOT-format conversion.
5. BoT-SORT-style tracking on per-frame detections.
6. Annotated output video with IDs and bounding boxes.

One-click notebook

A Colab-based one-click runner automates this entire pipeline end to end.

MOT20 Dataset

- One of the most challenging MOT datasets.
- Contains extremely dense pedestrian scenes.
- 4 train + 1 validation sequence (MOT20-04) + 3 test sequences (high-resolution videos).
- Strong occlusion, clutter, and scale variation.

Sequences used for training

MOT20-01, MOT20-02, MOT20-03, MOT20-05

Sequences used for testing

MOT20-06, MOT20-07, MOT20-08

Conversion to YOLO Format

- Original MOT20 annotations: `gt.txt` (MOTChallenge format).
- We developed a custom conversion script:
  1. Read all ground-truth bounding boxes.
  2. Filter out detections with *confidence* < 0.5 .
  3. Convert to YOLO normalised format:

$$(x_c, y_c, w, h) \in [0, 1]$$

  4. Save one `.txt` label file per frame.
- A clean YOLO dataset structure was created for Ultralytics training.

Core Approach: Transfer Learning (Fine-tuning)

- Base model: **YOLOv8s** pretrained on COCO.
- We fine-tuned it to specialise only on:

```
class = person
```

- Transfer learning allowed:
  - Faster convergence.
  - Higher accuracy with less data.
  - Better generalisation to crowded scenes.

Training Configuration ("The Recipe")

- **Image size**: 640×640
- **Batch size**: 4 (fits Tesla T4 GPU)
- **Total epochs**: 50
- **Optimizer**: AdamW
    - Learning rate: 0.002
    - Momentum: 0.9
- **Device**: NVIDIA Tesla T4 (Google Colab)

Single-class detection

YOLO was retrained to detect only "person", making it highly specialised.

Data Augmentation Pipeline

- To improve robustness to CCTV-style noise and lighting:
  - **Blur** and **MedianBlur** – handles motion blur and low-quality frames.
  - **ToGray** – robustness to grayscale / low-colour environments.
  - **CLAHE (Contrast Limited Adaptive Histogram Equalization)** – enhanced contrast in dark or overexposed scenes.
- These augmentations were auto-applied by Ultralytics during training.

Final Model Outputs

- Fully fine-tuned YOLOv8s pedestrian detector.
- Two output formats:
  - PyTorch model: `best.pt`
  - ONNX model: `best.onnx`
- Hyper-specialised for dense pedestrian scenes in MOT20-style environments.

Validation Metrics (MOT20-04)

At epoch 50, on the validation sequence (MOT20-04), the model achieved:

- **mAP@50**: 0.982
- **mAP@50–95**: 0.837
- **Precision**: 0.986
- **Recall**: 0.956
- **Fitness score**: 0.837

Interpretation

The detector is quite accurate and reliable even under dense, occluded conditions.

Testing and Analysis Pipeline

1. **Input**: Injection of novel test video (CCTV/Drone footage).
2. **Core Inference**: YOLOv8 Detection $\rightarrow$ BoT-SORT Tracking.
3. **Temporal Analytics**: Frame-by-frame occupancy counting and dwell time estimation.
4. **Spatial Analytics**: Density heatmaps, velocity flow fields, and trajectory mapping.
5. **Event Logic**: Zone entry/exit monitoring and social distancing violation checks.
6. **Output**: Annotated video render, CSV statistical logs, and visualization plots.

Automated Insight Generation

The pipeline processes raw tracking data into actionable insights (Heatmaps, Dwell Times, Flow) automatically.

Results

Results on MOT20:

- Accurate detection even in highly crowded scenes.
- Good ID continuity when occlusion is low to moderate.
- Some ID switches still occur in extreme overlap and moderate to high occlusions.

Tested on Real-world Videos:

- Indoor corridors and halls.
- Outdoor footpaths and streets.
- CCTV-style surveillance videos.

Model generalised well despite being trained only on MOT20.

Current Limitations

- YOLO still struggles in extremely dense, fully packed crowds.
- Our tracking uses only motion and spatial information:
  - No appearance features $\rightarrow$ more ID switches.
- No quantitative MOT metrics (MOTA, IDF1) computed yet.

Future Improvements

- **Better detectors for very dense crowds**
  - Faster R–CNN or Mask R–CNN on MOT20.
  - Aim for better recall under heavy occlusion.
- **Advanced tracking**
  - DeepSORT with appearance embeddings.
- **High-level analytics**
  - Crowd density estimation.
  - Abnormal behavior and anomaly detection.

Live Demonstration

- Show the Colab notebook workflow:
  - Frame extraction.
  - Detection with fine-tuned YOLOv8s.
  - Tracking and video rendering.
- Play the final annotated output video:
  - Highlight stable IDs and dense detections.
  - Point out a few typical failure cases as well.

Testing Instructions

We have provided a Google Colab notebook for easy reproduction of our results.

Access the Project

[Click here to open Google Drive Folder]

File: `robot_vision_crowd_surveillance_demo_1.ipynb`

How to test with your own video:

1. Open the notebook in Google Colab.
2. Execute all cells to run the pipeline. (Click Run All)
3. Upload your test video in **Cell 4** when prompted.
4. Once execution completes, download the output from: `/content/robo_pipeline/output_videos`
5. View the processed video locally.

Conclusion

- **High-Density Specialisation**: Successfully engineered a surveillance pipeline capable of tracking pedestrians in extreme density scenarios (avg. 149 people/frame).
- **Technical Execution**:
  - Fine-tuned YOLOv8s via transfer learning, adapting specifically to MOT20 occlusion patterns.
  - Integrated BoT-SORT to ensure tracking stability under camera motion.
- **Performance**: Achieved **98.2% mAP@50** and **83.7% Fitness**, demonstrating exceptional precision in crowded environments.
- **Deliverable**: Built an automated end-to-end tool that converts raw CCTV footage into actionable analytics (Heatmaps, Flow, Dwell Time).