Person Tracking System Technical Report

Hamza Abdelmoreed

May 9, 2025

1 Introduction

The project aims to track multiple people in a video and represent their walking path on the floor using bounding rectangles and colored the distance. The method utilizes YOLOv8 for detection, ByteTrack for tracking, and OpenCV for visualization.

- Detect all persons in video frames
- Assign unique persistent IDs to each person
- Track movements across frames
- Visualize paths and identities

2 Model Selection

2.1 Detection Model: YOLOv8n

Feature	**Description**
Choice Rationale	YOLOv8 offers excellent speed-accuracy tradeoff
Variant Selection	Nano version (yolov8n) chosen for real-time performance
Performance	$\sim$80 FPS on modern GPU, $\sim$12 FPS on CPU (varies by hardware)
Accuracy	37.3 mAP on COCO dataset (general purpose)

2.2 Tracking Algorithm: ByteTrack

Feature	Description
Choice Rationale	Balances performance and accuracy
Advantages	• Handles occlusion well • Maintains IDs effectively • Works with low-confidence detections
Implementation	Through supervision library

3 System Architecture

1. **Input Layer**: Video stream reader
2. **Detection Layer**: YOLOv8 person detection
3. **Tracking Layer**: ByteTrack ID assignment
4. **Visualization Layer**: Bounding boxes, IDs, and paths
5. **Output Layer**: Processed video writer

4 Key Features

- **Persistent Identification**: Maintains consistent IDs across frames
- **Movement Visualization**: Shows historical paths
- **Efficient Storage**: Only stores necessary tracking data
- **Customizable Visualization**: Unique colors per track

5 Performance Considerations

- **Memory**: Stores only tracking points, scales linearly with number of persons
- **Speed**: Dominated by YOLO detection (tracking is lightweight)
- **Accuracy Limitations**: Depends on detection quality, may struggle with:
  - Heavy occlusion
  - Similar-appearing persons
  - Fast movements

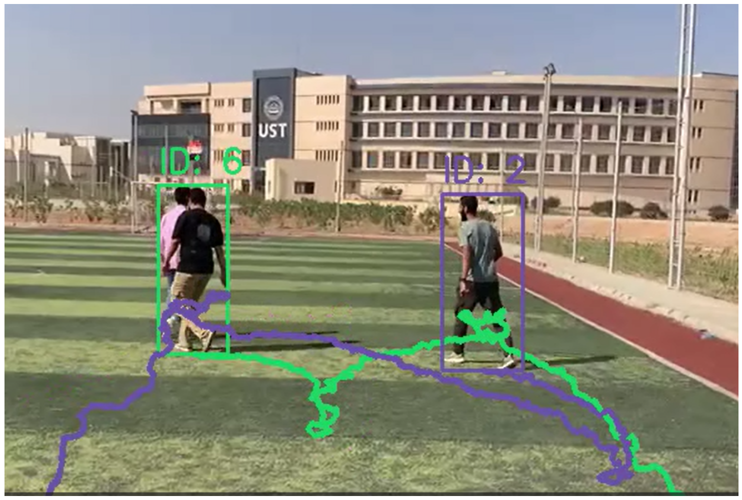

Figure 1: hard overlap

6 Potential Improvements

1. **Model**: Upgrade to YOLOv8s/m for better accuracy
2. **Tracking**: Add ReID features for better occlusion handling
3. **Visualization**: Add speed/direction indicators
4. **Optimization**: Implement batch processing for faster throughput

7 Conclusion

This implementation provides a solid foundation for person tracking in videos, suitable for applications like:

- Surveillance systems
- Crowd analysis
- Retail analytics
- Security monitoring

The modular design allows for easy swapping of detection models or tracking algorithms as needed for specific use cases.

Appendix: Code Implementation

Listing 1: Main Tracking Implementation

```
import cv2
import numpy as np
from ultralytics import YOLO
import supervision as sv
import uuid
from collections import defaultdict
import random

model = YOLO('yolov8n.pt')
trackers = defaultdict(list)
colors = {}

def generate_random_color():
    return (random.randint(0, 255), random.randint(0, 255), random.randint(0, 25

def process_video(input_path, output_path):
    cap = cv2.VideoCapture(input_path)
    if not cap.isOpened():
        print("Could not open video.")
        return

    width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    fps = int(cap.get(cv2.CAP_PROP_FPS))
    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    out = cv2.VideoWriter(output_path, fourcc, fps, (width, height))

    tracker = sv.ByteTrack()

    frame_count = 0
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        results = model(frame)[0]
        detections = sv.Detections.from_ultralytics(results)
        person_mask = detections.class_id == 0
        person_detections = detections[person_mask]
        tracked_detections = tracker.update_with_detections(person_detections)
```

```
    for track_id, box in zip(tracked_detections.tracker_id, tracked_detectio
        if track_id not in colors:
            colors[track_id] = generate_random_color()

        x1, y1, x2, y2 = box
        bottom_center_x = int((x1 + x2) / 2)
        bottom_center_y = int(y2)
        trackers[track_id].append((bottom_center_x, bottom_center_y))

        color = colors[track_id]
        cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), color,
        cv2.putText(frame, f"ID: {track_id}", (int(x1), int(y1) - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.9, color, 2)

        for i in range(1, len(trackers[track_id])):
            if trackers[track_id][i] is None:
                continue
            cv2.line(frame, trackers[track_id][i-1], trackers[track_id][i],

    out.write(frame)
    frame_count += 1
    print(f"Processing frame {frame_count}/{total_frames}")

cap.release()
out.release()
cv2.destroyAllWindows()
print(f"Output video saved as {output_path}")
```