VIDEO OBJECT DETECTION AND TRACKING PIPELINE

TEAM: Bhanu Prakash Vangala, Nolan Rink

GITHUB REPOSITORY: https://github.com/bhanuprakashvangala/VideoPipeline

This report documents the implementation of a video object detection and tracking pipeline for Milestone 1. The system processes the MOT17-04-DPM dataset, consisting of 1050 frames, implementing three pipeline configurations using different detection models (DETR, YOLOv8, FasterRCNN) combined with OC-SORT tracking. All configurations achieved tracking accuracies between 99.997% and 100%.

1. PIPELINE OVERVIEW AND OBJECTIVE

The primary objective of this pipeline is to implement, evaluate, and optimize an end-to-end machine learning system for detecting and tracking objects in video sequences. The pipeline processes video data frame-by-frame to identify persons and maintain their identities across consecutive frames, enabling continuous tracking of multiple individuals throughout a video sequence. This system is designed to handle real-world scenarios with multiple moving objects, occlusions, and varying lighting conditions.

Architecture

The pipeline implements a two-stage architecture:

Stage 1: Object Detection

- Processes individual frames to detect persons
- Returns bounding boxes with confidence scores
- Filters detections using 0.3 confidence threshold

Stage 2: Object Tracking

- Uses OC-SORT algorithm for temporal association
- Maintains track identities across frames
- Implements Kalman filtering for motion prediction

2. DATASET AND MODELS

Dataset: The MOT17-04-DPM dataset from the Multi-Object Tracking benchmark is used. It contains 1,050 sequential frames of an urban street scene at 1920x1080 resolution, providing challenging conditions with occlusion and lighting variation.

- Total frames: 1,050
- Resolution: 1920×1080 pixels
- Content: Urban street scene with multiple pedestrians
- Format: Sequential JPEG images representing video frames
- Target class: Person detection only

Detection Models Implemented:
- DETR (facebook/detr-resnet-50): Transformer-based, global reasoning.
- YOLOv8-nano: Real-time, optimized for speed and low latency.
- FasterRCNN (resnet50_fpn): Two-stage detector with high accuracy.

Tracking Algorithm:
- OC-SORT (Observation-Centric SORT):

1. - Detection threshold: 0.3
2. - Max age: 30 frames
3. - Min hits: 3 detections
4. - IoU threshold: 0.3
5. - Distance threshold: 100 pixels

Assertions and Metrics

Distance-based Assertion: Objects with the same track ID must not move more than 100 pixels between consecutive frames

Implemented Metrics:

- Tracking Accuracy: Valid transitions / Total transitions
- Processing Latency: Total time for 1050 frames
- Detection Count: Total objects tracked
- Confidence Scores: Average detection confidence

3. PIPELINE CONFIGURATIONS

We implemented 3 configurations.

Implemented Configurations:
- Configuration 1: DETR + OC-SORT
- Configuration 2: YOLOv8 + OC-SORT
- Configuration 3: FasterRCNN + OC-SORT

4. PERFORMANCE RESULTS

Configuration	Model	Latency (s)	Objects Tracked	Tracking Accuracy	Avg Confidence
Config 1	DETR + OC-SORT	234.03	53,922	1.000 (53,570/53,570)	0.714
Config 2	YOLOv8 + OC-SORT	67.74	26,594	1.000 (26,438/26,438)	0.545
Config 3	FasterRCNN + OC-SORT	130.01	65,096	1.000 (64,642/64,644)	0.664

Analysis:
- YOLOv8 achieved the fastest processing (67.74s, ~15.5 FPS).
- DETR was slowest (234.03s, ~4.5 FPS).
- FasterRCNN provided moderate speed (130.01s, ~8.1 FPS).
- Detection coverage highest with FasterRCNN (65,096 objects).
- All models achieved near-perfect tracking accuracy.

Latency Results:

- Fastest: YOLOv8 at 67.74 seconds (15.5 FPS average)
- Slowest: DETR at 234.03 seconds (4.5 FPS average)
- Moderate: FasterRCNN at 130.01 seconds (8.1 FPS average)

Detection Coverage:

- Most comprehensive: FasterRCNN with 65,096 total detections
- Moderate coverage: DETR with 53,922 detections
- Lowest coverage: YOLOv8 with 26,594 detections

Tracking Quality:

- DETR: Perfect accuracy (100%)
- YOLOv8: Perfect accuracy (100%)
- FasterRCNN: Near-perfect accuracy (99.997%, 2 invalid transitions out of 64,644)

Confidence Analysis:

- Highest average: DETR (0.714)
- Lowest average: YOLOv8 (0.545)
- Moderate: FasterRCNN (0.664)

5. IMPLEMENTATION

The implementation is organized as a Jupyter notebook with sections for setup, dataset loading, model initialization, pipeline execution, performance evaluation, and results visualization. Enhancements included NumPy 2.x compatibility fixes, GPU acceleration with CUDA, robust error handling, and progress monitoring. Reproducibility is ensured with Docker containerization, pinned dependencies, and random seed settings.

6. TEAM ORGANIZATION AND COLLABORATION

The team consists of two members:
- Bhanu : Focused on DETR and Faster RCNN implementation and transformer-based approaches.
- NOLAN : Implemented YOLOv8 pipeline and optimized for real-time performance.

All members contributed to tracker development and evaluation. Collaboration occurred via GitHub for code sharing, Microsoft Teams for real-time communication, and Google Drive for document sharing. Issues such as NumPy compatibility and GPU memory limitations were resolved collaboratively.

7. DELIVERABLES

Milestone 1 Deliverables:
- Jupyter Notebook: complete_video_pipeline.ipynb with pipeline configurations.
- Dockerfile and docker-compose.yml for containerized reproducibility.
- Report: This document.

- Results: pipeline_comparison_results.csv and tracking_results.json.
All deliverables are available on the GitHub repository.

8. Docker Environment

We provide a containerized setup to ensure consistent results across machines. The Dockerfile pins PyTorch 2.5.1 with CUDA 12.4 and related deps; docker-compose.yml maps port 8888, mounts ./notebooks and ./results for persistence, and enables NVIDIA GPUs when available. Typical use: docker compose build then docker compose up and open Jupyter at http://localhost:8888, running complete_video_pipeline.ipynb. For sharing, the image is tagged nolvic/video-pipeline:latest and can be pushed/pulled via Docker Hub to run the same pipeline on other hosts.

9. CONCLUSIONS

The video object detection and tracking pipeline successfully demonstrates three distinct configurations with varying trade-offs between speed and detection coverage. The comprehensive evaluation of the three pipeline configurations reveals distinct performance characteristics across multiple metrics. YOLOv8 + OC-SORT emerged as the fastest processing pipeline, completing the entire 1,050-frame sequence in just 67.74 seconds, achieving approximately 15.5 frames per second processing speed. DETR + OC-SORT demonstrated superior tracking accuracy with a perfect score of 1.0 and the highest average confidence level of 0.714, though at the cost of the longest processing time at 234.03 seconds. FasterRCNN + OC-SORT distinguished itself by detecting the most objects with 65,096 total detections, providing the most comprehensive coverage with moderate processing speed of 130.01 seconds.

The implementation successfully demonstrates that the OC-SORT tracking algorithm effectively handles object association and tracking across all three detection models. All pipelines achieved exceptional tracking accuracy between 99.997% and 100%, with the 100-pixel distance threshold proving appropriate for the dataset characteristics. The tracking system maintained 352 unique tracks for DETR, 156 for YOLOv8, and 452 for FasterRCNN, with average track lengths ranging from 144 to 170 frames, indicating robust temporal consistency. The processing of 1,050 frames from the MOT17 dataset provides a robust evaluation framework that validates the pipeline's performance under realistic urban pedestrian tracking conditions.