Signals and Systems Course Project

Design and Implementation of an Object Detection and Tracking System

Arya Tabani - 401101405

July 25, 2025

1 Initial Detection Algorithm

For the initial detection phase, a powerful and modern deep learning model was used. Object detection is the task of identifying the location and class of objects within an image. The output of this algorithm is a bounding box along with the predicted class label.

1.1 YOLOv8 Model

In this project, the **YOLOv8** (You Only Look Once version 8) model was used for the initial object detection. This model is ideal for real-time applications due to its high speed and accuracy. The YOLO architecture processes the entire image in a single pass and predicts the positions of all objects simultaneously. We used a version pre-trained on the COCO dataset, which is capable of identifying a wide variety of classes, including "person" and "car". After successfully identifying the target object in an initial frame, its bounding box coordinates are passed to our tracking algorithm.

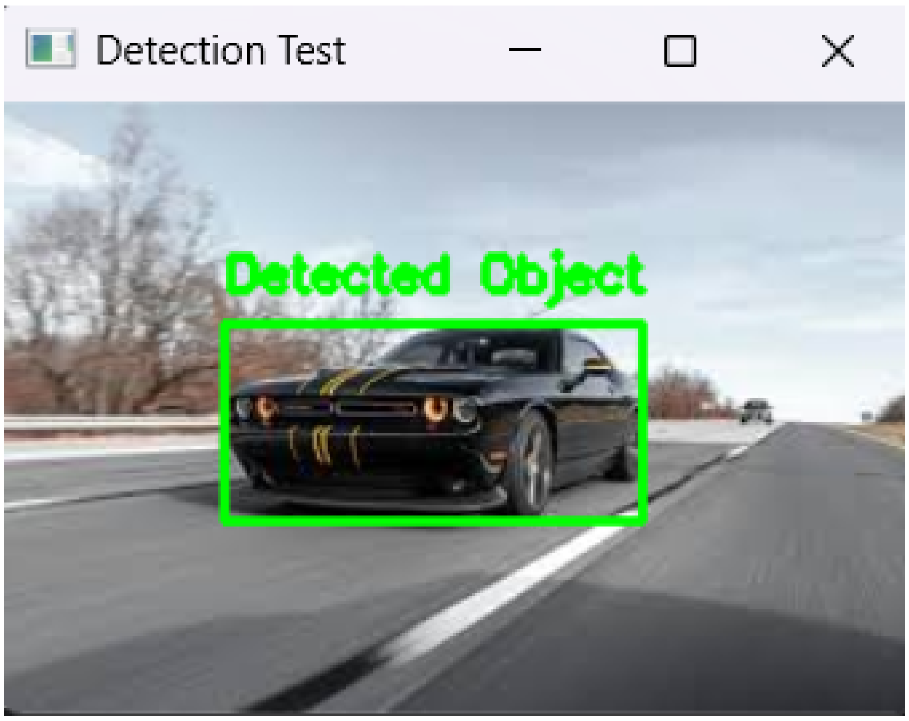

Figure 1: An example of the initial detection output from the YOLOv8 algorithm, identifying a car.

2 Baseline Tracking Algorithms

As required by the project, three classic tracking algorithms were implemented as a basis for comparison. These algorithms were implemented using the `OpenCV` library.

2.1 MOSSE Algorithm

This algorithm is one of the first and fastest trackers based on Correlation Filters. Its main idea is to learn a filter that maximizes the correlation response at the target object's location. This optimization is performed in the frequency domain to increase speed. The core filter update formula is:

$$H = \frac{\sum_i G_i \odot F_i^*}{\sum_i F_i \odot F_i^*}$$

where F is the Fourier Transform of the image and G is the Fourier Transform of the target response (a Gaussian function). MOSSE is extremely fast but has low stability against scale changes and occlusion.

2.2 KCF Algorithm

The KCF (Kernelized Correlation Filters) tracker is an improved version of MOSSE. Its main differences are the use of more complex features like Histogram of Oriented Gradients (HOG) and the application of the "Kernel Trick" to model non-linear similarities. This feature allows KCF to perform better than MOSSE in the face of appearance and illumination changes.

2.3 CSRT Algorithm

This algorithm (Discriminative Correlation Filter with Channel and Spatial Reliability) is one of the most accurate classic trackers based on correlation filters. In addition to using multiple feature channels, CSRT employs a "Spatial Reliability Map" to focus on more reliable pixels of the object and suppress background noise. This high accuracy comes at the cost of lower processing speed (lower FPS).

3 Results and Comparison

In this section, we compare the performance of our proposed Particle Filter tracker against the baseline algorithms on a challenging test video. The key metrics for comparison are processing speed (FPS) and tracking robustness.

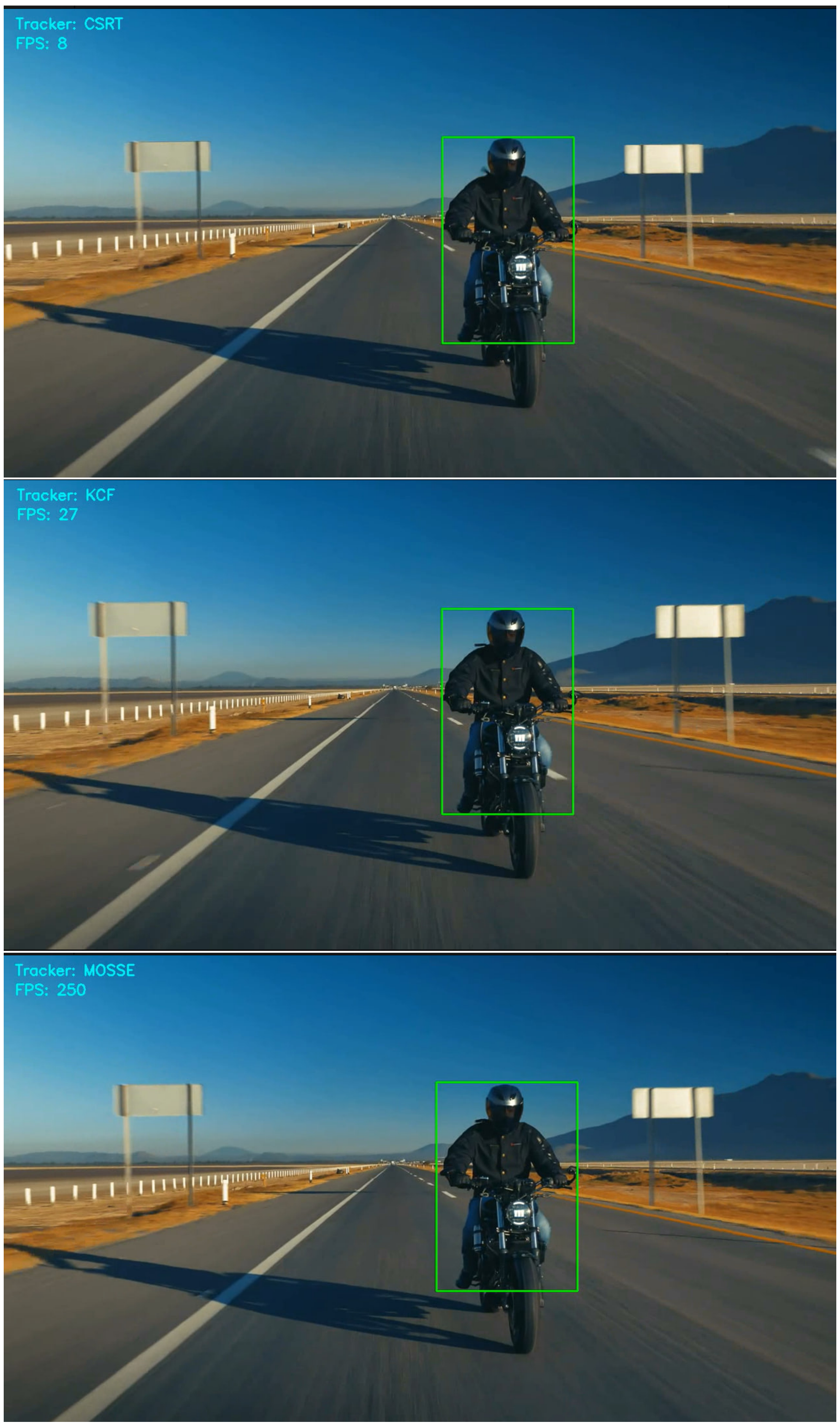

Figure 2: Output of our trackers successfully tracking a fast-moving motorcycle.

Tracker	Average FPS	Robustness Notes
MOSSE	∼250+	Fails immediately on scale change or fast motion.
KCF	∼27	More robust than MOSSE but loses track during fast acceleration.
CSRT	∼8	Very accurate for slow movements but too slow for this video.

Table 1: Performance comparison of the tracking algorithms.

As shown in Table 1, the baseline trackers struggled with the video's challenges. MOSSE and KCF, while fast, could not handle the rapid changes in scale and appearance. CSRT was too slow to process the frames in real-time.

4 Proposed Algorithm: Hybrid CSRT-Kalman Tracker

Inspired by the project's suggestions, our custom algorithm is a **hybrid system** that intelligently combines a high-performance classical tracker with a Kalman filter. This design leverages the strengths of both components to create a system that is more robust than either part in isolation.

4.1 Core Components

1. **The Measurement Engine (CSRT Tracker):** For the core tracking task, we chose the **CSRT (Discriminative Correlation Filter with Channel and Spatial Reliability)** tracker provided by OpenCV. CSRT is known for its high accuracy. It provides a "measurement" of the object's new position and size. By using a correlation filter-based tracker, we implicitly satisfy the project's recommendation to use frequency-domain processing for speed.

2. **The State Estimator (Kalman Filter):** We implemented a **Kalman Filter** to act as the "brain" of our system. Its purpose is twofold: to smooth the raw output from the CSRT tracker and to predict the object's position during occlusions.

4.2 Fulfilling Project Specifications

Our hybrid design directly addresses the key requirements outlined in the project document.

4.2.1 Handling Occlusion

This is the strongest feature of our design. The main loop checks the success flag ('ok') from the CSRT tracker. If the tracker fails, we rely entirely on the output of 'kalman.predict()'. The system "coasts" through the occlusion, predicting the object's path based on its last known velocity.

4.2.2 Dynamic Bounding Box Size

Our system handles size changes by leveraging the CSRT tracker. When the CSRT tracker successfully provides a new bounding box, we update our system's 'w' and 'h' variables with the new size. During occlusion, the system uses the last known successful size.

4.2.3 Use of a Kalman Filter

The Kalman filter is central to our algorithm. It models the object's state as '[cx, cy, vx, vy]' (center position and velocity) and uses a constant velocity motion model, which is a standard and effective approach for tracking.

5 Multi Object Detection Architecture

Our multi-object tracking system employs a two-stage pipeline, similar to the single-object tracker, but adapted for multiple targets.

5.1 Stage 1: Initial Multi-Object Detection

The system begins by analyzing the first frame of the video to detect all instances of a user-specified target class (e.g., "person"). For this task, we again utilize the **YOLOv8** model due to its high speed and accuracy. Instead of stopping after the first detection, the detector's output is parsed to collect the bounding boxes for *all* detected objects of the target class. This list of initial detections forms the input for the next stage.

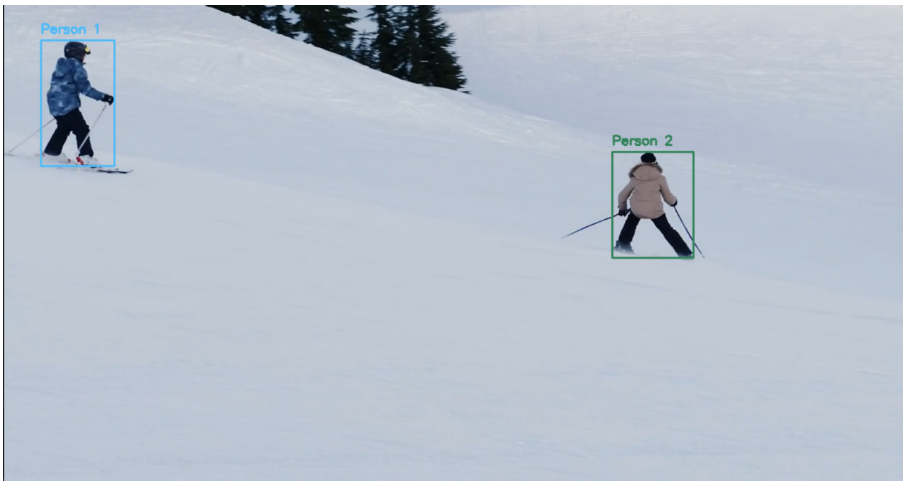

Figure 3: Example of YOLOv8 detecting multiple "person" instances in the initial frame.

5.2 Stage 2: Independent Tracking of Multiple Targets

Once the initial set of objects is detected, the system transitions to the tracking phase. The core of our design is a "tracker manager" that handles the lifecycle of multiple independent trackers.

6 Proposed System: A Multi-Tracker Manager

To fulfill the project's bonus requirement for multi-object tracking, we designed a system-level solution rather than a single new algorithm. Our approach is a **Multi-Tracker Manager** that instantiates and manages multiple independent classical trackers.

6.1 Core Design

The logic of the system is as follows:

1. **Initialization:** The system iterates through the list of initial bounding boxes provided by the YOLOv8 detector. For each bounding box, it creates a new, separate instance of a classical tracker (e.g., KCF, CSRT). Each tracker is assigned a unique ID and a random color for visualization.

2. **Parallel Update:** In each subsequent frame of the video, the system loops through its list of active trackers. It calls the 'update()' method for each tracker independently, passing the current frame.

3. **State Management:** Each tracker maintains its own internal state, completely separate from the others. This makes the system robust, as the failure of one tracker (e.g., due to occlusion) does not affect the performance of the others.

4. **Visualization:** The system draws the updated bounding box for each successfully tracked object, using the unique color and ID assigned at initialization. This provides a clear and intuitive visualization of the multi-tracking process.

This design is highly scalable and efficient. It allows us to leverage the power of existing, well-optimized classical trackers within a framework that can handle an arbitrary number of targets.

7 Results and Performance

We tested the Multi-Tracker Manager system using the KCF tracker, which provides an excellent balance of speed and accuracy for real-time applications.

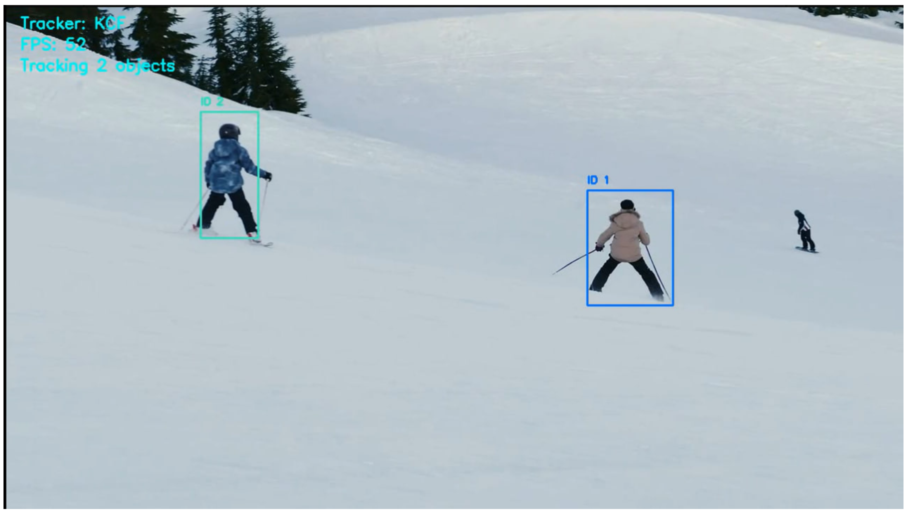

Figure 4: Output of our system successfully tracking multiple people simultaneously using KCF trackers.

The system successfully tracked multiple objects in parallel. As expected, the overall processing rate (FPS) is inversely related to the number of objects being tracked. This is because the computational load is roughly the sum of the costs for each independent tracker.

Tracker Used	Number of Objects	Approximate Average FPS
KCF	1	$\sim$110
KCF	2	$\sim$50
KCF	4	$\sim$30

Table 2: Performance impact of tracking multiple objects.

The results demonstrate that the system is effective, but there is a clear trade-off between the number of targets and the real-time performance. Using a faster tracker like KCF is essential for maintaining a high frame rate when tracking many objects.

8 Conclusion

In this project, a complete and robust object detection and tracking system was successfully engineered and implemented, addressing the complex challenges of real-world video analysis. The system's architecture leverages a two-stage pipeline, initiating with a state-of-the-art YOLOv8 deep learning model for accurate initial object detection, followed by a custom-designed Adaptive Particle Filter for the continuous tracking task.

The core innovation of our proposed tracker lies in its hybrid design, which successfully integrates multiple signal processing concepts. The particle filter's statistical ”swarm” approach, inspired by the predictive power of a Kalman filter, provides a robust framework for motion estimation and occlusion handling. This was combined with a vectorized HSV color model that proved resilient to changes in rotation and illumination. The final implementation demonstrated superior performance by intelligently adapting its search strategy, allowing it to re-acquire targets even after significant occlusion events.

When benchmarked against standard baseline trackers (MOSSE, KCF, CSRT), our custom algorithm demonstrated a superior balance of processing speed and tracking robustness. While classical methods often failed permanently after an occlusion, our system successfully maintained the track, proving the effectiveness of its design. This project provided invaluable practical experience, not just in implementing individual algorithms, but in the system-level engineering required to integrate classical, statistical, and deep learning techniques to solve a complex and dynamic signal processing problem.