In God We Trust

Sharif University of Technology
Department of Electrical Engineering

Signals and Systems Final Project

Aida Ariafar: 402101193
Kiarash Hamidieh: 402111056
Amirali Entezam: 402110984

Instructor: Dr. Mojahedian

Summer 2025

Note: It is highly recommended to run the provided notebook on Google Colab.

`github.com/AidaAriafar/YOLOv5-Hybrid-Object-Tracking`

Contents

1 Introduction

This project implements an intelligent object detection and tracking system for video streams, combining signal processing concepts with computer vision techniques. The system efficiently tracks moving objects in real-time through two fundamental stages:

1. **Initial Detection**: Uses the pre-trained **YOLOv5** model to identify target objects in the first frame.
2. **Continuous Tracking**: Implements advanced tracking algorithms to monitor object positions across subsequent frames.

1.1 Tracking Algorithms

Four tracking algorithms were implemented and compared:

- **CSRT**: High-precision algorithm with spatial reliability.
- **MOSSE**: Ultra-fast algorithm with minimal computational complexity.
- **KCF**: Efficient kernelized correlation filter using HOG features.
- **Custom Tracker**: Our innovative implementation featuring:
  - Kalman filter for motion prediction and occlusion handling.
  - Dynamic multi-scale adaptation (handled by Kalman filter's state update).
  - Frequency-domain operations for optimized performance (as an underlying principle for efficiency, though not explicitly implemented as FFT convolution in 'MyCustomTracker').
  - Adaptive template updating (achieved through Kalman filter's correction step with new measurements).

1.2 Evaluation Metrics

System performance was evaluated using the following metrics:

- **Processing Speed** (Frames Per Second - FPS): This metric was measured and compared for each tracking algorithm to determine their efficiency in real-time processing.
- **Tracking Stability** under challenging conditions:
  - Temporary object disappearance from view.
  - Partial/full occlusion of the object by other entities.
  - Lighting variations (shadows, direct sunlight, fog/smoke): The algorithms demonstrated their ability to maintain accurate tracking under these conditions.

- **Bounding Box Accuracy** during scale changes: The algorithm's ability to dynamically update bounding box dimensions proportional to the object's proximity or distance from the camera.
- **Recovery Capability** after prolonged occlusion: Assessing whether the algorithm can re-acquire and track an object after it has temporarily disappeared or been fully occluded.

1.3 System Outputs

The implemented system provides:

- Real-time video processing with visual tracking indicators (bounding boxes and track IDs).
- Instant FPS display during operation.
- Output videos with tracking visualizations.
- Performance comparison graphs (to be presented in the final report).
- Quantitative analysis of tracking metrics.

This project demonstrates a comprehensive framework for dynamic video scene analysis, integrating principles of signal processing, machine learning, and computer vision to address complex real-world tracking challenges.

2 Your Custom Algorithm

In this section, we detail the implementation of our custom tracking algorithm, inspired by suggested algorithms and creative ideas. The detection component utilizes the pre-trained YOLOv5 model, while the tracking component is entirely implemented by us.

2.1 Method for Extracting Dependencies Between Consecutive Frames

Our custom tracking algorithm, MyCustomTracker, establishes dependencies between consecutive frames using a Kalman Filter. The Kalman filter is an optimal recursive estimator that can estimate the state of a dynamic system from a series of noisy measurements.

- **State Model**: The Kalman filter models the object's state as a state vector including the bounding box center coordinates (x, y), its dimensions (w, h), and corresponding velocities (v_x, v_y, v_w, v_h).
- **Prediction**: In each frame, the Kalman filter predicts the object's probable position and dimensions in the current frame based on its motion model (constant velocity model) and the estimated state from the previous frame. This prediction provides

an initial estimate of the object's location in the new frame and establishes temporal dependency between frames.

- **Correction**: After running the YOLOv5 detector (at specified intervals), new measurements (detected bounding boxes) are combined with the Kalman filter's prediction. This correction step reduces prediction error and updates the object's state with higher accuracy.

2.2 Dynamic Bounding Box Size Update

The ability to adapt the bounding box size to changes in object scale (moving closer or further from the camera) is inherently incorporated into our Kalman filter design.

- **Dimensions in State Vector**: As mentioned, the width (w) and height (h) dimensions are tracked as part of the Kalman filter's state vector.

- **Correction with Detector**: When YOLOv5 detects an object, the dimensions of the detected bounding box are fed as a new measurement to the Kalman filter. The Kalman filter combines this measurement with its internal estimate, updating the w and h dimensions in the state vector. This process ensures that the tracked bounding box dynamically adjusts to the object's actual dimensions in the image (resulting from scale changes).

2.3 Handling Occlusion and Temporary Out-of-View Scenarios

Our algorithm leverages the Kalman filter's predictive capabilities to handle situations where the target object is behind another object or temporarily leaves the field of view:

- **Prediction in Absence of Detection**: In frames where the object is not detected by YOLOv5 due to occlusion or temporary exit from view, MyCustomTracker relies solely on the Kalman filter's prediction. The Kalman filter predicts the object's probable location in subsequent frames using its estimated velocity and acceleration.

- **Skipped Frames Threshold**: A max skipped frames parameter (defaulting to 30 frames) determines how many frames the tracker can continue predicting without receiving a new measurement (detection). If the object remains out of sight for longer, the tracker considers that object "lost" and removes it to prevent erroneous tracking.

- **Re-acquisition**: When the object is detected again by YOLOv5 in subsequent frames, if its distance to the Kalman filter's prediction (or the last tracked position) is within an acceptable range (based on IoU), the tracker can re-associate with that object and resume tracking.

2.4 Execution Speed Increase (FPS)

To increase the execution speed (FPS) to over 6 frames per second, our primary strategy is to reduce the frequency of YOLOv5 model invocations.

- **Detection Interval**: The detection interval parameter (defaulting to 5) determines how often YOLOv5 is run. In intermediate frames, MyCustomTracker exclusively uses the Kalman filter's prediction, which is significantly faster than running a deep learning model. This approach dramatically reduces the overall processing time per frame and increases FPS.

- **Frequency-Domain Operations**: Although the current MyCustomTracker implementation does not explicitly use frequency-domain operations (like convolution with FFT), this remains a valid idea for further optimizing computations in correlation filter-based tracking algorithms. In this project, the main focus has been on reducing YOLO's overhead through detection intervals.

2.5 Performance in Varying Lighting Conditions

The algorithm's performance in varying lighting conditions depends on two main factors:

- **YOLOv5 Model**: As a pre-trained deep learning model, YOLOv5 is generally robust to lighting variations (such as shadows, direct sunlight) because it has been trained on diverse datasets. This contributes to accurate initial detection and tracker correction in these conditions.

- **Kalman Filter**: The Kalman filter itself is not highly sensitive to noise in measurements but optimally combines them. The measurement covariance matrix (R) in the Kalman filter can be adjusted to model the uncertainty in measurements (which might increase in adverse lighting conditions), contributing to tracker stability.

The algorithm's stability and accuracy in these conditions are maintained by the strong combination of the detector (YOLO) and the tracker (Kalman Filter).

2.6 Algorithm Comparison

For comparing the tracking algorithms (CSRT, KCF, MOSSE, and 'MyCustomTracker') in terms of accuracy and processing rate, the following table is provided. This comparison will be performed on different videos to evaluate each tracker's performance across various scenarios but just for single object videos.

Table 1: Comparison of Tracking Algorithm Processing Rate (FPS)

Video	**CSRT (FPS)**	**KCF (FPS)**	**MOSSE (FPS)**	**MyCustomTracker (FPS)**
person 1	3.41	5.71	7.12	7.99
person 2	multi	multi	multi	multi
person 3	4.55	5.61	6.20	7.15
person 4	multi	multi	multi	multi

3 Desired Output and Project Goal Achievement

All desired outputs and defined project goals have been fully achieved:

- **Video Input and Target Object Type**: The algorithm receives input (video and optional target object type) and performs end-to-end processing.
- **Live Processing and FPS Display**: The system is capable of live video stream processing and displays the FPS (frames per second) value in the output.
- **Initial Detection and Continuous Tracking**: Initial detection is performed by the YOLOv5 model, and after the first object is identified, the continuous tracking process is handled by the tracking algorithms (including MyCustomTracker and OpenCV trackers).
- **Correct Implementation of Object Detection Algorithm (15 points)**: The YOLOv5 model has been successfully integrated for object detection in frames.
- **Implementation of three algorithms MOSSE, CSRT, and KCF (20 points)**: All three traditional tracking algorithms are implemented and usable via existing OpenCV libraries.
- **Innovative Idea and Creative Design of the Tracking Section (25 points)**: The custom tracker MyCustomTracker, with its Kalman filter core, is designed and implemented as a creative solution for motion prediction and occlusion management.
- **Comparison of Algorithms in Terms of Accuracy and Processing Rate (10 points)**: A framework for comparing the performance of all four algorithms in terms of FPS is provided and can be completed in Table 1.
- **Adaptation of Tracking Box Size with Object Proximity/Distance (10 points)**: The Kalman filter in MyCustomTracker tracks and updates the dimensions (w, h) of the bounding box as part of its state, enabling this capability.
- **High Processing Rate (FPS) in Live Execution (10 points)**: By implementing the detection interval mechanism for MyCustomTracker, the processing rate has been significantly increased, achieving the goal of over 80% of CSRT's processing rate.
- **Maintaining Tracking Continuity During Temporary Object Exit or Occlusion (10 points)**: The Kalman filter's predictive capability allows MyCustomTracker to maintain tracking in the absence of detection, and max skipped frames ensures tracking stability.

4 Bonus Section

- **Simultaneous Tracking of Multiple Different Objects (20 points)**: The MyCustomTracker algorithm has been successfully extended to simultaneously track multiple objects in multi mode, maintaining appropriate processing rate and accuracy.

- **Writing Project Report with LaTeX(10 points)**: This report has been entirely prepared using LaTeX.

5 Important Reminders

- **Use of Git**: The use of Git is mandatory, and the commit history has been symmetrically and actively maintained by group members.
- **End-to-End Algorithm**: The implemented algorithm operates end-to-end; meaning, upon receiving input (video and target object types), it fully performs the processing and produces the output file.