



Introductory seminar on Computer Vision

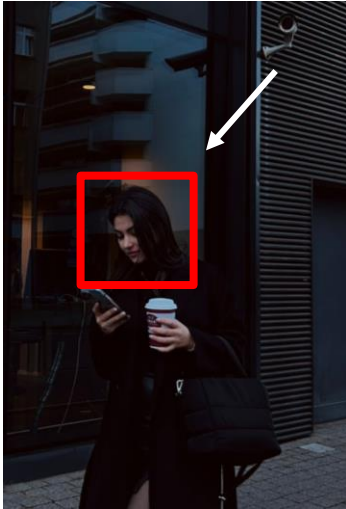
Sara Concas, Cagliari Digital Lab 2024 - Day 5



sara.concas90c@unica.it

Object Tracking

Process used to monitor and follow the movement of an object across a sequence of frames in a video or a stream of images. The goal is to identify the position and other relevant attributes (like speed, size, and trajectory) of the object as it moves through the scene.

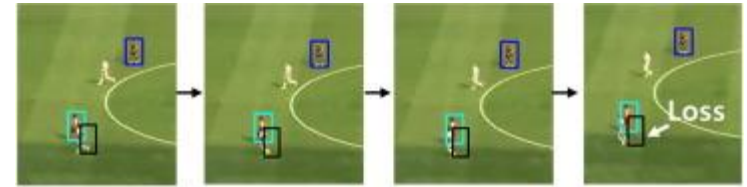


Surveillance



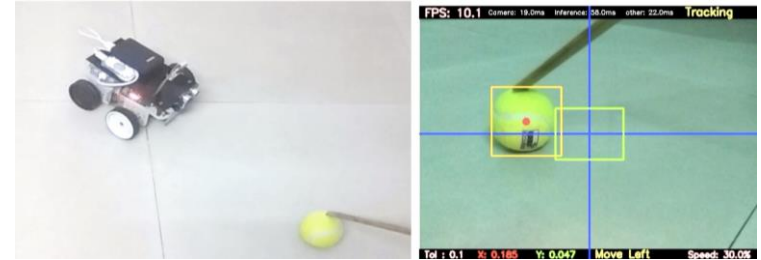
Autonomous Vehicles

<https://www.mdpi.com/1424-8220/18/7/2004>



Sports Analysis

<https://www.sciencedirect.com/science/article/pii/S1047320319303049>



Robotics

<https://helloworld.co.in/article/ai-robot-object-tracking-object-following-robot-using-tensorflow-lite>

Object Tracking vs Object Detection



Object Detection

Goal: identify and locate objects of interest within a single image or a single frame of a video.

Process: scan an image or frame and predict the presence of objects by analyzing features such as color, texture, and shapes.

Object Tracking vs Object Detection



Object Tracking

Goal: monitor the movement of one or more objects across a sequence of frames in a video. It aims to maintain the identity of the object(s) as they move, dealing with their changing position, size, and appearance over time.

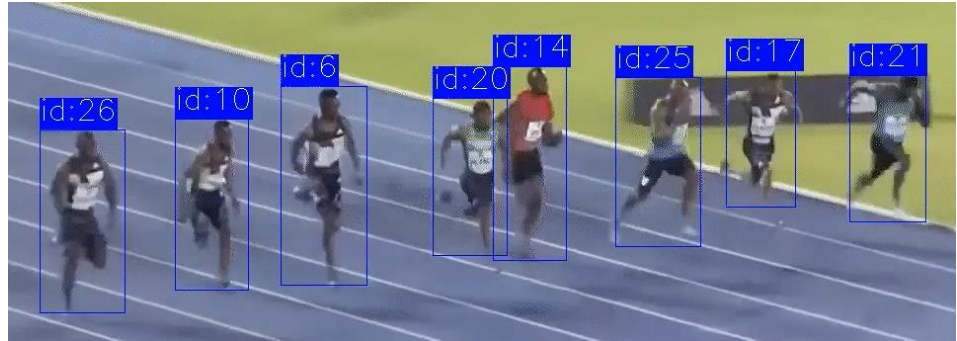
Process: starts with detecting an object (often using an object detection algorithm) and then follows that object through subsequent frames, updating its location and appearance. Tracking algorithms predict the object's position in the next frame and adjust based on new information.

Types of Object Tracking

- **Single Object Tracking:** Involves tracking one object at a time, focusing on accurate and precise tracking of that specific object.
- **Multiple Object Tracking (MOT):** Involves tracking multiple objects simultaneously, which is more complex due to challenges like object interactions, occlusions, and overlapping objects.



<https://pyimagesearch.com/2018/07/30/opencv-object-tracking/>



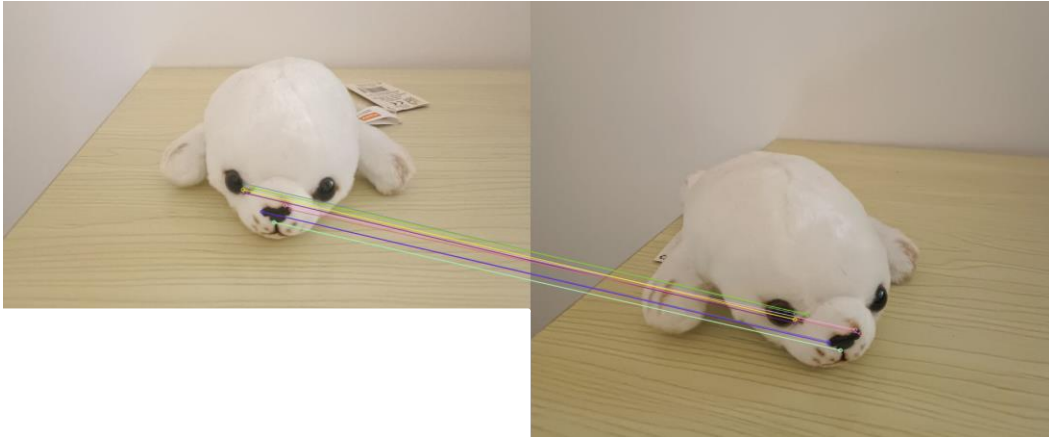
<https://www.datature.io/blog/introduction-to-bytetrack-multi-object-tracking-by-associating-every-detection-box>

Image Tracking VS Video Tracking

Image Tracking

Input: static images, where the objective is to detect and possibly track an object within a single image or between two related images.

Objective: identify and locate specific features, objects, or patterns within one or more still images. It often involves matching a known template or pattern within the image to find its position.



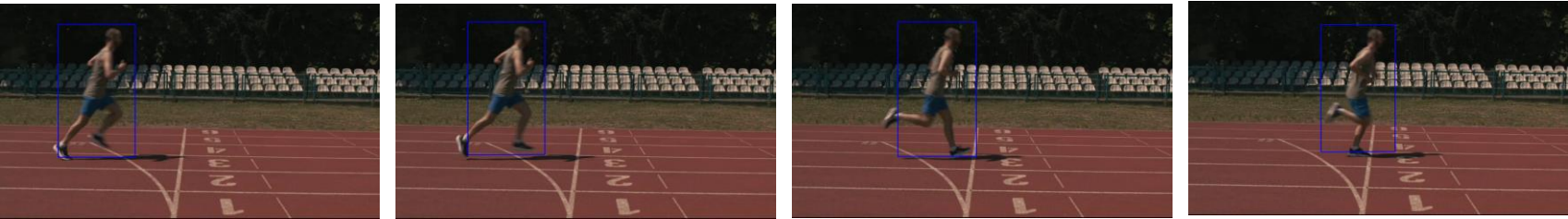
Keypoints are detected in both images and then matched between the two.

Image Tracking VS Video Tracking

Video Tracking

Input: A continuous stream of video frames, typically from a camera feed or pre-recorded video. This input is dynamic and can involve multiple objects moving across the video frames.

Objective: detect and track one or more objects or features over time across multiple frames. The goal is to follow the motion and location of objects within the video, often updating the object's position in real-time.



The object is detected in the first frame and then it is tracked throughout the video.

Image Tracking - Example



We will use feature matching to track an object (or specific features) from one image to another. This approach is useful when the object has moved, rotated, or scaled slightly between the two images.

<https://bit.ly/3Tewacp>

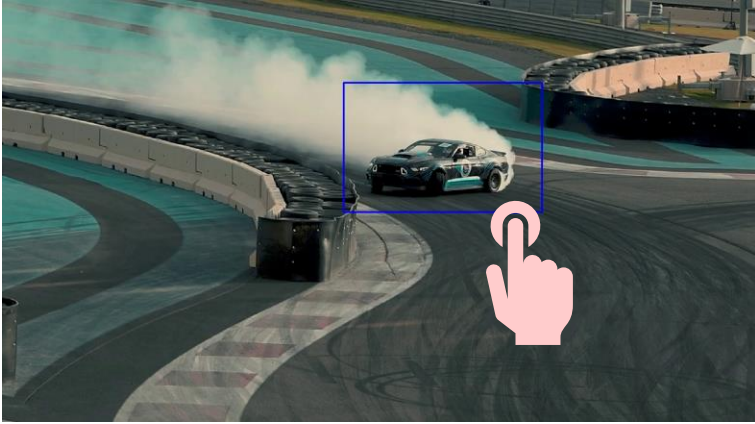
Object Trackers

An **object tracker** follows the movement of an object through consecutive frames in a video. It identifies the object's position and appearance and updates its understanding of these characteristics over time.

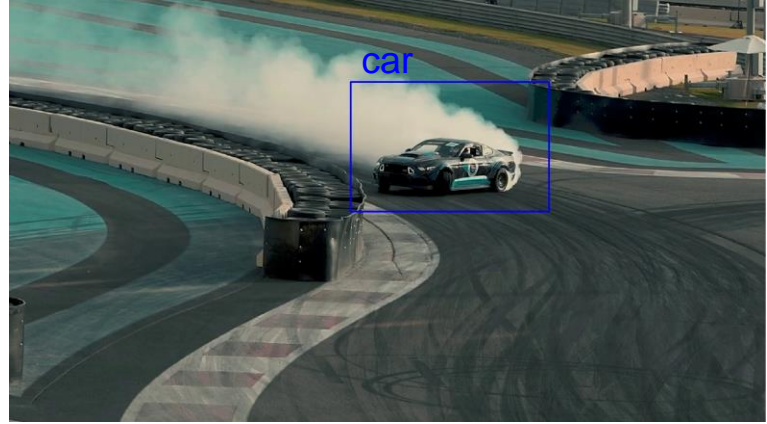


Object Trackers - Initialization

The first step is to detect and locate the object to be tracked in the initial frame of the video. This can be done manually (through user input) or automatically (using object detection algorithms).



Manually



Automatically

Object Trackers - Tracking




The tracker estimates the object's position in each subsequent frame by analyzing its movement and appearance.

How?

- Using image correlation to track an object based on its visual appearance in a small region around the detected object.
- Estimating motion between two consecutive frames by tracking pixel-level movement.
- Predicting the object's position in the next frame based on its motion (velocity, acceleration) and updating with new information.
- Tracking the object's color or intensity distribution by adjusting the region's size and location to fit the object's changing appearance.

Object Trackers - Model Updating



As the object moves, its appearance or position may change due to rotation, scaling, or environmental factors (like lighting).

Static Trackers: rely on fixed characteristics or pre-defined models to follow an object. They do not change or update their internal model based on the object's appearance or environmental conditions during the tracking process.

Simple models but may struggle when there are significant changes in the object's appearance, lighting conditions, scale, or orientation.

Adaptive Trackers: dynamically update their internal model of the object or its features as the object moves through the scene. They adjust based on changes in appearance, lighting, or environmental conditions.

Better equipped to handle complex scenes, occlusions, and variations in lighting or orientation but the models are more complex

Object Trackers - Model Updating

	Static Trackers	Adaptive Trackers
Model Update	No updates; uses a fixed model	Continuously updates based on new data
Handling of Variability	Struggles with changes in appearance or conditions	Adapts to changes in appearance, lighting, etc.
Complexity	Simpler, more computationally efficient	More complex, often computationally heavier
Suitability	Best for stable, controlled environments	Best for dynamic, real-world environments
Robustness	Less robust to occlusions or scale changes	More robust to occlusions, appearance changes

Challenges in Object Tracking



- Occlusion: The object might be hidden or blocked.
- Illumination changes: Sudden changes in lighting conditions can alter the object's appearance.
- Scale and rotation: The object's size and orientation can change as it moves.
- Background clutter: The object may blend in with complex backgrounds.

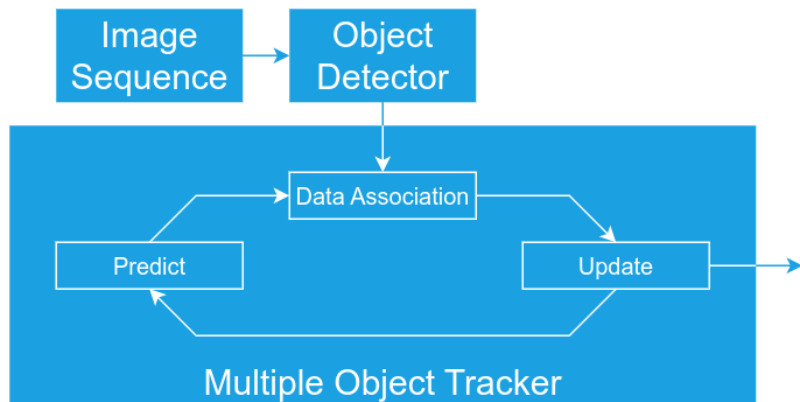
SORT (Simple Online and Realtime Tracking)

Lightweight and efficient multi-object tracking algorithm that operates in real-time.
It associates objects detected in video frames with their corresponding tracked objects over time.

1. Object Detection

An external object detection algorithm (such as YOLO), provides bounding boxes for the objects of interest in each frame.

The object detector outputs bounding boxes at each frame, which serve as the observations that SORT will use to track objects across frames.



SORT (Simple Online and Realtime Tracking)



2. Kalman Filter for Object State Estimation

The state of each tracked object is estimated. The state consists in:

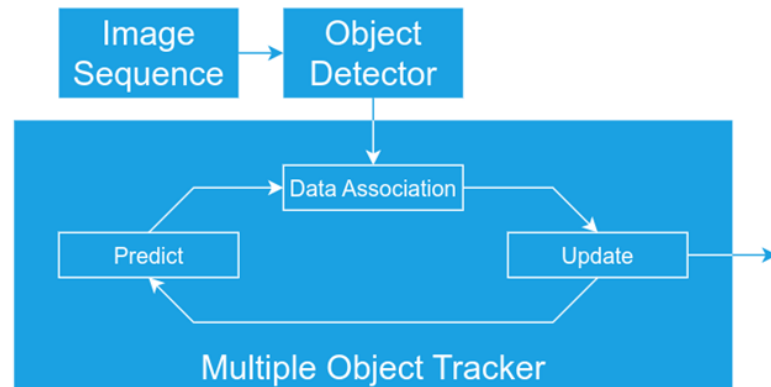
- The bounding box coordinates: (x, y, width, height).
- The velocity of the object (i.e., how fast and in what

direction it is moving).

The **Kalman filter**: recursive algorithm that combines noisy measurements (detections) with predictions to estimate an object's state.

It operates in two phases:

- Prediction: Based on the object's previous state (e.g., position, velocity), the Kalman filter predicts the next location of the object in the current frame.
- Update: Once the new detection arrives, the Kalman filter updates the predicted state with the observed detection.



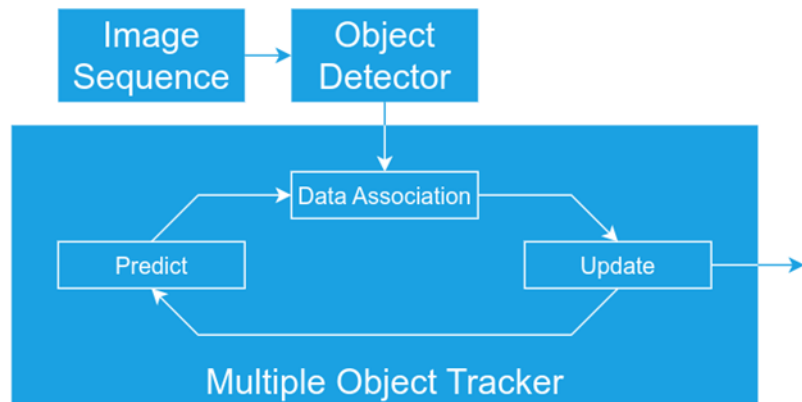
© 2023 Luffca Inc.

<https://www.luffca.com/2023/04/multiple-object-tracking-sort/>

SORT (Simple Online and Realtime Tracking)

3. Data Association with the Hungarian Algorithm

match the detected bounding boxes to the predicted locations of previously tracked objects. This is known as the data association problem. the Hungarian algorithm, which finds the optimal assignment of detected bounding boxes to existing tracks by minimizing a cost function. The cost function is computed between the predicted bounding boxes (from the Kalman filter) and the newly detected bounding boxes.



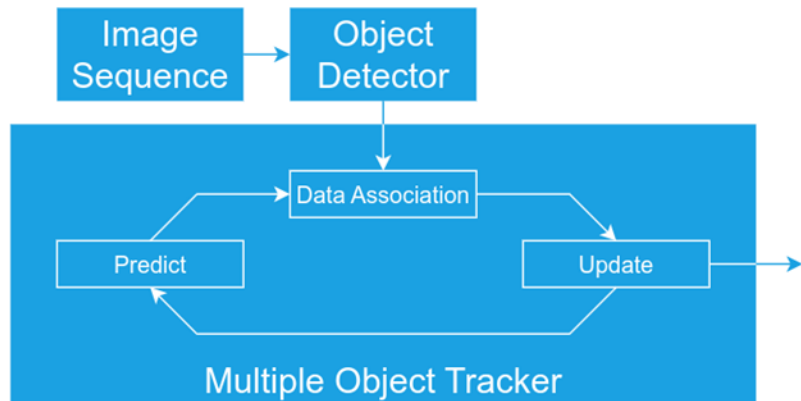
© 2023 Luffca Inc.

<https://www.luffca.com/2023/04/multiple-object-tracking-sort/>

SORT (Simple Online and Realtime Tracking)

4. Track Management

- If a detected bounding box cannot be matched to any existing tracks, SORT initializes a new track for the object.
- If an existing track fails to get matched with a detection for a certain number of consecutive frames, the track is deleted.
- Tracks that are successfully matched with new detections are updated, and their Kalman filter is refined.



© 2023 Luffca Inc.

<https://www.luffca.com/2023/04/multiple-object-tracking-sort/>

SORT (Simple Online and Realtime Tracking)



Lightweight and efficient multi-object tracking algorithm that operates in real-time.
It associates objects detected in video frames with their corresponding tracked objects over time.

Limitations

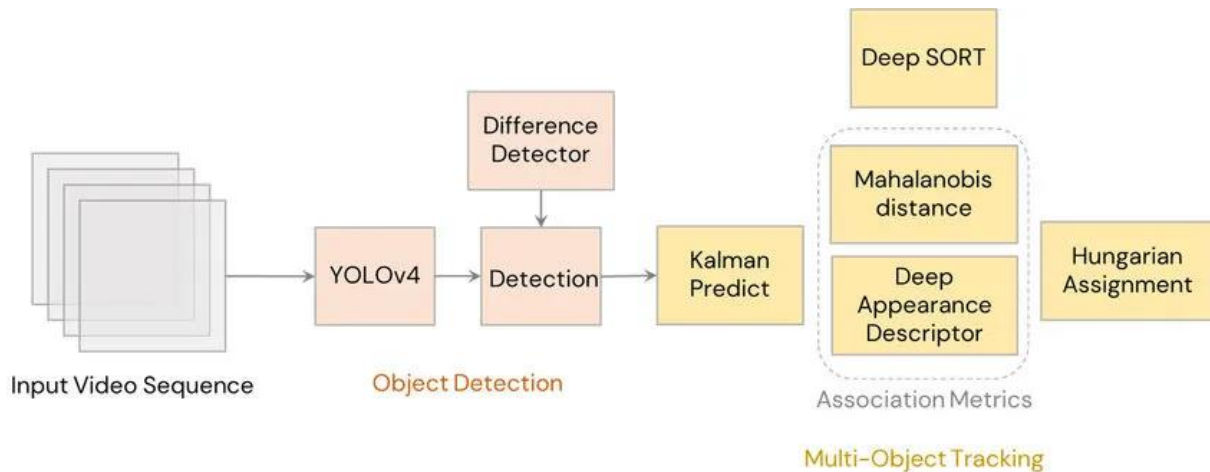
- It relies solely on the spatial information of bounding boxes (positions and sizes) for tracking. It does not use any appearance information (e.g., color, texture) to distinguish between objects.
- SORT can struggle when objects move close together, overlap, or are temporarily occluded, as it may lose track of which object is which

DeepSORT (Simple Online and Realtime Tracking with Deep Association Metric)



Extension of the SORT algorithm, incorporating deep learning to enhance object tracking by improving the association between detected objects across frames.

To address SORT's limitations, DeepSORT adds a deep learning-based appearance feature extractor to improve the robustness of object tracking in scenarios with occlusions, re-entrances, and complex environments.



Channel and Spatial Reliability Tracking (CSRT) algorithm



State-of-the-art object tracking method that builds upon Discriminative Correlation Filters (DCF), designed to improve the robustness and accuracy of object tracking in challenging scenarios.

It focuses on using both channel and spatial information to more effectively track objects over time, especially in dynamic environments where objects may undergo deformations, occlusion, or background clutter.

Channel Reliability

It works with multiple feature channels (color, texture and gradients) to describe the object being tracked

It assesses the reliability of each channel and assigns different weights based on how useful each channel is for tracking the object.

It maintains better accuracy in environments where certain features (like color or texture) may change or be affected by background noise.

Channel and Spatial Reliability Tracking (CSRT) algorithm



State-of-the-art object tracking method that builds upon Discriminative Correlation Filters (DCF), designed to improve the robustness and accuracy of object tracking in challenging scenarios.

It focuses on using both channel and spatial information to more effectively track objects over time, especially in dynamic environments where objects may undergo deformations, occlusion, or background clutter.

Spatial Reliability

It computes the spatial reliability of different parts of the object in the frame.

This allows the tracker to focus more on reliable parts of the object's spatial structure, such as stable regions, and to downplay parts that may be occluded or change shape.

By assessing spatial reliability, CSRT is able to adapt to changes in the object's appearance, size, and position more effectively.

Single Video Object Tracking – Example 1 – ROI SELECTION



We will load a video and visualize its first frame. In this frame we will manually select the region of interest (ROI) depicting the object we want to track. This will generate a bounding box that will be used to initialize the tracker and follow the object throughout the video.

[GitHub Repo](#)

Script name: video_tracking_single_ROI_selection.py

Single Video Object Tracking– Example 2 – OBJECT DETECTION



Now we will see how to automatically obtain the bounding box of the object we are interested in without having to manually select the region of interest.

The program will show the first frame, perform object detection by itself, and then it will ask us which object, among the detected ones, we want to track throughout the video.

[GitHub Repo](#)

<https://bit.ly/3Xi49IF>

Script name: video_tracking_single_singolo.py

Multiple Video Object Tracking – Example 1 – ROI SELECTION



We will load a video and visualize its first frame. In this frame we will manually select multiple regions of interest (ROI) depicting the objects we want to track. This will generate a bounding box that will be used to initialize the tracker and follow the objects throughout the video.

[GitHub Repo](#)

Script name: video_tracking_multiple_ROI_selection.py

Multiple Video Object Tracking– Example 2 – OBJECT DETECTION



Now we will see how to automatically obtain the bounding box of the objects we are interested in without having to manually select the regions of interest.

The program will show the first frame, perform object detection by itself, and then it will ask us which objects, among the detected ones, we want to track throughout the video.

[GitHub Repo](#)

Script name: video_tracking_multiple.py

<https://bit.ly/3XvVmhf>