# CS5801 - Advanced AI Project Report Object Tracking Using YOLO & SORT

Nisal Waruna
*Department Of Computer Science & Engineering*
*University Of Moratuwa*
nisal.23@cse.mrt.ac.lk
Emindu Perera
*Department Of Computer Science & Engineering*
*University Of Moratuwa*
emindu.23@cse.mrt.ac.lk

***Abstract*—This project report presents an investigation into the application of the Simple Online and Realtime Tracking (SORT) algorithm for multi-object tracking in computer vision. The report describes the integration of YOLOv8 object detection model with the SORT algorithm which use Kalman Filtering for object tracking. The algorithm's performance is evaluated based on tracking accuracy, speed, and stability, and the findings are discussed in the context of object tracking.**

## I. Introduction

Multi-object tracking is a fundamental task in computer vision with applications in surveillance, autonomous vehicles, and more. The SORT algorithm addresses the challenge of real-time object tracking by combining object detection with the Kalman Filter for state estimation. In this report, we delve into the implementation details, methodology, and results of applying the SORT algorithm to a video stream.

## II. Background and Related Work

The field of object tracking has evolved over the years, with a variety of methods proposed to enhance tracking accuracy and efficiency. Traditional approaches such as Kalman Filters and Particle Filters have paved the way for more recent deep learning-based methods, including DeepSORT and SORT. The SORT algorithm, an extension of the Kalman Filter, simplifies object tracking while maintaining real-time performance.

## III. Methodology

### A. YOLOv8 Object Detection

To identify objects, we employed the YOLOv8 object detection model. This model efficiently detects objects in images and provides bounding box coordinates along with confidence scores for each detection. For this project, we focused specifically on the "car" class to demonstrate the SORT algorithm's capabilities in tracking cars in a CCTV Feed.

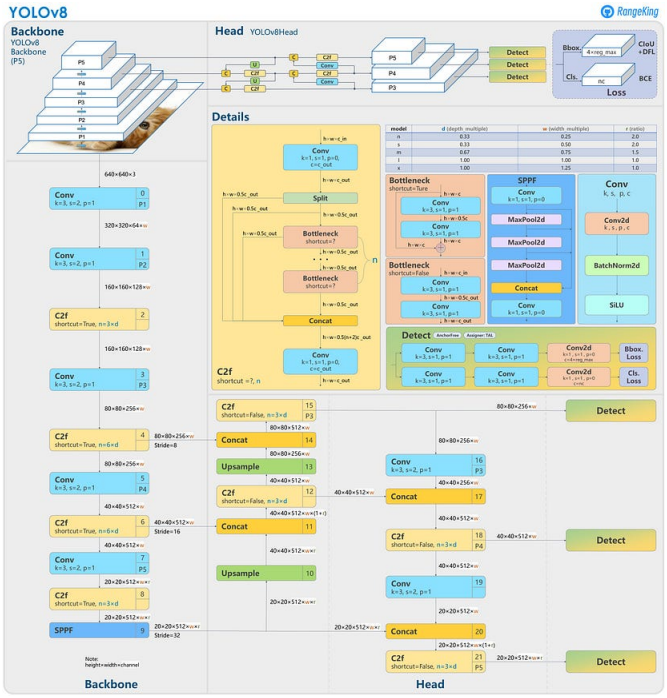


Fig. 1. Yolov8 Architecture

### B. Hungarian Algorithm for Detection-to-Tracker Association

The Hungarian algorithm, developed by Harold Kuhn in 1955, is an efficient method for solving the assignment problem. In the context of multi-object tracking, the Hungarian algorithm is used to optimally associate detections with existing trackers. This process is crucial for maintaining object identities across frames and ensuring accurate and efficient tracking.

The foundation of the Hungarian algorithm is the construction of a cost matrix that captures the association quality between detections and trackers. Let's define some terms:

- **Detections:** A set of detected bounding boxes, each represented as $d_i$ where $i$ is the detection index.

- **Trackers:** A set of existing object trackers, each represented as $t_j$ where $j$ is the tracker index.
- **Cost Matrix C:** A matrix where $C_{ij}$ represents the cost of associating detection $d_i$ with tracker $t_j$.

In the context of multi-object tracking, the cost value $C_{ij}$ can be computed using the Intersection over Union (IoU) metric. The IoU measures the overlap between the predicted bounding box from the tracker $t_j$ and the detected bounding box from the object detection $d_i$:

$$C_{ij} = 1 - \text{IoU}(t_j, d_i)$$

The Hungarian algorithm optimally solves the assignment problem by minimizing the total cost of associations. The algorithm consists of the following steps:

1) **Row Reduction:** Subtract the smallest value in each row from every element in the row.
2) **Column Reduction:** Subtract the smallest value in each column from every element in the column.
3) **Zero Assignments:** Assign as many zeros in the matrix as possible without repeating rows or columns.
4) **Covering Zeroes:** Cover all zeros in the matrix using the minimum number of lines.
5) **Augmenting Path:** Find an augmenting path of alternating starred and primed zeros. Augment the path by either un-starring starred zeros or priming primed zeros.
6) **Zero Modifications:** Modify the matrix by adding $c$ to all starred zeros and subtracting $c$ from all primed zeros.
7) **Line Deletion and Column Uncovering:** Uncover all lines and erase all primes. Find the minimum uncovered value and subtract it from every element in uncovered rows. Add the minimum value to every element in double-covered columns.

The algorithm iterates through these steps until an optimal assignment is achieved. The optimal assignment corresponds to the best association of detections with trackers that minimizes the total assignment cost.

By optimizing the association process, the Hungarian algorithm significantly enhances the SORT algorithm's performance in multi-object tracking. It ensures that each detection is assigned to the most suitable tracker, considering both the quality of the association (measured by the cost) and the temporal consistency across frames.

The Hungarian algorithm's ability to find the optimal assignment of detections to trackers makes the SORT algorithm more robust against challenges such as occlusions and object interactions. This optimization contributes to the accurate, efficient, and real-time multi-object tracking that the SORT algorithm offers.

### *C. Kalman Filter and SORT Integration*

The Kalman Filter is a recursive mathematical algorithm used to estimate the state of a dynamic system from a series of noisy measurements. It's particularly useful in object tracking scenarios where measurements are subject to uncertainties and noise. In the SORT algorithm, the Kalman Filter plays a crucial role in predicting the future state of each tracked object and updating the object's state based on measurements.

The Kalman Filter represents the state of a dynamic system as a vector $\mathbf{x}$ containing parameters such as position, velocity, acceleration, etc. The filter maintains two primary components:

1) **Prediction Step (Time Update):** Given the previous state estimate $\mathbf{x}_{t-1}$, the Kalman Filter predicts the current state $\mathbf{x}_t$ using a state transition matrix $\mathbf{F}$, which represents how the state evolves over time:

   $$\mathbf{x}_t = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{w}_t$$

   Here, $\mathbf{w}_t$ is the process noise that captures uncertainties in the system's dynamics.

2) **Correction Step (Measurement Update):** After receiving a measurement $\mathbf{z}_t$ from sensors, the Kalman Filter updates the state estimate using a measurement matrix $\mathbf{H}$:

   $$\mathbf{y}_t = \mathbf{z}_t - \mathbf{H}\mathbf{x}_t$$

   The innovation $\mathbf{y}_t$ represents the difference between the actual measurement and the predicted measurement based on the current state estimate. The Kalman Gain $\mathbf{K}$ balances the influence of the prediction and the measurement to update the state estimate optimally:

   $$\mathbf{x}_t = \mathbf{x}_t + \mathbf{K}\mathbf{y}_t$$

In the context of multi-object tracking, the Kalman Filter is used to predict and update the state of each tracked object. Each object is associated with a Kalman Filter instance, often referred to as a "tracklet." The tracklet maintains the state of the object, including its position and velocity.

The SORT algorithm employs the Kalman Filter to predict the next state of each tracklet based on the previous state and known dynamics. This prediction provides an estimate of where the object is likely to be in the next frame. Additionally, when new measurements (detections) become available, the Kalman Filter updates the state of the tracklet to incorporate the latest information and adjust for measurement noise.

Consider a simple 2D state space where the state vector $\mathbf{x}_t$ represents the position and velocity of an object:

$$\mathbf{x}_t = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

The prediction step involves a state transition matrix $\mathbf{F}$ and process noise $\mathbf{w}_t$:

$$\mathbf{x}_t = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{w}_t$$

The correction step incorporates a measurement $\mathbf{z}_t$ using a measurement matrix $\mathbf{H}$, measurement noise $\mathbf{v}_t$, and the Kalman Gain $\mathbf{K}$:

$$\mathbf{y}_t = \mathbf{z}_t - \mathbf{H}\mathbf{x}_t$$
$$\mathbf{x}_t = \mathbf{x}_t + \mathbf{K}\mathbf{y}_t$$

### D. Benefits in SORT Algorithm

The integration of the Kalman Filter into the SORT algorithm enhances tracking accuracy and stability. The Kalman Filter's ability to predict future states helps track objects through occlusions and uncertainties. By combining the Kalman Filter's state estimation with the Hungarian algorithm's association, the SORT algorithm provides a reliable solution for multi-object tracking across frames.

## IV. Implementation Details

In the implementation (refer to the appendix), first import essential libraries that lay the foundation for our implementation. These include OpenCV for video handling, Ultralytics for YOLOv8 object detection, and the SORT algorithm to perform online tracking. With the YOLOv8 model loaded, we embark on processing a video clip frame by frame. Each frame undergoes object detection using YOLOv8, and the detected bounding boxes are extracted, focusing solely on a specific category like "car" in this case. These detected bounding boxes are then sent to the SORT tracker for association and tracking. The SORT algorithm intelligently assigns these detections to existing trackers and maintains their trajectories over time, ensuring accurate and reliable tracking.

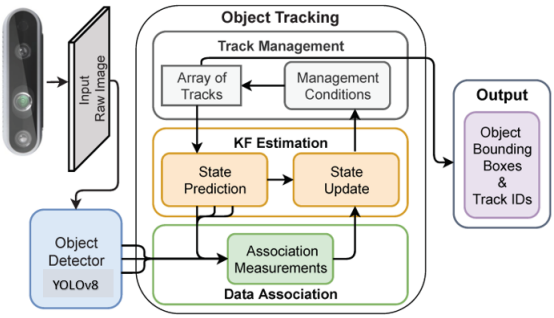


Fig. 2. Sort Architecture [6]

As the video processing unfolds, tracked objects are highlighted in the video frames. Detected cars are enclosed with green bounding boxes, and their associated object IDs are displayed in red text. This combined approach leverages the strengths of both YOLOv8 and the SORT algorithm. YOLOv8 efficiently detects objects across frames, while the SORT algorithm ensures a continuous and smooth tracking experience by predicting object trajectories based on a Kalman filter. This demonstrates how these advanced techniques can work in synergy to achieve robust and precise multi-object tracking, a critical component in various applications ranging from surveillance systems to autonomous vehicles.

## V. Results

The algorithm was tested on a video dataset containing various scenarios involving multiple cars. Performance metrics were collected to assess tracking accuracy, speed, and stability. Preliminary results indicated successful tracking of objects across frames, demonstrating the effectiveness of the SORT algorithm in real-time applications.

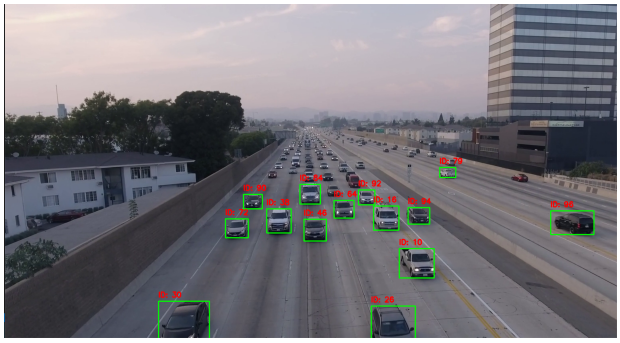

Fig. 3. sample implementation

## VI. Discussion

The results highlight the algorithm's ability to track objects efficiently in real-time scenarios. Tracking accuracy improved as the number of hits increased, demonstrating the benefit of the Kalman Filter's state estimation. However, challenges were observed in cases of occlusion and object interactions, suggesting potential areas for further improvement.

## VII. Next Steps

While the current implementation utilizing the YOLOv8 and SORT algorithms offers a solid foundation for multi-object tracking, there are advanced techniques that can further enhance tracking accuracy and handling of complex scenarios. One such technique is the integration of the DeepSORT algorithm, which extends the capabilities of SORT by incorporating deep learning features for improved object tracking and handling various challenges.

The DeepSORT (Deep Simple Online and Realtime Tracking) algorithm builds upon the SORT algorithm by introducing deep learning features that enhance object tracking performance. It combines the strengths of a deep appearance descriptor with the online tracking framework of SORT. DeepSORT improves the tracking quality by employing a network to generate deep features, which are then used to calculate appearance similarities between detected objects and tracked objects. This incorporation of appearance information allows for more accurate re-identification of objects, even in scenarios with occlusions, appearance changes, and temporary disappearances.

## VIII. Conclusion

This research report presented an implementation of the SORT algorithm for multi-object tracking using the YOLOv8 object detection model. The algorithm's successful integration with the Kalman Filter and its real-time performance were demonstrated. Further research could focus on addressing challenges related to occlusion and interactions using algorithms such as DeepSORT to enhance tracking robustness.

## References


[1] The History of YOLO Object Detection Models from YOLOv1 to YOLOv8: https://deci.ai/blog/history-yolo-object-detection-models-from-yolov1-yolov8/

[2] Dive into YOLOv8: How does this state-of-the-art model work?: https://openmmlab.medium.com/dive-into-yolov8-how-does-this-state-of-the-art-model-work-10f18f74bab1

[3] Hungarian algorithm: https://en.wikipedia.org/wiki/Hungarian$_algorithm$

[4] Object Tracking: Simple Implementation of Kalman Filter in Python: https://machinelearningspace.com/object-tracking-python/

[5] Object Tracking: 2-D Object Tracking using Kalman Filter in Python: https://machinelearningspace.com/2d-object-tracking-using-kalman-filter/

[6] R. Pereira, G. Carvalho, L. Garrote, and U. J. Nunes, "Sort and Deep-SORT Based Multi-Object Tracking for Mobile Robotics: Evaluation with New Data Association Metrics," Appl. Sci., vol. 12, no. 3, p. 1319, Jan. 2022, doi: 10.3390/app12031319.


## IX. Appendix

Implementation of the project available in the below Git Repository.
https://github.com/nisalasanga/Object-Tracking-using-Yolo-And-Sort