**Aim: Vehicle tracking simulation in Electric vehicles**

**Abstract:**

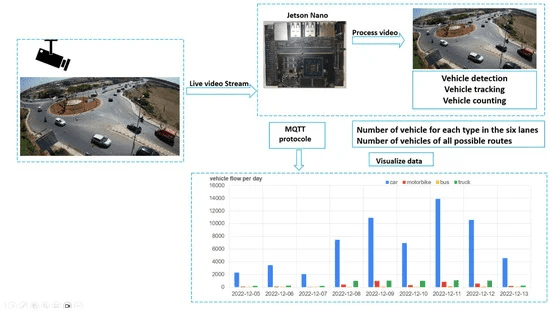
The application of object detection algorithms in video analytics has revolutionized various domains such as surveillance, transportation management, and safety systems. This project explores the utilization of video object detection, intra-frame, and inter-frame distance analysis, coupled with speed calculation, to gain deeper insights into object relationships, motion patterns, and behavioral dynamics. It also warns the electric vehicles, if any object comes into the proximity range. The study demonstrates the significance of integrating spatial and temporal analysis techniques in video object detection. Through experiments and case studies, we showcase the effectiveness of the proposed methodology in enhancing surveillance, transportation management, safety systems, and beyond. The project contributes to advancing the capabilities of video analytics systems, empowering stakeholders with actionable insights for informed decision-making and risk mitigation.

**Literature Survey:**

**Real-Time Vehicle Detection and Tracking on Nvidia Jetson Nano**

Traffic-management system aims to analyze the traffic at an intersection located between Bouskoura and Casablanca. For this purpose, we installed an IP camera connected to an Nvidia Nano Jetson card, which serves as our processing unit. Our system functions as a smart camera capable of detecting and tracking vehicles in real time. One of the most important features of our camera is that it follows the concept of edge computing, which means that the analysis is performed directly on the device, and only the counting results are transmitted. This has two main advantages: first, we do not need to transmit the images, which reduces bandwidth requirements; second, we respect the privacy of citizens, which is a crucial consideration for any application deployed in a public place.

The system architecture of the proposed traffic management system is composed of several components, including data acquisition, object detection and classification, object tracking and identification, counting, and data transmission. The data-acquisition component captures video footage of the traffic at each intersection using a smart camera and statistics are calculated and transmitted via the MQTT protocol. The object detection and classification component uses the YOLOv7-tiny algorithm for vehicle identification and classification. The object tracking and identification component uses the Deep SORT algorithm to track and identify vehicles. The counting component combines image processing techniques and deep-learning algorithms to count the number of vehicles passing through each lane. Finally, the data-transmission component is responsible for transmitting the processed traffic data to a centralized server for further analysis and decision-making via the MQTT protocol.

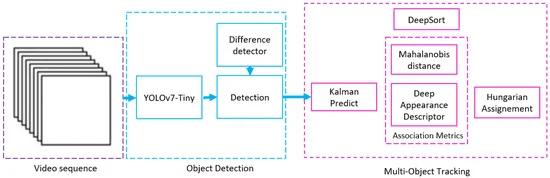


**Figure 1** shows the architectural overview of the proposed system, including its components.

***Vehicle tracking:***

Deep SORT addresses this issue by assigning a unique identification (ID) to each detected object and by using a combination of appearance and motion information to track the object over time. To achieve the objective of our project, which is the counting of vehicles, we need to track all the vehicles accurately, which is a crucial task in this type of application. The tracking of objects involves detecting the object, identifying it in a frame, and tracking it across all sequences until it leaves the scene. To achieve this task, we chose to use the Deep SORT algorithm (simple online and real-time tracking with a deep association metric), which is a multiobject tracking method. The Deep SORT algorithm uses the spatial and temporal characteristics of the targets to track and maintain the identification of all moving objects in all sequences. By assigning each detected object a unique ID, Deep SORT ensures that the same vehicle is consistently identified and tracked across multiple frames, allowing for accurate vehicle-counting.

The Deep SORT algorithm consists of the following steps (as is shown in below figure): First, the vehicles are detected using the YOLOv7-tiny. Then comes the multiobject tracking step, where the Kalman filter predicts the trajectories based on the position and speed of the target object; meanwhile, the Hungarian algorithm measures the correlation

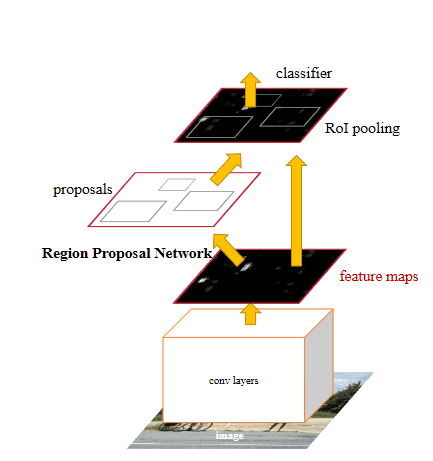


The architecture of the Deep SORT algorithm.

Deep SORT distinguishes itself by using a combination of two convolutional layers, followed by six residual blocks and an appearance descriptor. This makes the system more robust against missing-object occlusions by significantly reducing the number of identity changes. This combination makes Deep SORT an appropriate choice for applications requiring real-time monitoring. Furthermore, Deep SORT has been shown to be effective in handling complex scenarios with significant occlusion, which is a common challenge in tracking vehicles that are using a roundabout. The algorithm is also capable of tracking vehicles with varying speeds and trajectories, making it ideal for real-world applications.

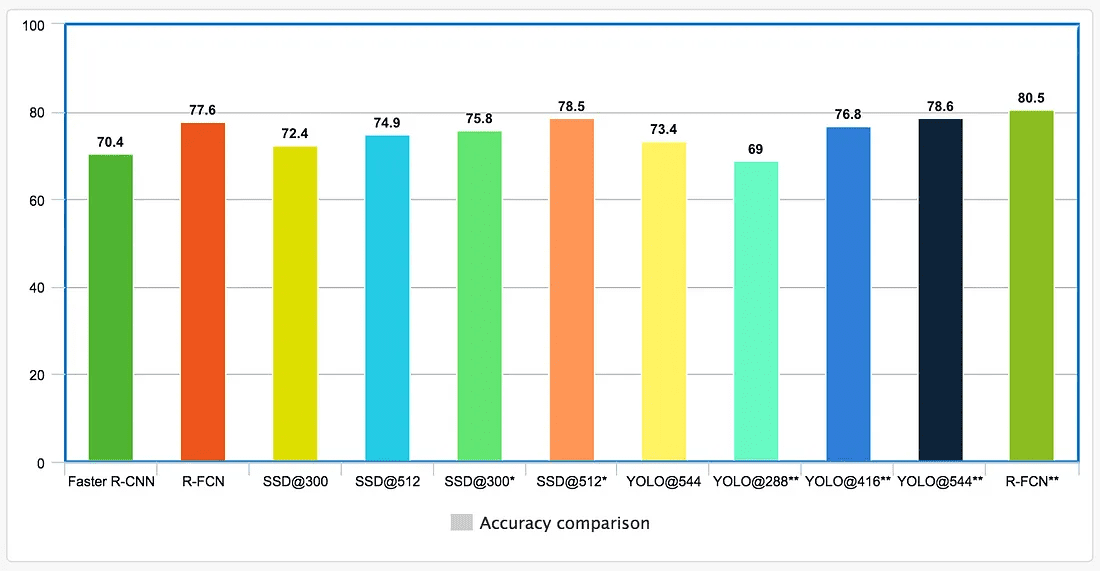
**YOLO**

The key innovation of YOLO is its ability to perform real-time object detection in a single pass through the neural network, making it incredibly fast and efficient. Unlike traditional CNNs, which use complex multi-stage pipelines, YOLO uses a single unified model for both region proposal and classification.

**Faster R-CNN**

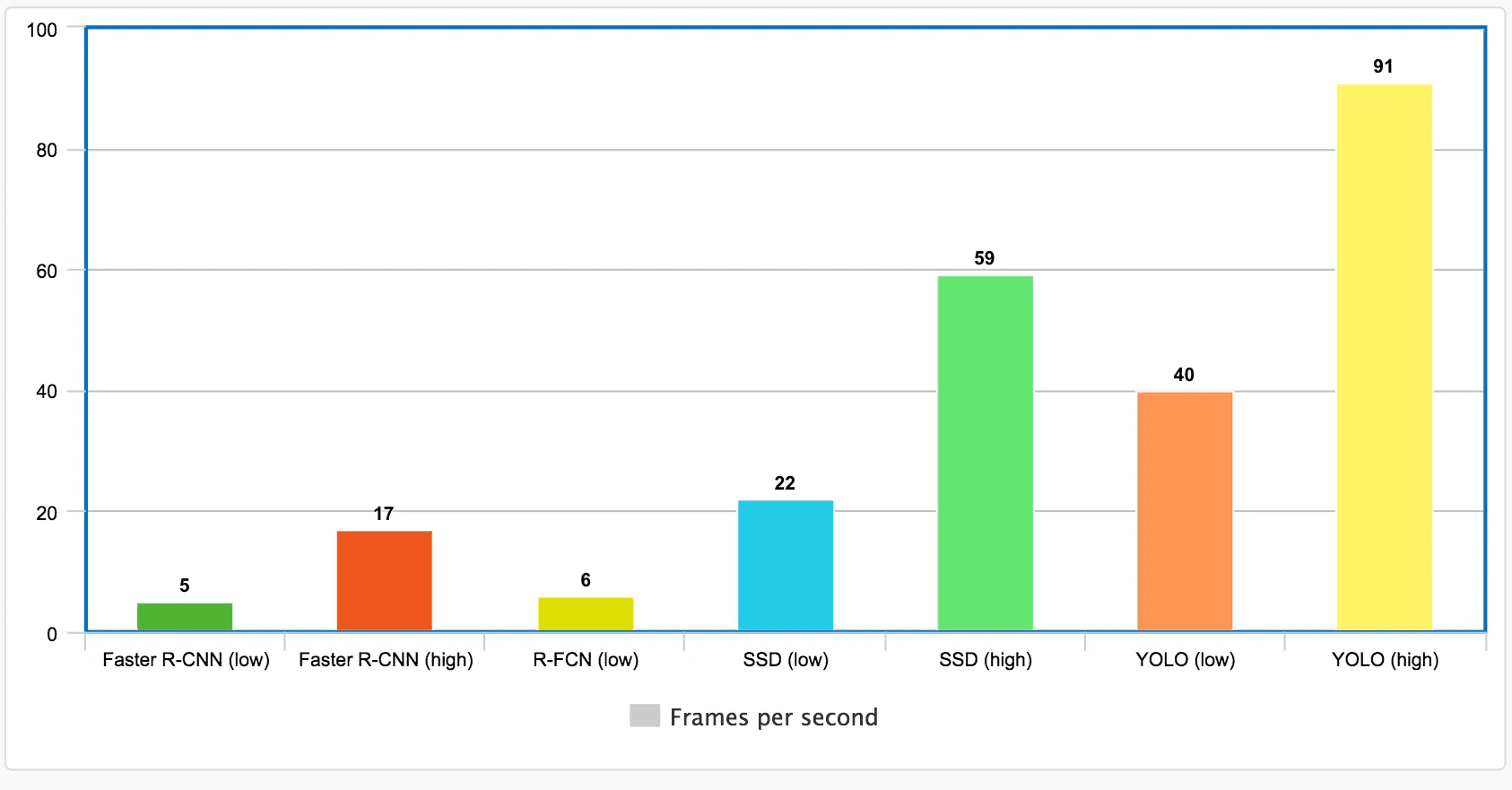
The Faster R-CNN utilizes is a two-stage deep learning object detector: first, it identifies regions of interest and then passes these regions to a convolutional neural network. The outputted feature maps are passed to a support vector machine (SVM) for classification. Regression between predicted bounding boxes and ground truth bounding boxes is computed. Below is the general architecture for the Faster R-CNN:

**Accuracy comparison of all object detection models**

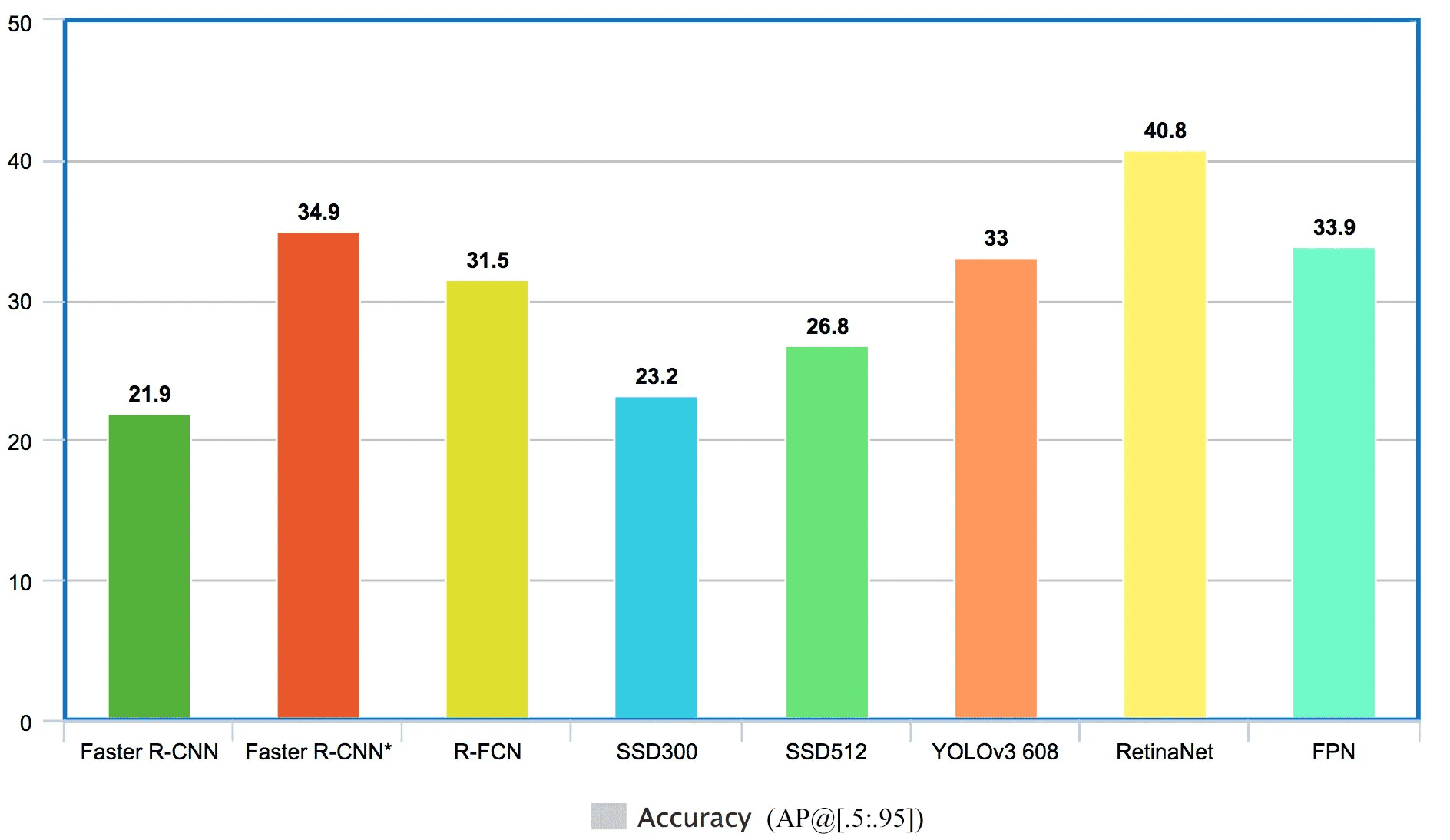


Indicates the results are measured on VOC 2007 testing set. We include those because the YOLO paper misses many VOC 2012 testing results. Since VOC 2007 results are in general performs better than 2012, we add the R-FCN VOC 2007 result as a cross reference.

Input image resolutions and feature extractors impact speed. Below is the highest and lowest FPS reported by the corresponding papers. Yet, the result below can be highly biased in particular they are measured at different mAP.



For the last couple years, many results are exclusively measured with the COCO object detection dataset. COCO dataset is harder for object detection and usually detectors achieve much lower mAP. Here are the comparison for some key detectors.



RetinaNet does not meet the real-time requirements, which limits its potential applications. A Feature Pyramid Network, or FPN, is a feature extractor that takes a single-scale image of an arbitrary size as input, and outputs proportionally sized feature maps at multiple levels, in a fully convolutional fashion. It consists of Residual Network, Feature Pyramid Networks (FPN).

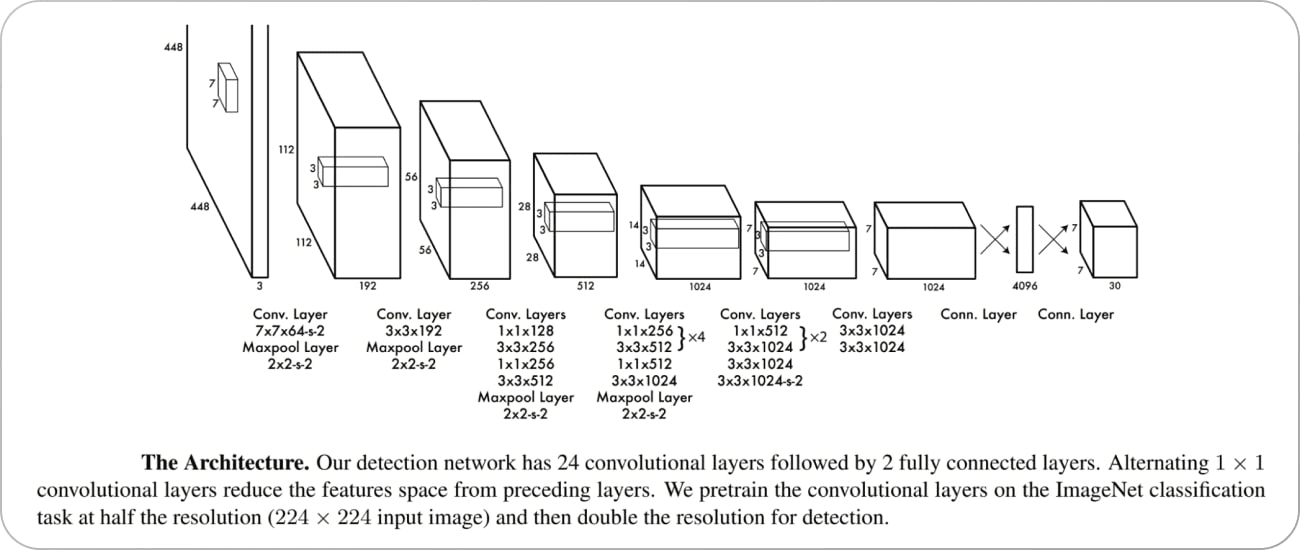
**Methodology:**

## **Why Choose Ultralytics YOLO for Training?**

Here are some compelling reasons to opt for YOLOv8's Train mode:

* **Efficiency:** Make the most out of your hardware, whether you're on a single-GPU setup or scaling across multiple GPUs.
* **Versatility:** Train on custom datasets in addition to readily available ones like COCO, VOC, and ImageNet.
* **User-Friendly:** Simple yet powerful CLI and Python interfaces for a straightforward training experience.
* **Hyperparameter Flexibility:** A broad range of customizable hyperparameters to fine-tune model performance.

**YOLO Object Detection Algorithm: How Does it Work?**



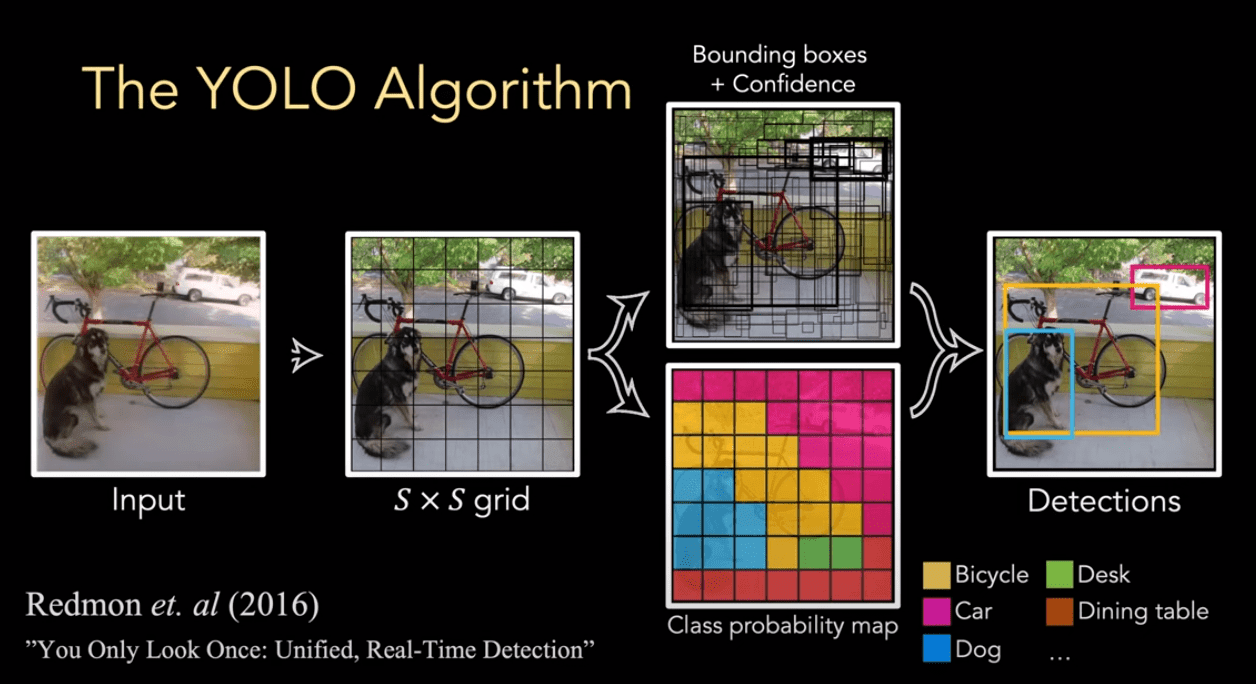
**Bounding Box Recognition Process**

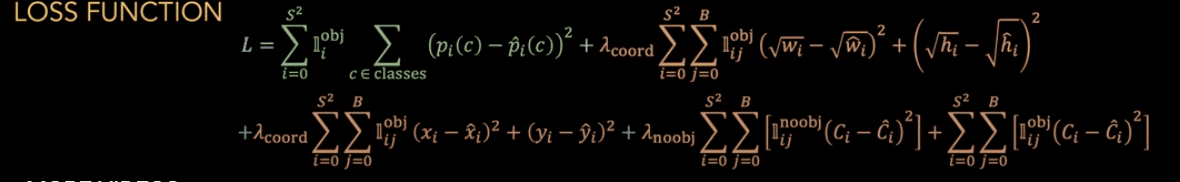
The bounding box recognition process in YOLO involves the following steps:

**Grid Creation:** The image is divided into an SxS grid. Each grid cell is responsible for predicting an object if the object’s center falls within it.

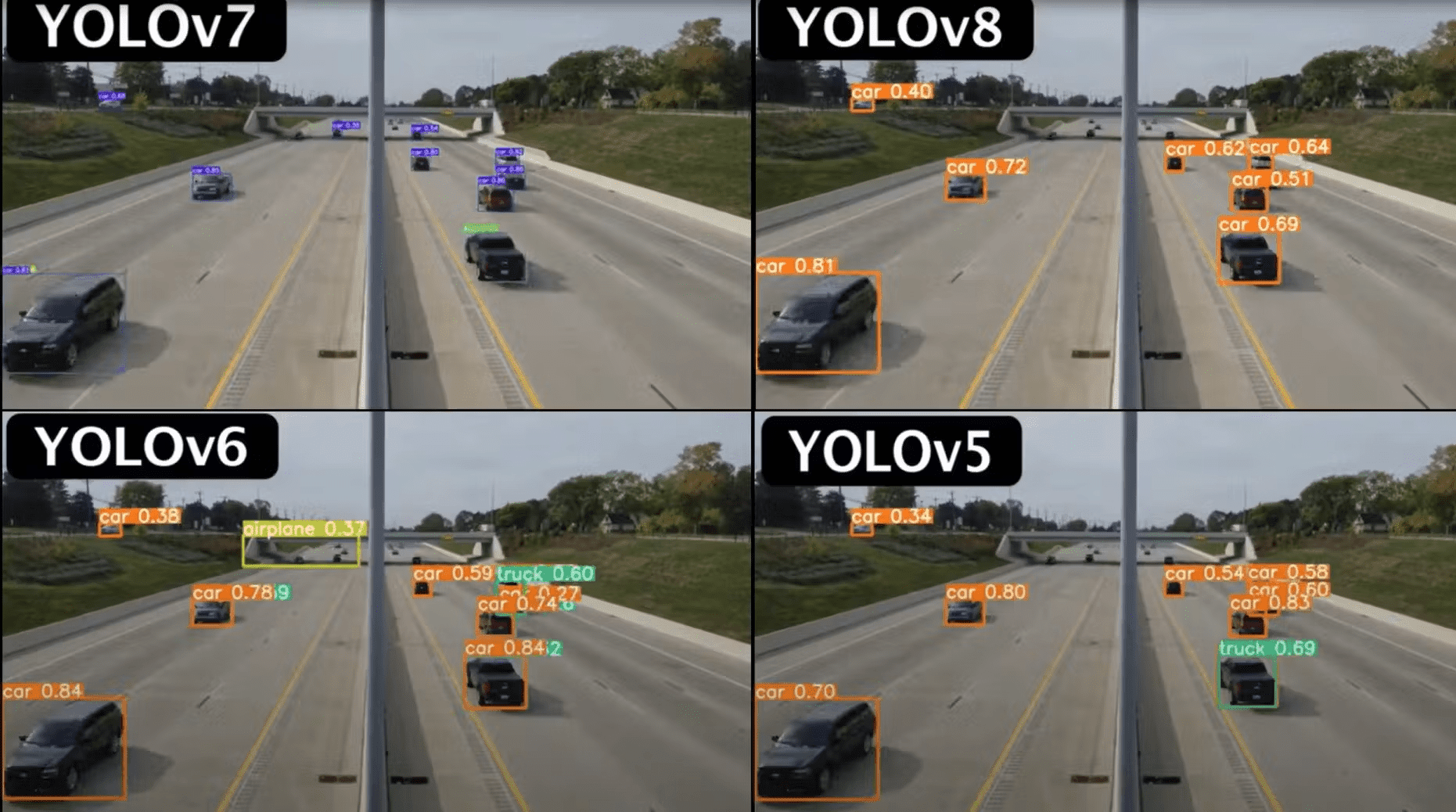
**Bounding Box Prediction:** Each grid cell predicts B bounding boxes and confidence scores for those boxes. The confidence score reflects how certain the model is that a box contains an object and how accurate it thinks the box is.

**Class Probability Prediction**: Each grid cell also predicts C conditional class probabilities (one per class for the potential objects). These probabilities are conditioned on there being an object in the box.





**YOLOv8 Comparison with Latest YOLO models**



**Detecting Vehicles with YOLOv8:**

YOLOv8, short for "You Only Look Once version 8," is a state-of-the-art deep learning model used for object detection tasks. This model is particularly popular for its efficiency and accuracy in detecting various objects in images and videos, including vehicles.

**How YOLOv8 Works:**

YOLOv8, or "You Only Look Once version 8," is an advanced deep learning model used for object detection tasks. It builds upon the principles of its predecessors in the YOLO series to achieve improved accuracy and efficiency. Here's a detailed explanation of how YOLOv8 works:

**1. Single Shot Detection:**

YOLOv8 follows the concept of single-shot detection, meaning it performs object detection in a single pass through the neural network. This is in contrast to two-stage detectors, like Faster R-CNN, which involve separate region proposal and object classification stages.

**2. Grid-based Detection:**

YOLOv8 divides the input image into a grid of cells. Each grid cell is responsible for predicting bounding boxes and associated class probabilities for objects contained within it.

**3. Anchor Boxes:**

To handle objects of various shapes and sizes, YOLOv8 employs anchor boxes. These are pre-defined bounding boxes of different aspect ratios and scales. During training, the model learns to adjust these anchor boxes to better fit the objects in the dataset.

**4. Predictions at Multiple Scales:**

YOLOv8 operates at multiple scales within the network architecture. It makes predictions at different levels of feature maps, allowing it to detect objects of various sizes and scales in the input image.

**5. Convolutional Neural Network (CNN) Backbone:**

YOLOv8 typically utilizes a deep CNN backbone, such as Darknet or ResNet, to extract features from the input image. These features are then passed through additional layers to perform object detection.

**6. Output Format:**

For each grid cell, YOLOv8 predicts multiple bounding boxes (usually 3 to 5) along with confidence scores for each class. The confidence score indicates the likelihood that the bounding box contains an object of a particular class. Additionally, class probabilities are assigned to each bounding box to indicate the likelihood of different object classes.

**7. Non-maximum Suppression (NMS):**

After predictions are made, YOLOv8 applies non-maximum suppression to remove redundant bounding boxes. This technique ensures that only the most confident and accurate bounding boxes are retained, eliminating duplicate detections of the same object.

**8. Loss Function:**

During training, YOLOv8 optimizes its parameters using a combination of localization loss (to minimize errors in bounding box coordinates) and classification loss (to minimize errors in class predictions). This training process enables the model to learn representations that effectively detect and classify objects in images.

**9. Efficiency and Speed:**

YOLOv8 is known for its efficiency and speed, making it suitable for real-time applications. By performing object detection in a single pass through the network and utilizing techniques like anchor boxes and multi-scale predictions, YOLOv8 achieves a balance between accuracy and computational efficiency.

Overall, YOLOv8's design and architecture enable it to efficiently detect objects in images and videos, making it a popular choice for various computer vision tasks, including object detection, tracking, and autonomous driving.

**Vehicle Detection:**

In the context of vehicle detection, YOLOv8 can accurately identify vehicles within a scene, providing bounding box coordinates and confidence scores for each detected vehicle instance. This allows for real-time monitoring of traffic flow, parking lot occupancy, and other applications related to vehicle tracking and management.

**Calculating Distance between Vehicles:**

When analyzing object detection in video data, understanding the spatial relationships between objects within individual frames as well as across multiple frames is crucial. Let's delve into how these distances are computed and their significance:

**Distance between Objects within the Frame:**

In object detection within a single frame, the distance between objects is typically calculated based on their spatial coordinates within the image. This distance can provide insights into the relative positioning of objects, helping to identify patterns or anomalies in their distribution.

For example, in a traffic surveillance scenario, detecting the distances between vehicles within the same frame can be useful for assessing traffic congestion, identifying lane violations, or detecting potential collisions.

**Distance between Electric Vehicles (or objects) from One Frame to Another:**

In video object detection, tracking objects across frames allows us to analyze their motion and behavior over time. The distance between objects detected in consecutive frames reflects the displacement or movement of objects between those frames.

This information is invaluable for various applications, such as:

1. **Motion Analysis:** By tracking the movement of objects over time, we can analyze their trajectories, speed, and direction of motion. This is vital in applications like pedestrian tracking for crowd management or analyzing vehicle flow in traffic studies.

2. **Event Detection:** Changes in the distances between objects across frames can indicate significant events or behaviors. For instance, sudden changes in the distance between vehicles might signal abrupt braking or lane changes.

3. **Behavioral Analysis:** Studying the relative distances between objects over time can reveal patterns in their interactions and behaviors. This is crucial for applications like activity recognition in surveillance or understanding social dynamics in crowded environments.

**Computational Considerations:**

Both types of distance calculations involve processing spatial coordinates and applying mathematical formulas like Euclidean distance. While intra-frame distance calculations are relatively straightforward since they involve objects detected within the same image, inter-frame distance computations require object tracking across consecutive frames, adding complexity to the process.

Efficient algorithms and techniques, such as Kalman filtering, Hungarian algorithm-based assignment, or deep learning-based object tracking, are commonly used to address these challenges and accurately compute distances between objects across frames.

Calculating the distance between vehicles in a video involves several steps within the context of object detection:

**1. Object Detection:**

Initially, the video frames are analyzed using an object detection model such as YOLOv8 to identify and localize vehicles. This process involves predicting bounding boxes around vehicles along with their associated confidence scores.

**2. Extracting Vehicle Positions:**

Once the vehicles are detected, the coordinates of their bounding boxes are extracted. These coordinates represent the positions of the vehicles within the image frame.

**3. Calculating Distance:**

With the coordinates of the bounding boxes, the distance between vehicles can be computed. This can be achieved using various methods:

- **Euclidean Distance:** The Euclidean distance formula calculates the straight-line distance between two points in a two-dimensional space. For vehicle detection, the coordinates of the centers of the bounding boxes for each pair of vehicles can be used to compute the distance.

- **Pixel-based Distance:** In this method, the number of pixels separating the centers of the bounding boxes is measured. This distance can be directly converted to real-world units (e.g., meters) using a known scale factor.

**4. Real-world Distance Conversion:**

To obtain the distance in real-world units (such as meters), the pixel-based distance is converted using a conversion factor. This factor depends on factors such as the camera's focal length, sensor size, and the distance between the camera and the scene.

By accurately calculating the distance between vehicles in a video, it becomes possible to monitor their spacing and detect potential collision scenarios, thereby enhancing safety and situational awareness in various applications such as traffic management, autonomous driving, and surveillance.

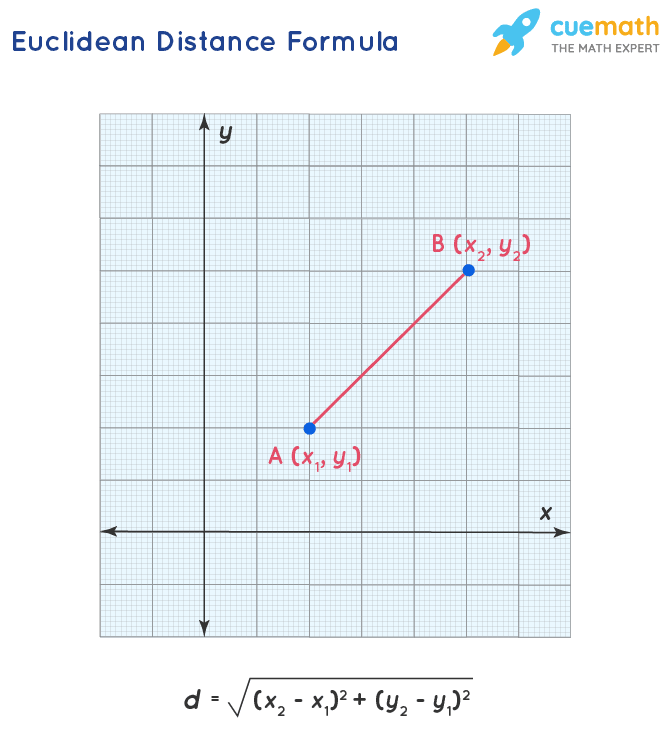
**Warning for Close Proximity:**

Calculating the distance between vehicles enables the implementation of a warning system for close proximity. By setting a threshold distance, the system can trigger a warning or alert when vehicles are too close to each other. This feature enhances safety in scenarios such as traffic congestion, lane merging, or parking maneuvers, helping to prevent accidents and collisions.

**Speed Calculation of Electric vehicles:**

1. Loading the YOLO model: It loads the YOLO model from a pre-trained file (yolov8n.pt).
2. Reading the video: It reads a video file containing footage of vehicles.
3. Object detection: For each frame of the video, the YOLO model detects vehicles and returns their bounding boxes (coordinates) and class labels.
4. Speed calculation:To calculate the speed of a vehicle between two frames, we need to measure how far it has traveled.

* We use the Euclidean distance formula, which calculates the straight-line distance between two points in a plane. In this case, the points represent the center of the detected vehicle in the current frame and the center of the same vehicle in the previous frame.

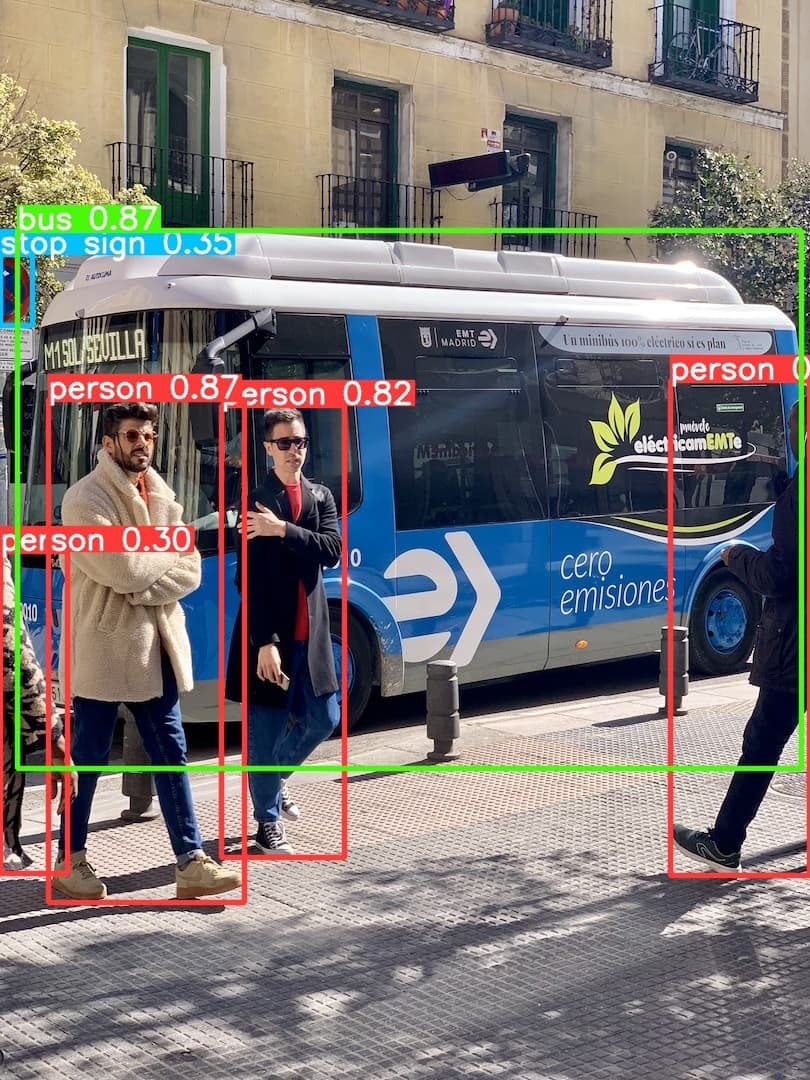


* The formula for Euclidean distance between two points (*x*1,*y*1) and (*x*2,*y*2) is:  
  Distance=sqrt((*x*2−*x*1)2+(*y*2−*y*1)2)
* Once we have the distance traveled by the vehicle between frames, we convert it to meters per second (m/s). This is done by dividing the distance by the time difference between the two frames (in seconds) and then multipying it with number of frames to obtain the overall speed of the moving vehicle.
* Finally, we convert the speed from meters per second to kilometers per hour (km/h) by multiplying it by a conversion factor (3.6). This gives us a more understandable measure of speed.

1. Displaying the results: After calculating the speed for each detected vehicle, we print these speeds to the console. This allows us to see the speed of each vehicle as the program runs.
2. Updating previous frame data:

* To calculate the speed of each vehicle in subsequent frames, we need to compare their positions in the current frame with their positions in the previous frame.
* We store the detected objects (vehicles) and the time stamp of each frame. This information is used in the next iteration of the loop to calculate the speed of each vehicle.
* By updating the previous frame data, we ensure that we always have the necessary information to calculate the speed of vehicles between consecutive frames.

**Results:**



**Object detection with accuracy using yolov8**

**Speed of the detected moving vehicles**

Vehicle ID 14 (car): speed is 3.48 km/h.  
  
Vehicle ID 5 (car): speed is 69.01 km/h.  
  
Vehicle ID 10 (car): speed is 4.56 km/h.  
  
Vehicle ID 11 (car): speed is 3.36 km/h.  
  
Vehicle ID 12 (car): speed is 62.93 km/h.  
  
Vehicle ID 13 (truck): speed is 45.22 km/h.  
  
Vehicle ID 14 (car): speed is 3.48 km/h.  
  
Vehicle ID 11 (car): speed is 3.36 km/h.  
  
Vehicle ID 12 (car): speed is 62.93 km/h.  
  
Vehicle ID 13 (truck): speed is 45.22 km/h.  
  
Vehicle ID 11 (car): speed is 3.36 km/h.  
  
Vehicle ID 12 (car): speed is 62.93 km/h.  
  
Vehicle ID 13 (truck): speed is 45.22 km/h.  
  
Vehicle ID 5 (car): speed is 69.01 km/h.  
  
Vehicle ID 11 (car): speed is 3.36 km/h.  
  
Vehicle ID 12 (car): speed is 62.93 km/h.  
  
Vehicle ID 5 (car): speed is 69.01 km/h.

**Proximity distance display**

Distance between ('car', 5) and ('car', 1) in the frame: 0.02627246888321414 meters

Distance between ('car', 5) and ('car', 3) in the frame: 0.09833283817980991 meters

Distance between ('car', 5) and ('car', 6) in the frame: 0.0992275565513929 meters

Distance between ('car', 5) and ('car', 8) in the frame: 0.03158198203024795 meters

Distance between ('car', 6) and ('car', 0) in the frame: 0.0930479694083111 meters

Distance between ('car', 6) and ('car', 3) in the frame: 0.004620748684451627 meters

Distance between ('car', 6) and ('car', 4) in the frame: 0.021952443076936078 meters

**Conclusion:**

In summary, the analysis of both intra-frame and inter-frame distances in video object detection allows for a comprehensive understanding of object relationships, motion patterns, and behavioral dynamics. By incorporating speed calculation into this framework, we enhance our ability to interpret and utilize the spatial and temporal information extracted from video data.Intra-frame distance analysis provides valuable insights into the spatial distribution of objects within individual frames. By calculating distances between objects detected within the same frame, we can assess their relative positioning and identify patterns or anomalies in their distribution. Additionally, incorporating speed calculation allows us to infer the velocity of objects within a frame, providing further context for their interactions and behaviors.

Inter-frame distance computations enable the tracking of object movement and behavior over time. By analyzing changes in distances between objects across consecutive frames, we can detect motion patterns, monitor object trajectories, and identify significant events or behaviors. Integrating speed calculation into this analysis allows us to quantify the speed and direction of object movement, facilitating applications such as traffic flow analysis, pedestrian tracking, and activity recognition.Overall, the combination of intra-frame and inter-frame distance analysis, coupled with speed calculation, enables a holistic understanding of object dynamics in video data. This comprehensive approach enhances the capabilities of surveillance systems, transportation management systems, and safety applications, empowering stakeholders to make informed decisions and mitigate risks effectively.

**References:**

<https://github.com/roboflow/supervision/tree/develop/examples/speed_estimation>

<https://jonathan-hui.medium.com/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359>

<https://encord.com/blog/yolo-object-detection-guide/#:~:text=YOLO%20Architecture,certain%20number%20of%20bounding%20boxes.>