2D Multi-Vehicle Tracking in Carla Simulator

Bhavana Sornapudi
Department of Computer Science
University of Exeter
Exeter, UK
bs571@exeter.ac.uk

Wu Han
Department of Computer Science
University of Exeter
Exeter, UK
hw630@exeter.ac.uk

Nick Ross
Department of Computer Science
University of Exeter
Exeter, UK
n.d.f.ross2@exeter.ac.uk

***Abstract*—Autonomous vehicles also known as self-driving vehicles are a curious and intriguing concept in the modern world. These vehicles understand their surrounding without any human interventions, all while seamlessly adapting to the ever-shifting tapestry of their environment. The journey towards fully functional autonomous vehicle comes with formidable challenges along with several technological hurdles and require ingenious solutions. At the heart of autonomous vehicle technology is Multi-Object Tracking (MOT) that can adapt to environmental changes [1]. In this paper, we present an approach for 2D multi-vehicle tracking using the YOLOv8(You Only Look Once) model combined with Deep-SORT (which is an extension of the original SORT) within the Carla simulator. By integrating YOLOv8's real-time object detection capabilities with Deep-SORT's advanced data association and tracking abilities, we can track multiple vehicles simultaneously and accurately. The fusion of these two deep learning techniques with real-time simulations in Carla's digital realm allows us a vantage point to validate the behavior of autonomous vehicles in different weather and complex traffic conditions. In conclusion, the performance of YOLOv8 and Deep SORT for multi-object tracking for autonomous vehicles can be evaluated using metrics like MOTA and MOTP (accuracy and precision) [2]. These metrics give us an objective measure of the tracking system. The developed tracking model not only enhances the state of the research in the field of autonomous vehicles but also adds to the safety and dependability of autonomous vehicles in the real world. Overall in this thesis, we are providing a comprehensive solution to real-time vehicle tracking in the Carla simulator using deep learning technologies.**
Keywords - YOLOV8, Deep-SORT, Carla , MOT(Multi-Object Tracking) .

I. Introduction

There has been a rise in the number of automobiles on the streets in recent years has led to a rise in the number of accidents. Accidents caused by human errors can be significantly reduced using autonomous vehicles [3]. For this reason, it is imperative to develop a sophisticated system capable of simultaneously detecting and tracking multiple vehicles on the road.

Research on autonomous vehicles has gained significant momentum in recent decades due to its potential to transform transportation industries. Autonomous vehicles were developed with the sole purpose of reducing human intervention. These vehicles utilize highly accurate and dependable sensors to detect their surroundings, thereby enhancing driving safety, precision, and dependability. Numerous studies on autonomous vehicles are published every year. These vehicles are fully equipped with cutting-edge sensors, including thermal imaging cameras, radar, LiDAR, GPS, optical sensors, and plenty more that assist in developing a 2-D/3-D simulation [4]. Autonomous vehicles identify the road and nearby obstacles for navigation and speed control to reduce traffic problems and improve safety. Besides, autonomous vehicles provide a driving solution for those who cannot drive, such as the elderly. However, all this comes with several challenges and risks. The main reason for doubts about this technology is navigation and object tracking. The technology will fail if the simulated 2-D/3-D model cannot differentiate between a pedestrian or a tree on the roadside or another driving vehicle [5]. The ability to track multiple vehicles is an absolute must for self-driving cars. The importance of dependable object detection and tracking cannot be overstated. In order to operate at maximum efficiency, real-time monitoring of objects, especially vehicles, is absolutely essential.

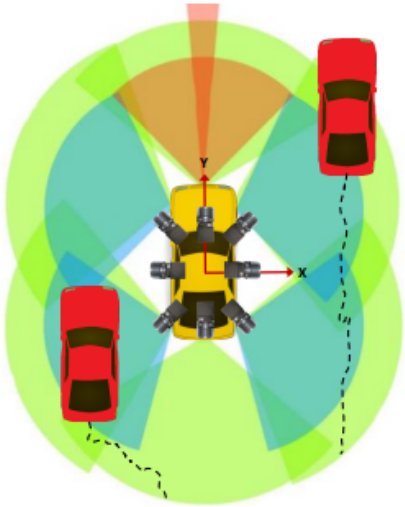

Fig. 1. This image illustrates the MOT of an autonomous vehicle, where the red vehicles are tracked by the ego vehicle in the center using different sensors.

Experiments on autonomous vehicles have started since 1925 with promising outcomes in the 1950s. Since then, several tech giants and research institutes have developed working models. All these endeavors led to a transition of autonomous vehicles from testing environments to real-world transportation [3]. As per Mckinsey's recent survey on

autonomous vehicles, the research in this field is growing exponentially and autonomous driving has the potential to provide numerous benefits to customers and transportation industries. Currently, most cars only include basic ADAS (Advanced Driver Assistance System) features, but significant advancements in self-driving vehicle capabilities are on the horizon.

The real-time simulations can be generated using the Carla simulator. Carla is an open-source simulator designed for autonomous driving research. It offers a digital platform with various layouts, buildings, and vehicles for this purpose. Carla [6] is highly flexible with the environments that one can simulate, which makes controlled testing for self-driving vehicles easy. The monocular camera can visually detect the vehicles in the simulated environment. The monocular camera is a single-lens visual sensor that emerges as a subtle but vital sentinel. Despite the broad spectrum of sensors available, the monocular camera stands out in providing numerous perspectives into Carla's virtual surroundings. Its adaptability in Carla Simulator with several parameters like the field of vision (fov) [6], resolution, and exposure allows developers to generate real-time conditions in the virtual world.

In the last few years, Deep learning (DL) models more precisely Convolution Neural networks (CNN) have demonstrated dramatic improvement over traditional methods [3]. Multi-object tracking involves a number of phases including object detection, assigning id to the detected objects, and classifying objects. In this report, we are using YOLOv8 [7] a deep learning model for visual recognition is used to detect objects. The feed from the monocular camera with a stream or a video will become akin to the object detection model. This involves scaling, normalization, and transformation of data for efficient neural network computation before the frame enters YOLO. In the later part the tracking is done by other deep learning model based on Kalman filter Deep SORT (Simple Online Real-time Tracking) which overcomes several limitations of SORT [8] like identity switches, addressing detection failures, more sophisticated data association techniques.

The adaptability of the approach is clearly demonstrated through its capability to simultaneously track multiple objects in the Carla simulator. Employing cutting-edge deep-learning methodologies, it effectively addresses challenges pertaining to object detection, tracking, and navigation, thereby significantly enhancing the precision and security of self-driving systems.

II. BACK GROUND & LITERATURE REVIEW

A. Carla simulator

The CARLA (Car Learning to Act) [9] simulator was collaboratively developed by a team from the Computer Vision Centre (CVC) and the Universitat Autonoma de Barcelona (UAB) to provide a platform for the development, training, and validation of autonomous driving systems. The high level of realism combined with diverse scenarios with dynamic traffic, controlled weather conditions, and various may layouts are the key strengths enabling it to be used for validating and testing autonomous vehicles. Carla is an API, which can be customized with a nearly real backdrop, several sensors for speed, position, orientation, and depth calculation along with a few more opening larger possibilities to create multiple scenarios. These can be as close as possible to real-world scenarios, making the virtual environment risk-free.

Carla Environments

Fig. 2. Illustrates different virtual environments that can be simulated in Carla using weather controls.

In simple, CARLA is a powerful tool for researchers and developers working on autonomous driving systems. It's realistic simulations and flexible API make it an ideal platform for testing and validating autonomous driving models across a wide range of scenarios.

B. Sensor - Monocular camera

Within the Carla simulator, a diverse range of cameras assumes a fundamental role in replicating real-world vision and enhancing the perception capabilities of autonomous vehicles [9]. Carla has a range of cameras with various characteristics like monocular, stereo, and semantic segmentation. All the cameras can be tailored to meet specific requirements, and this diversity facilitates in assessment of detection, tracking, lane detection, and a few other tasks in the controller's virtual environment.

Our project effectively uses a monocular camera (RGB) [10] in the Carla simulator. This camera captures 2D visual information from a singular perspective and possesses intrinsic properties, such as principal point, focal length, and distortion coefficient, which define its optical characteristics. Moreover, we can adjust the extrinsic properties of the camera, such as its position, orientation, and placement with respect to the vehicle.

C. Object Detection

Real-time object detection remains a challenge due to variations in object spatial size and aspect ratio, inference, speed, and noise. [5] Detecting moving objects has traditionally been done by subtracting and feature extracting using SIFT or HOG descriptors. However, these methods have a high error rate due to variations in scale and appearance along with noise, occlusion, and lighting conditions. Convolution Neural Networks performed object detection and outperformed traditional techniques.

There are three types of object detection algorithms: single-stage, two-stage, and point cloud (3D LIDAR)-based detectors.

Single-stage detectors: These models can recognize objects in images and videos directly in a single pass. Some models are YOLO, SSD.

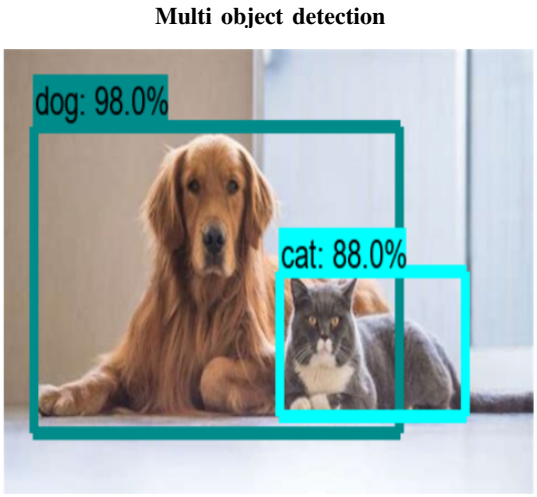

Fig. 3. Object detection is applied on an image with different objects. The bounding boxes and the confidence score can be seen.

Two-stage detectors: [5] These models can recognize objects in two phases, firstly by identifying where the region of interest is present followed by predicting refined bounding box coordinates. The bounding boxes can be infinite, then a second classifier only processes the region candidates. Some examples are R-CNN, Faster R-CNN, and Mask R-CNN.

Point clouds-based detectors: [11] Unlike the previously mentioned model, these detectors require data from 3D LIDAR that provides depth information along with dense features from an image or video. Some models are PointNet , Vote3D , VeloFCN

In 2D object detection [12] which revolves around determining the position and label of every object to extract the bounding box details, later this information will be a base for tracking models and making decision-making. There is some research stating that single-shot detectors are more efficient due to the simple architectures, but two-stage detectors take the lead in accuracy.

D. YOLOv8

YOLO was developed in 2015 by Joseph Redmon and Ali Farhadi. Since its inception in 2015 [13] [7] the YOLO series has always evolved and refined. The first version YOLO1 was efficient and very fast, which made it a state-of-the-art detection model when compared to other models like R-CNN. Several researchers have contributed to the development of YOLO and improvised it in different areas. YOLOv2 and YOLOv3 both were developed by original author Joseph Redmon. YOLOv4 was developed by Alexey Bochkovskiy, Chien Yao Wang, and Hong-Yuan Mark Liao.

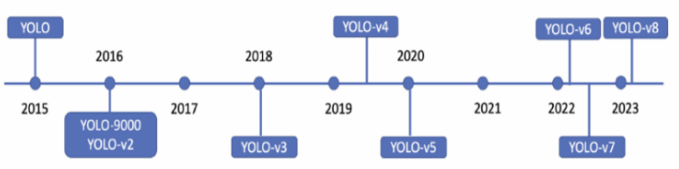

Fig. 4. The YOLO timeline in recent years. [1]

Later YOLOv5 was developed by Ultralytics, which was the precursor model for YOLOv8 [13] [14] [7]. YOLOV8 is released in January 2023. This is the latest version, from Ultralytics with new features and enhancements which increased the accuracy of the model by increasing augmentation during the training process. YOLOv8 can perform several tasks like detection, segmentation, pose estimation, tracking, and classification. This diverse scope of capabilities allows users with the ability to leverage YOLOv8 across different applications.

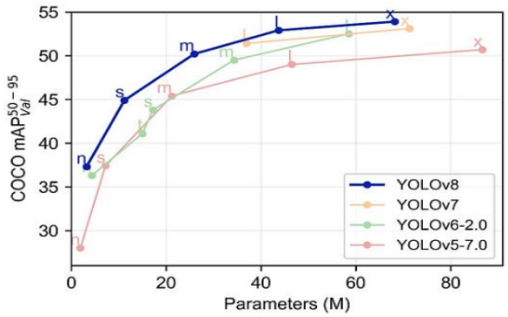

Fig. 5. shows that YOLOv8 model performs better compared to other models [15]

YOLOv8 uses a similar backbone as YOLOv5 with some changes on CSPLayer [16] [13] [14] or C2f module. Within the C2f module, two convolution layers effectively blend high-level features and contextual information, leading to a marked increase in detection accuracy. YOLOv8 boasts five scaled versions - YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), and YOLOv8x (extra-large) - to cater to varying requirements based on the number of classes to classify and the dataset size.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Fig. 6. Details of scaled versions of YOLOv8 model [14] which shows different performance stats of individual models.

E. YOLOv8 Architecture

YOLOv8 is the latest version in the YOLO family which is developed by Ultralytics. The architecture of YOLOv8 can be divided into two main parts: (i)backbone and (ii)head. The backbone is a modified version of CSPDarknet53 architecture [15] [13], which has 53 convolution neural network layers and cross-staged partial connections. These layers improve the flow of data between the layers easily. On the other hand, the Head of YOLO consists of multiple convolution layers followed by a series of fully connected layers. These layers are responsible for detecting and predicting bounding boxes and confidence scores of the objects detected in the image.

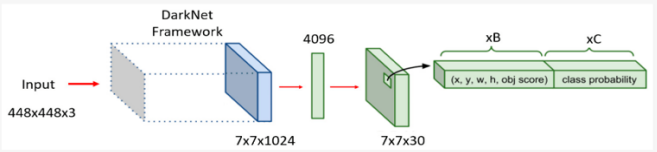

Fig. 7. Base architecture of YOLO family. [15].All the YOLO modes were designed in a similar fashion, with enhancements making each version more accurate than its previous one.

YOLOv8 [14] uses a self-attention mechanism in the network head to focus on different parts of the image and adjust feature importance.

F. Multi-Object Tracking

Multi-object tracking is a crucial component of vehicle tracking for autonomous vehicles. It enables vehicles to identify, recognize, and classify different objects in surrounding environments. Object tracking is following single or multiple objects in a sequence of frames from a video or from an image using their spatial and temporal features. The goal is to locate the object of interest and monitor its position over time, while dealing with challenges like occlusion [**?**], change in orientation, speed, scale, etc. Traditional object tracking methods include feature-based which analyzes object characteristics, segmentation-based which segment objects from the background, and learning-based methods which use predefined models to predict trajectory. These methods provide the foundation and help us understand how object-tracking models evolved.

Multi-object tracking is classified into multiple ways, one way is how they extract data from the frame and track the objects present in it. Based on this we have three classifications: (i). point-based tracking, (ii) Kernel Based Tracking, and (iii). Silhouette Based tracking [17]. Particle-based tracking represents objects as particles and uses statistical models to track objects.

Tracking by detection (TBD): In contrast to traditional methods, models based on TBD integrate object detection as an initial step. an independent detection model is applied to all the available frames to obtain detection. Then a tracker model will run on a set of detection and attempts to perform data associations [18] [17]. Because of processing in two stages, these models overcome several challenges like occlusion, and scale variations leading to accurate results compared to traditional methods. Deep SORT is a TBD-based model, that uses a Kalman filter. It starts by detecting objects using a deep neural network commonly a convolution neural network (CNN). Then the detected objects are fed into the Kalman filter-based framework. Researchers are working hard to improve the efficiency and accuracy of object-tracking models used in autonomous vehicles, trying to make vehicles suitable for busy and complex roads.

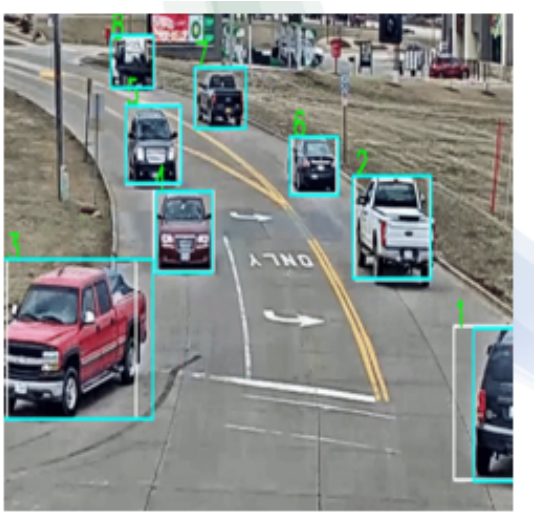

Fig. 8. Example Multi-object tracking

G. 2D Vs 3D MOT

For this project, 2D tracking is used to monitor object movements within a 2D [19] plane based on their motion characteristics. While 3D tracking demands additional data such as object position, depth, and orientation for accurate detection and tracking, to estimate these we need to use LIDAR and stereo camera which are computationally expensive and might reduce overall efficiency [17]. The decision to use 2D tracking is motivated by its inherent speed and computational efficiency. Moreover, 2D tracking exhibits greater resilience in scenarios where depth information is unavailable.

H. Deep SORT

In the realm of multi-object tracking Deep SORT is the biggest leap of advancement. It is based on the classic concept of SORT (Simple Online and Real-time Tracking) [8]. SORT is designed to track objects based on the object-identifying principle but was made simple to avoid complex and time-consuming tasks. Deep SORT is developed by combining the SORT technique with a deep convolution Neural Network (CNN) for the faster, more complex task of MOT. Because of the use of deep learning techniques, Deep SORT is more accurate, robust and resolves the issue of ID switching and tracking object occlusion. Deep SORT [8] [20] is an offline deep learning model based on TBD, which is trained using

REID data sets. It is developed by Alex Bewley and gained a lot of traction for its ability to track objects over frames in videos sequence in 2018. Below are the components of Deep SORT.

Detection: Detection is the initial step of any tracking by detection-based tracking models. It involves using an object detection model to identify and locate objects of interest within individual frames. This can be done by R-CNN, Faster R-CNN, or YOLO model [8]. An appearance descriptor is a feature vector that represents the object's appearance. this feature vector is generated from the detected bounding boxes from the image.

Matching cascade: Deep SORT using cascade matching to associate objects in multiple previous frames [20]. In other words, it refines data associations by considering both spatial and temporal relationships between objects in consecutive frames. This matching improves tracking quality by reducing identity switches and maintaining consistent trajectories.

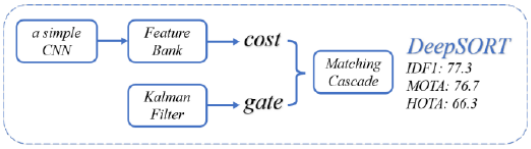

Fig. 9. Simple Deep SORT algorithm. [20]

Cost matrix: Intersection over union, is used as a distance metric used to associate defections and tracks in the consecutive frames. A cost matrix is computed as the distance between each track and detection from the existing targets. If the computed IOU value is less than a certain IOU threshold then the detection and track pair are ignored.

Kalman filter: Once the object is detected, the estimations component predicts the future position based on the state of the previous position and motion. Kalman filter [21] is used for this purpose. This is a linear filter and assumes the same for noise for all the objects. Kalman filter for object tracking uses the information to co-relate objects in consecutive frames. For instance, object A is detected in from t, and object B is detected in the frame (t+1), both A and B are associated as some object using IOU (Intersection over union)

Data Association: The association between defections and tracks is resolved using the linear assignment [8] technique. It uses the appearance features output for bounding boxes detected by the detection model. Data association also uses a matching cascade, which uses to improve the accuracy and robustness of the system.

The feature vectors [8] are compared to the existing tracks using a cosine similarity matrix. The matching cascade for data association where tracks and detection were matched, at each stage the algorithm uses to find the match for each track with detection and each detection with a track. All the unmatched tracks and detections were removed leaving with a pair of track and detection, from where a tracking ID is assigned based on the IoU threshold.

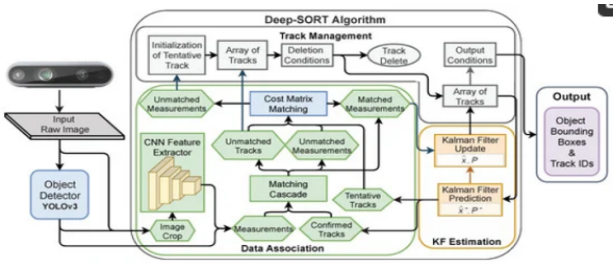

Fig. 10. Simple Deep SORT architecture. [18]

III. AIMS & OBJECTIVES

A. Aim

The aim of this project is to create a real-time vehicle tracking system employing camera image analysis in the Carla simulator. Simply expressed, the project's goal is to provide a deep learning-based system for multi-object tracking in autonomous cars that uses 2D data to assess its performance with the help of YOLOv8 and Deep SORT in the Carla simulator's virtual environment for autonomous vehicles.

B. Objectives

- First step is to develop a YOLOv8 model and train it with compatible data sets, that are tailor-made for YOLO, allowing it to detect multiple objects of different classes within the simulated environment. This will be integrated with the tracking model in later stages.
- The second objective is to implement the Deep SORT algorithm which is known for its strong multi-object tracking abilities. Once this is developed, we have a multi-object tracking system.
- The developed tracking system, needs to be integrated with the Carla simulator for validation and testing our model in various conditions that imitate real-world scenarios.
- The analysis of different metrics like MOTA and MOTP is generated by comparing the ground truth with the tracking model output to evaluate the tracking model.
  By achieving these objectives the project strives to establish an innovative multi-vehicle tracking system. This solution aims to recognize, monitor and forecast vehicle movement in the Carla simulator.

IV. DATA SET

The data sets are a big source of positive concern in the field of visual multi-object tracking. These data sets are particularly difficult to deal with since they are often updated and created to closely match real-world scenarios. [22] To more effectively manage huge amounts of data and enhance the precision of their monitoring systems, researchers continuously strive to enhance their methods and algorithms. The MOT data sets are frequently the first choice when it comes to data sets for creating and evaluating autonomous car technology. However, several of these data sets, such as MOT16 and MOT20 are largely focused on pedestrian tracking. There are several other

data sets from KITTI, Nu-Scenes which have huge amounts of data

Pascal VOC , Ms-Coco , ILSRVC [22] along with Open Image are different formats of data sets that can be used for MOT problems. Of all these Open Image and ILSRVC are the largest data sets with over a million images in each. Even though Pascal VOC and Coco have thousands of images, they stand small in comparison with the previous one. Coco Data set has 80+ different object classifications and has been trained and validated on over 20000 images.

For this project, we are using a dataset from Kaggle [23], which was curated by Roboflow which are collected from the CARLA simulator. This data set has been designed with 10 different classes like “traffic signals”, “bikes”, “motorbikes” and vehicles like “cars”, “trucks”, and “buses”. In most of the data sets all the 4 Wheeler’s are designed to be under a super-class vehicle. It consists of 1800+ images, which are annotated in COCO format

A. Data preparation

The data set is categorized into two sections one for training and the other for validation. each set has a group of images along with a text file, that contains a bounding box, confidence score, and class id. All the information in the text file is used to classify and identify the objects of multiple classes.

The data set available should be cleaned before using it to train the YOLO model. Initially, all the file names were cleaned such that there are no spaces or special characters. The bounding box details are in normalized format (xcenter, ycenter, width, height). These were converted to non-normalized pixel format (xmin,ymin,xmax,ymin). All this information is stored in a data frame. For training the YOLO model we have excluded irrelevant classes like traffic lights since we need a model that detects vehicles

V. Model Implementations

A thorough experiment has been conducted involving object detection and object tracking in the Carla simulator under different real-time conditions. This section provides a detailed analysis of training, implementation, and integration of the detection and tracking model with Carla

A. Carla Integration

To implement the trained model of YOLOv8 in Carla to identify the multiple vehicles, Carla needs to be set up [16]. A series of actions need to be performed for this. Carla is a highly resource-required platform. It needs a minimum of 6GB GPU [24]. All the code that is used to do any operations in the Carla simulator, can be termed boilerplate code since it remains the same for all instances. These were provided by Carla and are available from Carla docs [24].

- Carla acts as a server-client connection. once the Carla simulator is launched and the connection is established with port 2000 the world can be configured.
- The blueprints of cameras, and vehicles can be used to design and customize as per requirement. For this project, we are using a monocular RGB camera attached at the arm-length position.

B. YOLOv8 MODEL Training

Once the dataset is formatted to fit the YOLOv8 standards it can be used to train the detection model to generate best weights. This deep learning model has gained prominence for its superior object detection performance. The model uses pre-trained weights for the initial runs on the image dataset for fast convergence.

YOLOv8 [14] model comes with salable sizes from nana to extra-large. The size of the model can be objective based on requirements. Smaller models perform better and enhance detection speed, while larger models increase detection accuracy. We have trained our detection model in nano, small, and extra-large sizes to validate and generate the best weights that can be used during object tracking. YOLO can be run on a CPU, but for this project, I am running it on CUDA with Torch to train the model in a smaller amount of time.

```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
   332/1000      2.12G     0.6794     0.3943     0.9295         17        640: 100%|████████| 49/49 [00:08<00:00,  5.
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% ████████| 4/4 [00:03<0
                   all        106        187      0.893      0.885       0.95      0.716
Stopping training early as no improvement observed in last 50 epochs. Best results observed at epoch 282, best model saved as
best.pt.
To update EarlyStopping(patience=50) pass a new patience value, i.e. `patience=300` or use `patience=0` to disable EarlyStopp
ing.

332 epochs completed in 1.370 hours.
Optimizer stripped from runs\detect\train\weights\last.pt, 6.2MB
Optimizer stripped from runs\detect\train\weights\best.pt, 6.2MB
```

Fig. 11. Running of last epochs of YOLO training on custom dataset.

In the training processes, [14] the dataset was divided into small batches named mini-batches containing pre-processed images. Based on these images, the detection algorithm predicts bounding boxes, confidence scores, and class labels. The models’ predictive capabilities were adjusted in the last 20-25 runs using back-propagation via gradient descent and adjusting the weights. The model needs to be trained for an adequate number of runs until a convergence point is observed. The model was run for 1000 runs for both YOLOv8N and YOLOv8X sizes. The model will terminate itself if there is no improvement in performance in the last 50 epochs.

1) Training Performance: This model can be validated using other sets of images, and the results can be seen analyzed using several plots of recall, F1 confidence, precision and mean Average Precision(mAP) along with train-loss plots. All these were discussed in the Results section of this paper.

C. YOLO Detection on CARLA

Once the Carla setup is done, the developed model can be integrated to detect the objects in the Carla simulator. Individual frames from the Carla simulator are fed to the YOLO model for detection [16]. We are using the weights generated during the training process. Each frame coming from the Carla simulator has been resized which is the input convolution neural networks of the detection model. A collection of anchor boxes are generated with different shapes and sizes, which forecast the bounding boxes in xyxy format.

Output predictions: These bounding boxes are in non-normalized format with coordinates from the top-left point and

Metrics	Values
F1 Score	0.79
Recall value	0.96
Precision value	0.894
Precision-Recall value	0.824

Performance Metrics

Fig. 12. Tabular form of performance metrics for YOLO detection model which was trained on 10 different classes where data in MsCOCO annotated.

bottom-right points as (x_min,y_min) and (x_max,y_max). [14] Along with bounding boxes, the confidence score is also generated, which signifies the likelihood of an object. The confidence score is calculated as the product of the probability of the object present in the bounding box and IoU between the predicted bounding box and ground truth.

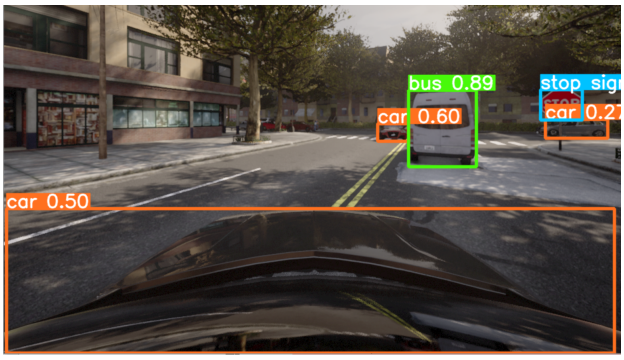

Fig. 13. How the YOLO model is detecting multiple objects in the Carla simulator.

Non-Max suppression: There is a possibility, that the same object can be in the same place for a period of time and to avoid duplicate or multiple bounding boxes Non-maximum suppression is used [13]. This technique removes overlapping bounding boxes resulting in the final output of the detection model.

For the visual aid of the output, we can employ OpenCV packages. The bounding box coordinates which were predicted frame by frame are overlaid on the original frame, showing how well an object is identified. Along with these confidence scores and label of the object is also populated.

D. Deep SORT customization

Deep Sort is an open source code that is present under general license [8]. The Original code was forked from GitHub by Wojke and 4 other members. There are changes made to the original code so that it can be made simple and compatible with the CARLA simulator. Below are the few modifications that were performed on the original Deep SORT repository.

- The actual deep sort expects the input in xywh format i,e. (x,y) centroid position along with the width and height of the bounding box from the detection model [25] [8]. Since YOLOv8 generates output in xyxy format, Deep SORT was modified to accept xyxy format rather than xywh format
- Deep SORT does not take class id's or class labels during its tracking process, as a result, the tracked output does not contain any label-based classification between different objects [25] [8]. The tracker file was modified to generate the class label as well when overlaying the tracked bounding box details.
- Since the original Deep SORT was designed to validate and run on video sequence there are several utility classes that were modified and removed as per requirements
- we are utilizing a pre-trained version of Deep SORT, adeptly omitting the train and test classes were removed, and all the methods invocation related to this process. [25] [8] The reduction in method calls is an appreciable gain in the tome complexity

E. Deep SORT Implementation

The algorithm encompasses several well-defined steps, each instrumental in achieving seamless object tracking:

A) The object detection from YOLO is used to extract required details like bounding boxes in xyxy format along with confidence score and the frame information. [25] [8]

B) The tracker file orchestrates the fusion of integrating the detection model, Carla simulation, and tracking model.

C) Later the tracking model is used to track the objects in the frame and assign a tracking id for the object along with new bounding box details. A more comprehensive dissection of this step follows:

1) Leveraging a Kalman filter, tracked objects with initial bounding boxes undergo state estimation encompassing position, velocity, and covariance. [21]
2) Non-max suppression is executed during every frame analysis that is obtained from the detection model. This step curtails redundant or low-confident detections optimizing the aver all tracking position.
3) A cosine matrix which is a precursor to cascade matching is generated based on the IoU threshold. This acts as the first stage of data association.
4) The Hungarian algorithm [25] [8] tackles the linear assignment problem, helping to identify and eliminate matched detections, unmatched detections, and unmatched tracks. This is a repetitive process that compares consecutive frames to find the matching pair of detections and tracks.
5) Once a pair is recognized a tracking ID is assigned to that track and is monitored and the state of an object is updated using the Kalman filter. [25] [8] This back-propagation will allow adjusting of the tracking score of the object allowing an increase in accuracy.
6) All the tracks that were devoid of any meaningful associations over a stipulated frame interval are pruned.

VI. Results

A. YOLO Training outcomes

Since YOLO mode is run on the custom dataset to understand how well our model understood the dataset and classified the classes can be understood using some performance metrics.

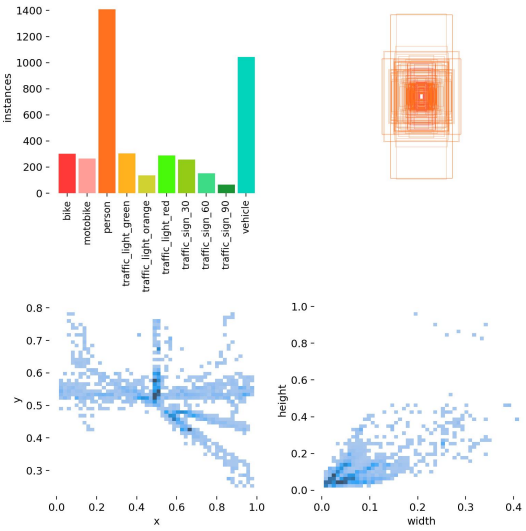

Fig. 14. Different class labels of the dataset.

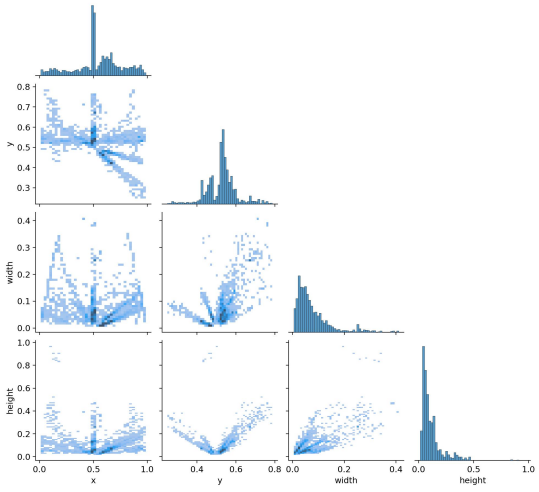

Fig. 15. Different labels correlogram from the dataset.

Fig 13 and Fig 14, show the visual representation of labels on which the model is trained and the corresponding relationships between different classes in the dataset. This helps in understanding the frequency at which different classes are predicted together.

A) Confusion matrix: Both normalized and non-normalized confusion matrices illustrated how well the model can predict the classes. While the diagonal values of the matrix show how well the model identifies that specific class. The normalized

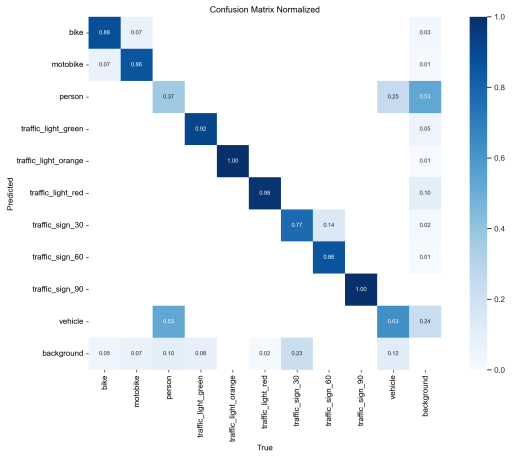

Fig. 16. Normalized confusion matrix generated from detection model.

matrix has all the values reduced to 0-1, where 1 indicates 100% true prediction. From the matrix, we can see that the bike with 0.95, the vehicle with 0.93, and the moto-bike with 0.89 have the highest true positives.

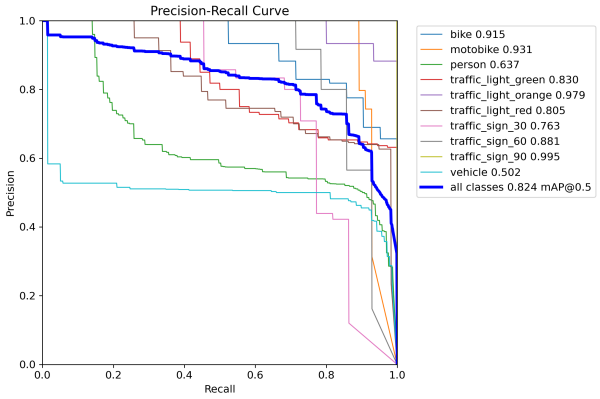

Fig. 17. Precision-recall curve

B) PR Curve: The precision-recall curve is used to evaluate the performance of the model. It plots the precision (fraction of true positive detection among all positive detections) against recall (fraction of true positive among all objects) at different thresholds. A higher area under the PR curve (AUC) indicates a better performance model. From Fig.16., we can see the PR curve values at 0.965 for all the classes.

C) F1 confidence curve: F1 score is a measure of the trade-off between precision and recall, a higher F1 score indicates, better performance. From the Fig.17, we can see that the F1 score is 0.92 at a threshold of 0.36

D) Train-loss results: These show how the training loss changes over multiple epochs of training. It helps in understanding how the model is converging and how the training process is going.

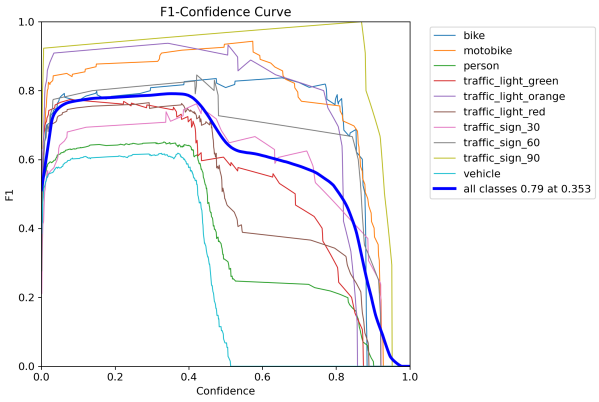

Fig. 18. F1 confidence curve

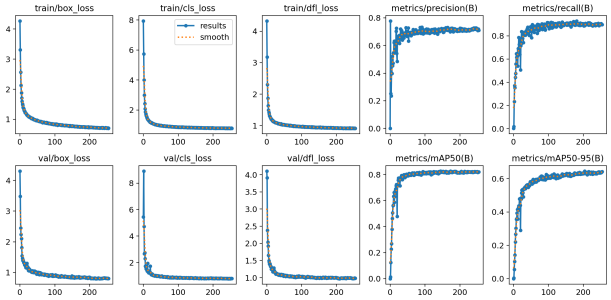

Fig. 19. Train-loss curves

B. Deep SORT Tracking model outcome

After using the detection weights, and integrating them with Carla below images show how the Deep sort model detected multiple objects. The deep SORT model is tested in normal and complex weather simulations in Carla.

Fig. 20. Deep sort vehicle tracking in normal conditions

C. Tracking system Evaluations

For any MOT tracking system, the two main performance metrics are MOTA and MOTP which are Multi-object Tracking Accuracy and Multi-object Tracking Precision. MOTA measures the overall tracking accuracy by taking into account three types of errors false positives, misses, and mismatches when comparing the ground truth from Carla to tracked predictions. On the other hand, MOTP measures the localization accuracy of the tracking algorithm by calculating the average overlap between ground truth and tracked predictions. Higher

Fig. 21. Deep sort vehicle tracking in high fog weather

values of MOTA and MOTP indicate better the performance of model.

$$\text{MOTA} = 1 - \frac{\sum_t (FN_t + FP_t + IDS_t)}{\sum_t N(GT)_t},$$

$$\text{MOTP} = 1 - \frac{\sum_t \text{IOU}(GT_t^i, H_t^i)}{\sum_t M_t},$$

Fig. 22. Mathematical formulation of MOTA and MOTP

MOTA and MOTP can be calculated using the above mathematical formulas directly or by using open packages like PyMOT or mot metrics. For this project, we are using the script developed from the inspiration of both the PYMOT and mot metrics packages to generate the values of MOTA and MOTP. The main requirement is to have accumulated information on bounding box details along with time stamps, frame to frame for both ground truth and the tracking model. Then the IoU threshold is calculated along with total misses and matches. All these values can be substituted to calculate MOTA and MOTP. For different runs below are the values shown.

Performance Metrics of tracking system

RUNS	MOTA	MOTP
200	0.27	0.18
450	0.47	0.23

Fig. 23. MOTA, MOTP predicted values of the tracking system. These values tend to change depending on the position and number of vehicles that spawn in the Carla environment

VII. Discussion

From the above results, we can see that the calculated accuracy and precision of the tracking system are on par with the benchmark problems. We have identified a few reasons for these.

A. Ground Truth generation complexity

While determining the Ground truth from the Carla, encompasses all the available objects including those that are occluded. However, the reason for not excluding the occluded

objects due to the requirement of advanced technology like a stereo camera or LIDAR to calculate the depth perception and subsequently project it to generate the ground truth. This approach is omitted due to its complexity and significant computational demands.

B. Choice of object detection model

In MOT object detection is the initial step. The object detection model relies on the YOLO model which is a commercial model and is known for with high speed, but the benchmark solutions often use a two-stage model like Faster R-CNN, used for academic purposes which outsmarts YOLO in terms of accuracy. Despite its relative trade-offs in accuracy the selection of YOLO is justified by its processing speed.

C. IoU threshold

The IoU threshold value for the YOLO model can not be customized and the low threshold value can result in high false positives affecting the overall accuracy of the tracking model.

D. Limited frame count

Evaluating any tracking model involves assessing its performance across a huge range of frames. The system operates with some restrictions, we are running for around 400 frames to evaluate the model which is very limited when compared to benchmark evaluations(For a benchmark problem researchers use 5000+ frames.

E. Parameter tuning of Deep Sort

Deep sort has several input parameters that can be customized like max_age, nn_budget, NMS_max_overlap, IoU_distance, and a few more. For this project, all the values were set as default. Depending on the tracking model output, we can hyper-tune these values to increase the efficiency of the model For instance, reduced NMS overlap values can impact and reduce false positives. Due to these reasons, the number of false positives increases impacting the accuracy and precision of the overall tracking system.

VIII. Future Scope

All the limitations of our project can be overcome by certain changes, like using better detection models like Faster R-CNN, DERT, or EffifientDET that have produced better object detection.

In real-time autonomous vehicles are equipped with several sensors that work simultaneously to improve and provide a greater understanding of the surrounding. Expanding the sensor suit beyond the monocular camera can greatly improve object tracking. Utilizing more sensors like stereo cameras, LIDAR, GPS, and radar can help the overall tracking system. This leverage multi-modal tracking where the system fuses several sensors and overcomes challenges of lighting, and weather conditions as well.

Multi-objective tracking systems are an ongoing developing sector, Every day new tracking methods are proposed. Use of the latest and updated tracking models like Transformer based, or hybrid learning models along with edge computing which reduces the latency can increase the capabilities of autonomous vehicles.

IX. Conclusion

In conclusion, this project shows a stride in the domain of 2D real-time multi-vehicle tracking. By integrating sophisticated deep learning technologies such as Deep SORT and YOLOv8, a comprehensive solution for a dynamic environment has been developed with the potential to construct intelligent systems capable of real-time object analysis. The 47% MOTA shows stability and improvement. of performance with the change is the tracking environment. The efficiency of the implementation of these techniques accentuates their effectiveness in various domains. The seamless integration of detection and tracking solutions can revolutionize the interactions of autonomous vehicles with their surroundings, resulting in safer transportation.

Broadening the scope of the project with advanced technologies like sensor fusion, and updated tracking algorithms proves the adaptability allowing to handle complex environments. With all these enhancements the existing boundaries of the multi-object tracking system can be raised to higher levels leading to more reliable autonomous vehicles.

In essence, this project exemplifies the significant progress achieved through the fusion of advanced computer vision techniques and innovative algorithms. As technology continues to grow, the insights gained from this project provide a solid foundation for further exploration in the realm of object detection and tracking.

X. Declaration

Declaration of Originality. I am aware of and understand the University of Exeter's policy on plagiarism and I certify that this assignment is my own work, except where indicated by reference, and that I have followed good academic practices.

Declaration of Ethical Concerns. This work does not raise any ethical issues. No human or animal subjects are involved nor has the personal data of human subjects been processed. Additionally, no security or safety-critical activities have been carried out.

References

[1] R. Hussain and S. Zeadally, "Autonomous cars: Research results, issues, and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1275–1313, 2018.

[2] Y. Xu, Y. Ban, X. Alameda-Pineda, and R. Horaud, "Deepmot: A differentiable framework for training multiple object trackers," *arXiv preprint arXiv:1906.06618*, vol. 10, no. 11, 2019.

[3] D. Sudha and J. Priyadarshini, "An intelligent multiple vehicle detection and tracking using modified vibe algorithm and deep learning algorithm," *Soft Computing*, vol. 24, pp. 17 417–17 429, 2020.

[4] R. Ravindran, M. J. Santora, and M. M. Jamali, "Multi-object detection and tracking, based on dnn, for autonomous vehicles: A review," *IEEE Sensors Journal*, vol. 21, no. 5, pp. 5668–5677, 2020.

[5] S. Guo, S. Wang, Z. Yang, L. Wang, H. Zhang, P. Guo, Y. Gao, and J. Guo, "A review of deep learning-based visual multi-object tracking algorithms for autonomous driving," *Applied Sciences*, vol. 12, no. 21, p. 10741, 2022.

[6] D. Niranjan, B. VinayKarthik *et al.*, "Deep learning based object detection model for autonomous driving research using carla simulator," in *2021 2nd international conference on smart electronics and communication (ICOSEC)*. IEEE, 2021, pp. 1251–1258.
[7] J. Terven and D. Cordova-Esparza, "A comprehensive review of yolo: From yolov1 to yolov8 and beyond," *arXiv preprint arXiv:2304.00501*, 2023.
[8] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*. IEEE, 2017, pp. 3645–3649.
[9] O. Dokur and S. Katkoori, "Carla connect: A connected autonomous vehicle (cav) driving simulator," in *2022 IEEE International Symposium on Smart Electronic Systems (iSES)*. IEEE, 2022, pp. 656–659.
[10] J. Wu, D. Yin, J. Chen, Y. Wu, H. Si, and K. Lin, "A survey on monocular 3d object detection algorithms based on deep learning," in *Journal of Physics: Conference Series*, vol. 1518, no. 1. IOP Publishing, 2020, p. 012049.
[11] D. Fernandes, A. Silva, R. Névoa, C. Simões, D. Gonzalez, M. Guevara, P. Novais, J. Monteiro, and P. Melo-Pinto, "Point-cloud based 3d object detection and classification methods for self-driving applications: A survey and taxonomy," *Information Fusion*, vol. 68, pp. 161–191, 2021.
[12] D. Cazzato, C. Cimarelli, J. L. Sanchez-Lopez, H. Voos, and M. Leo, "A survey of computer vision methods for 2d object detection from unmanned aerial vehicles," *Journal of Imaging*, vol. 6, no. 8, p. 78, 2020.
[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.
[14] Ultralytics, "Ultralytics documentation," https://docs.ultralytics.com.
[15] M. Hussain, "Yolo-v1 to yolo-v8, the rise of yolo and its complementary nature toward digital manufacturing and industrial defect detection," *Machines*, vol. 11, no. 7, p. 677, 2023.
[16] A. Mendhe, H. Chaudhari, A. Diwan, S. Rathod, and A. Sharma, "Object detection and tracking for autonomous vehicle using ai in carla," in *2022 International Conference on Industry 4.0 Technology (I4Tech)*. IEEE, 2022, pp. 1–5.
[17] X. Weng, J. Wang, D. Held, and K. Kitani, "3d multi-object tracking: A baseline and new evaluation metrics," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 359–10 366.
[18] 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," *Applied Sciences*, vol. 12, no. 3, p. 1319, 2022.
[19] H. Wang and B. Liu, "Detection and tracking dynamic vehicles for autonomous driving based on 2-d point scans," *IEEE Systems Journal*, 2022.
[20] Y. Du, Z. Zhao, Y. Song, Y. Zhao, F. Su, T. Gong, and H. Meng, "Strongsort: Make deepsort great again," *IEEE Transactions on Multimedia*, 2023.
[21] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.
[22] A. B. Amjoud and M. Amrouch, "Object detection using deep learning, cnns and vision transformers: A review," *IEEE Access*, 2023.
[23] dataset, "carla dataset link," https://www.kaggle.com/datasets/alechantson/carla
[24] carla docs, "carla docs - bounding box," https://carla.readthedocs.io/en/latest/tuto$_{G}b ounding_{b}oxes/bounding-boxes/$.
[25] Y. Park, L. M. Dang, S. Lee, D. Han, and H. Moon, "Multiple object tracking in deep learning approaches: A survey," *Electronics*, vol. 10, no. 19, p. 2406, 2021.

XI. APPENDIX

The additional plots from the YOLO training model were added in this section

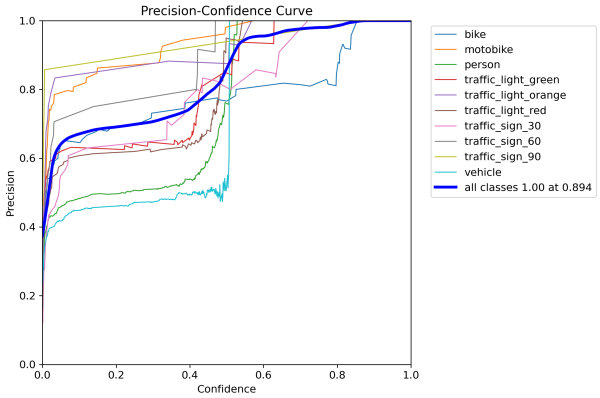

Fig. 24. Precision curve

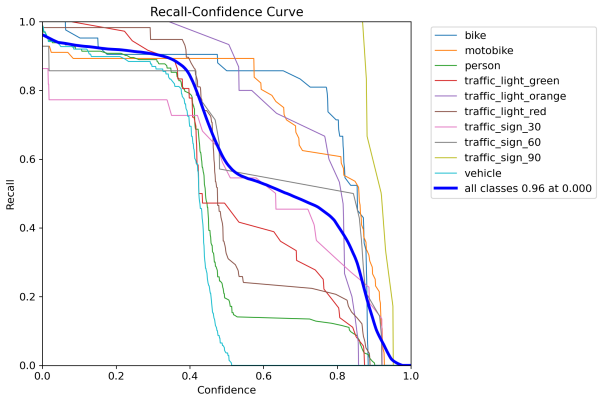

Fig. 25. Precision-recall curve

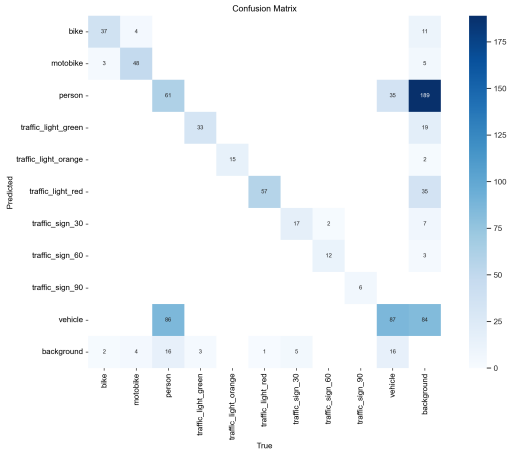

Fig. 26. Precision-recall curve