DEGREE PROJECT IN TECHNOLOGY,
SPECIALIZING IN SYSTEMS, CONTROL AND ROBOTICS
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2020

Evaluation and Analysis of Perception Systems for Autonomous Driving

DEVENDRA SHARMA

KTH ROYAL INSTITUTE OF TECHNOLOGY
ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Author

Devendra Sharma <devendra@kth.se>
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology

Place for Project

EZRMS department
MAN Truck & Bus SE
Munich, Germany

Examiner

György Dán
Division of Network and Systems Engineering
School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology

Internal Supervisor

Ezzeldin Zaki
Division of Network and Systems Engineering
KTH Royal Institute of Technology

External Supervisor

Gustav Åberg
Engineer - EZRMS department
MAN Truck & Bus SE

Arne Hildebrandt
Engineer - EZRET department
MAN Truck & Bus SE

Abstract

For safe mobility, an autonomous vehicle must perceive the surroundings accurately. There are many perception tasks associated with understanding the local environment such as object detection, localization, and lane analysis. Object detection, in particular, plays a vital role in determining an object's location and classifying it correctly and is one of the challenging tasks in the self-driving research area. Before employing an object detection module in autonomous vehicle testing, an organization needs to have a precise analysis of the module. Hence, it becomes crucial for a company to have an evaluation framework to evaluate an object detection algorithm's performance. This thesis develops a comprehensive framework for evaluating and analyzing object detection algorithms, both 2D (camera images based) and 3D (LiDAR point cloud-based). The pipeline developed in this thesis provides the ability to evaluate multiple models with ease, signified by the key performance metrics, Average Precision, F-score, and Mean Average Precision. 40-point interpolation method is used to calculate the Average Precision.

Keywords

Object detection, CARLA, evaluation, simulator, autonomous vehicles, KITTI, LiDAR, stereo camera.

Sammanfattning

För säker rörlighet måste ett autonomt fordon uppfatta omgivningen exakt. Det finns många uppfattningsuppgifter associerade med att förstå den lokala miljön, såsom objektdetektering, lokalisering och filanalys. I synnerhet objektdetektering spelar en viktig roll för att bestämma ett objekts plats och klassificera det korrekt och är en av de utmanande uppgifterna inom det självdrivande forskningsområdet. Innan en anställd detekteringsmodul används i autonoma fordonsprovningar måste en organisation ha en exakt analys av modulen. Därför blir det avgörande för ett företag att ha en utvärderingsram för att utvärdera en objektdetekteringsalgoritms prestanda. Denna avhandling utvecklar ett omfattande ramverk för utvärdering och analys av objektdetekteringsalgoritmer, både 2 D (kamerabilder baserade) och 3 D (LiDAR-punktmolnbaserade). Rörledningen som utvecklats i denna avhandling ger möjlighet att enkelt utvärdera flera modeller, betecknad med nyckelprestandamätvärdena, Genomsnittlig precision, F-poäng och genomsnittlig genomsnittlig precision. 40-punkts interpoleringsmetod används för att beräkna medelprecisionen.

Nyckelord

Objektdetektering, CARLA, utvärdering, simulator, autonoma fordon, KITTI, LiDAR, stereokamera

Acknowledgements

The path to academic and intellectual accomplishment is challenging but also exciting. I feel fortunate to attend the Master's Programme in Systems, Control & Robotics at KTH. I consider attending this Programme the most inspiring and vital experience for me.

Firstly, I would like to thank MAN Truck & Bus SE for giving me the excellent opportunity of doing my thesis with them. Special thanks to my supervisors at MAN, Gustav Åberg and Arne Hildebrandt for their immense support and advice throughout the project. Additionally, great thanks to my academic supervisor at KTH, Ezzeldin Zaki for his invaluable input during the thesis writing. Many thanks to my examiner György Dán for being the most understanding mentor and guiding me throughout my thesis. Finally, a big thanks to my family and friends for their continuous support and helping me stay positive during the whole process.

Contents

Chapter 1

Introduction

For a long time, vehicles were only thought of as a traditional mechanical enterprise until the novel fusion of software and hardware remodeled the automotive industry's vision and devised the philosophy of autonomous driving. Autonomous driving commonly refers to the research area where an autonomous vehicle tries to operate with minimal human assistance. An autonomous vehicle reduces the need for driver's engagement by performing intelligent operations such as collision avoidance, lane analysis, and traffic sign detection [1].

Key technologies associated with autonomous vehicles are perception, localization, planning & decision-making, and control, shown in Figure 1.0.1 [2]. The perception module administers sensing of the surroundings and provides the model of the local environment. Localization is responsible for precisely estimating the vehicle's pose in the map, planning and decision-making decides how the vehicle will move through the surroundings. Finally, control, the concluding move of the vehicle, deals with the system's actuation such as steering, break, and acceleration.

Vital advancements in the last decade have considerably brought the elevation in the self-driving industry and thus increase the demand for more reliable and safe perception systems. In this thesis, we focus on perception in autonomous driving and analyze the performance of the perception modules.

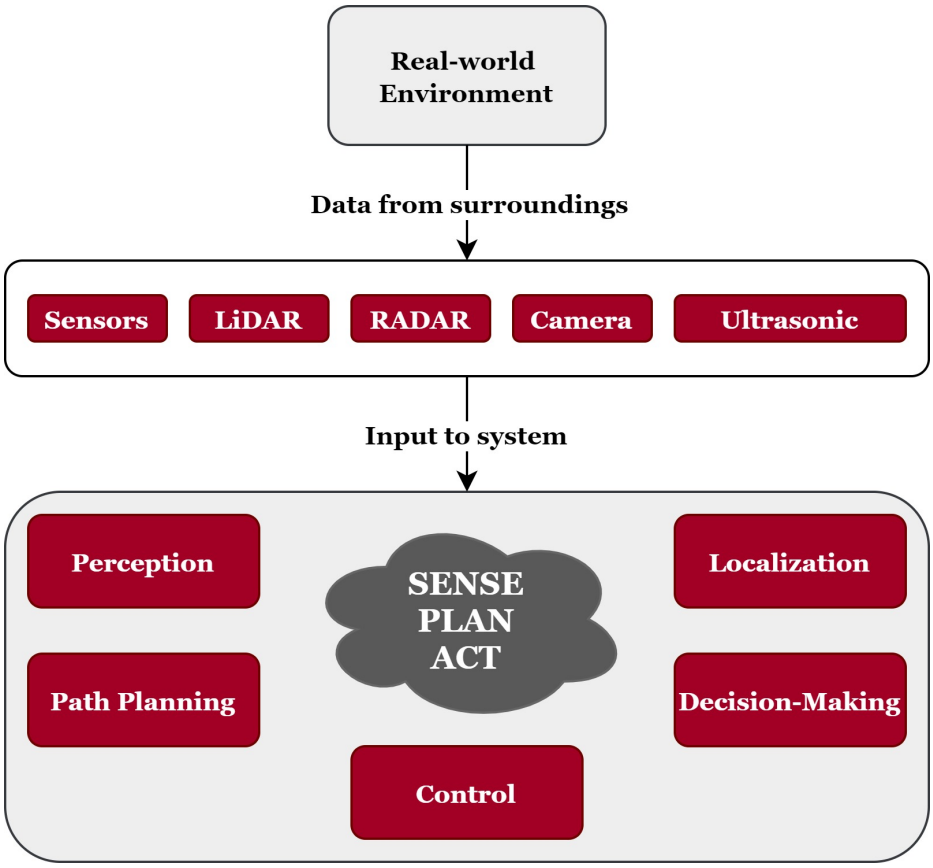

Figure 1.0.1: Framework of Autonomous Vehicle

1.1 Background

The first crucial step for an autonomous vehicle is awareness of the local environment. It is of utmost importance for safe mobility that the vehicle knows its surroundings and identifies different objects such as pedestrian, vehicles, etc. There are various perception tasks, such as object detection, lane analysis, and self-positioning. An autonomous vehicle uses multiple sensors, e.g., a camera, Light Detection and Ranging (LiDAR), and Radio Detection and Ranging (RADAR) to detect possible objects around it.

LiDAR emits thousands of light pulses every second and creates a 3D image of the surrounding with more accurate depth information as compared to cameras [3]. However, cameras play a vital role in scenic understanding compared to LiDARs, such as better performance in poor weather conditions, deciphering colors, and interpreting traffic signs. Though, the performance of

the camera gets poor in during low-light.

Determining the object's location is still a challenging task in the self-driving research area, making real-time object detection crucial for autonomous driving. It allows the vehicle to analyze the object type, separate it from the surrounding, and correctly locate its position in the image [4]. In computer vision, the most traditional way to localize an object in the surroundings is to represent the object's location by drawing a bounding box around it.

Recently, numerous approaches have been applied to solve the increasing demand for precise object detection. There are multiple perception algorithms for object detection, such as YOLO (You Only Look Once), Fast R-CNN (Region-Based Convolutional Neural Networks), Faster R-CNN, and SSD (Single Shot MultiBox Detector), providing fast computation and accurate results. Although various 2D and 3D object detection algorithms [5][6] have been introduced with time, still not all can achieve the ideal performance, i.e., able to distinguish distinct objects in the environment accurately. One of the significant problems for a robust object detection algorithm's problem is tough to address as self-driving cars have to operate in different weather conditions with varied visibility on the road.

This high-performance requirement expects a minimal margin of error on the performance of such object detection algorithms. Since it is vital to have a robust perception algorithm to achieve precise object detection, it is inevitable to evaluate such a perception system's performance and analyze its performance.

An object detection algorithm's performance can be determined by calculating the degree of area matching between the model's detections and the ground truth. For an Original Equipment Manufacturer (OEM), instead of using open source models or developing a new object detection module from scratch, it prefers to buy it from the supplier. Hence, they need to know how to evaluate object detection algorithms and how they can meticulously compare two different object detection algorithms. However, the need for evaluation

framework for an OEM is also to assess the in-house algorithms.

1.2 Objective

With the prominent use of object detection algorithms in self-driving vehicles, it becomes essential to evaluate an algorithm's performance. It's a reasonable question to ask whether we can rely on a single algorithm or not and how we can make a definite comparison between different modules. To answer such a problem, we need an empirical evaluation of algorithms based on quantitative metrics.

As both 2D and 3D object detection are of paramount importance, this thesis will evaluate the perception algorithm for each of them on selected metrics.

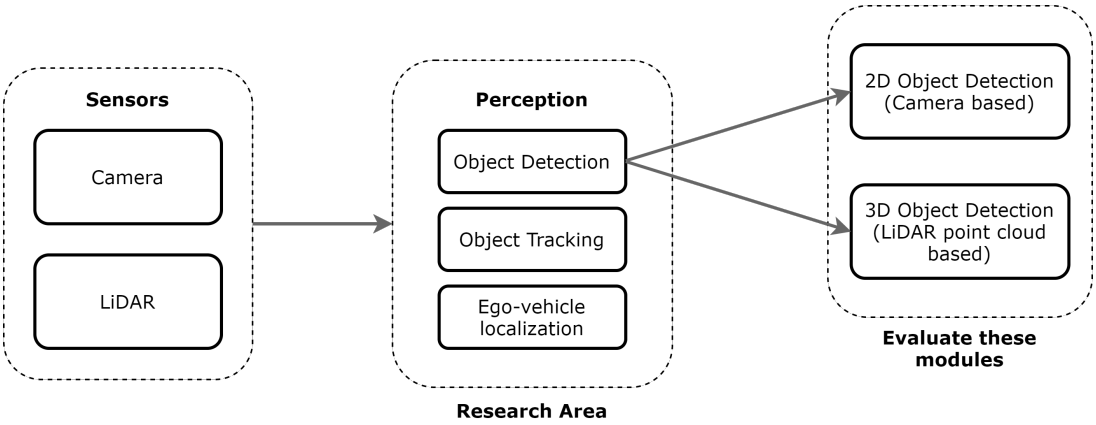

Figure 1.2.1: Research Area

This thesis will attempt to answer the following questions:

- *What could be the possible metrics for evaluating performance of object detection algorithms in the context of autonomous driving?*
- *What is the possible end-to-end framework to evaluate and analyse multiple 2D and 3D object detection algorithms?*

1.3 Ethical considerations

Object detection represents a crucial role in autonomous vehicles' safety, as even a small margin of error, e.g., false detection, can lead to a severe accident. Such points lead us to think, "Are self-driving vehicles safe?" there have been many debates and conferences over this perplexity.

Self-driving vehicles are expected to have a positive impact on road accidents as well as the environment. According to a study, around 94% of US road accidents occur due to human error [7]. On the other hand, Google claims that its driverless car has covered over 700,000 miles without a recorded accident [8]. There are many cases of human errors such as drink and drive, emotional aggression, or haste; such risks can be reduced with self-driving cars, which could lead to fewer fatalities on the road.

1.4 Sustainability

Self-driving cars are more efficient in using breaks and acceleration, thus reducing congestion on roads, leading to better fuel efficiency and reduced emissions. Congestion leads to wasted fuel, estimated at 3.1 billion gallons in 2014 in the US [9]. Therefore, autonomous vehicles can be beneficial in reducing the current rise in traffic pollution.

1.5 Delimitations

The thesis will only focus on the evaluation of single sensor-based perception modules, i.e., camera-based object detection algorithms and object detection algorithm based on point cloud from LiDAR. Due to time limitation, sensor-fusion[1] based evaluation is not evaluated.

[1]fusing images from camera and point cloud from LiDAR for 3D object detection

1.6 Outline

The report is divided into five chapters. Chapter 2 starts with introducing related work on algorithm's evaluation and forms a foundation for the study, along with the relevant theoretical concepts related to the thesis. Chapter 3 describes the methods that have been implemented for the evaluation of object detection algorithms and on the selection of suitable metrics for performance analysis along with the methods used to generate the dataset. Chapter 4 discusses the evaluation results . Finally, the conclusion and future work of the thesis work are discussed in the final chapter.

Chapter 2

Background Theory & Related Work

This chapter covers the theoretical concepts within the research area of the thesis. Since the aim is to evaluate 2D & 3D object detection algorithms' performance for autonomous vehicles, the chapter begins with a literature behind 2D & 3D object detection , understanding the fundamental difference between them. It is vital to know how these algorithms process the data and in what form the output is generated. Later, the theoretical concepts behind evaluation metrics for performance analysis will be discussed.

2.1 Object detection

Object Detection is one of the principal computer vision problems, it is widely prevalent among engineers and researchers due to its applications in the fields such as image classification, face recognition, and autonomous driving. In simpler words, object detection can be broken down into two straightforward tasks: object localization and object classification. Object localization is locating an object in the given image (in image frame) or point cloud (in world frame). A given image is simply an RGB (Red Green Blue) image, and a point cloud is a set of data points (3D points in space) obtained from scanning the

environment. Object classification provides us with the detected object class like a car, pedestrian, or motorbike. An object detection module takes an image or LiDAR point cloud as an input containing one or more objects and provides coordinates [1] of bounding boxes enclosing the detected objects as closely as possible, along with the object's class (type).

Besides, it gives each box a confidence score expressing the network's confidence for this bounding box containing the object. Further, object detection can be classified into 2D and 3D object detection based on the usage and sensors i.e. camera and LiDAR.

2.1.1 2D object detection

This section describes the theoretical concepts behind 2D object detection.

A 2D object detection model takes an RGB image as an input and outputs an image with bounding boxes, class labels for each detected object, and confidence the algorithm has on the detected object, i.e., confidence score. In 2D object detection, the bounding box drawn is represented by a rectangle with enclosing the detected object as precisely as possible. An example of a 2D bounding box ground truth is shown in the figure 2.1.1

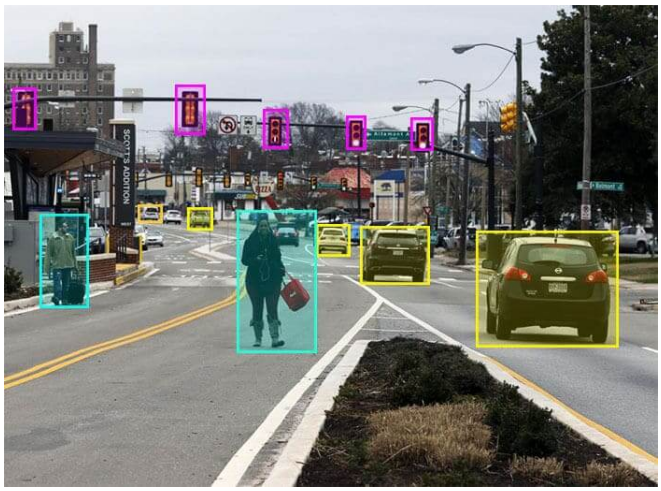

Figure 2.1.1: Example of a 2D bounding box [10]

[1] in the respective reference frame

The algorithms for object detection can be divided into two parts based on their action for a particular task, i.e., the algorithms which perform the detection in a single-stage, for example, SSD (Single Shot MultiBox Detector) and YOLO (You Only Look Once) are called single-stage architecture based algorithms. The ones who perform the detection in two stages, e.g., Faster-RCNN (Region-based Convolutional Neural Networks) [11] is called two-stage architecture based algorithms. In two-stage design, the first stage is the generation of region proposals, i.e., extracting features from the given input image, and the second layer work on the features extracted.

The two-stage architecture based algorithm was proposed by Girshick *et al.* in [12], where the authors have presented a basic R-CNN architecture to detect object region proposals based on the selective search method. These detected region proposals contain the objects of interest while filtering out most of the background region. Then these candidate regions are further refined to give the specific object class.

With time, continuous improvements have been done on R-CNN architecture, i.e., R-CNN, Fast R-CNN, and Faster R-CNN. With the evolution, the essential difference in all these versions is the improved computational efficiency with enhanced performance in terms of correct detections. Fast R-CNN [13] has a less complicated architecture as compared to R-CNN, can be seen in figure 2.1.2. However, the problem of dependency on external methods to extract the region proposals exists in Fast R-CNN, and an improved version is introduced by Faster R-CNN Ren *et al.* in [11]. This bottleneck issue has been solved in Faster R-CNN, i.e., for region proposal, it employs RPN (Regional Proposal Network) instead of selective search. Additionally, Lin *et al.* in [14] has proposed certain modifications on Faster R-CNN, enabling better detection performance on the COCO detection dataset [15], and increased usage for object detection. For the thesis, the main objective is not to train the models, instead create a pipeline for the evaluation. Therefore pre-trained basic models were used and ignored additional modifications done on the model, as introduced by Lin *et al.*[14]

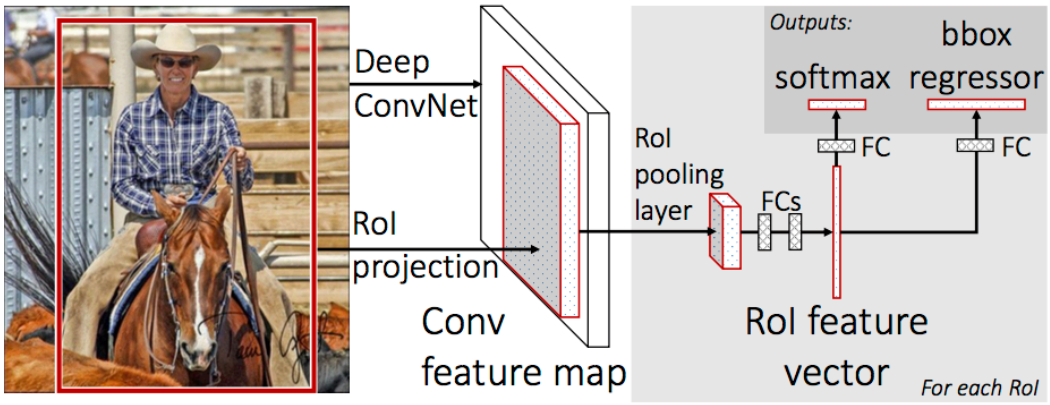

Figure 2.1.2: Fast R-CNN Architecture [13]

The models explained above are the examples of two-stage detectors, which generate region proposals in the first stage and perform classifying and refining in the second stage. In contrast, the single-stage object detectors for e.g. YOLO (You Only Look Once) perform the given task in the single-stage introduced by Redmon *et al.* in [5].

YOLO is a single convolutional neural network that performs the detection task in the single evaluation and predicts bounding boxes and class probabilities from the given input image, as shown in figure 2.1.3. Since the detection is done in a single stage, i.e., the classification and bounding box regression simultaneously. The architecture is less complex, and comparatively faster than two-stage detectors and can be optimized end-to-end for the desired performance. YOLO learns generalized representations of the objects; hence, when unexpected inputs are applied, there is very little probability that it will break down.

YOLO filters out the bounding boxes based on IoU (Intersection Over Union is a metric for comparison between ground truth bounding box and predicted bounding box, explained in detail in section 2.3) less than the threshold level. This threshold level is set by the user as per the usage, in general, the threshold value is 0.5. However, the initial YOLO version struggled with localizing small objects in the image. This issue is rectified by further releases of YOLO [16] [17].

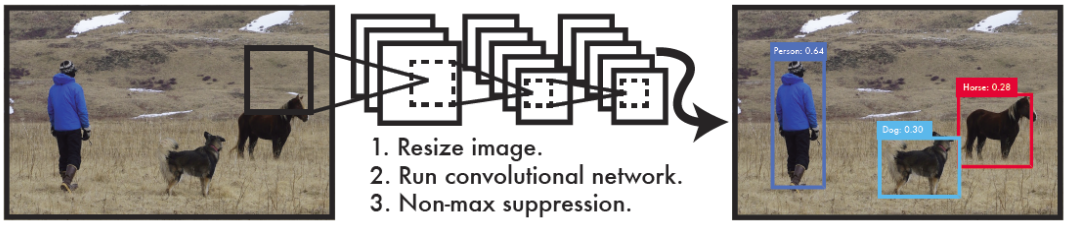

Figure 2.1.3: YOLO detection system [5]

The issue of the high computational requirement to run the object detector is one of the major obstacles when the system runs in real-time. In real-time, object detectors have high computational requirements with many GPUs, which is not economical. Therefore, Bochkovskiy *et al.* addressed this problem in [18] and introduced a more accurate and real-time performance efficient object detector YOLOv4, resulting in improved speed and accuracy.

2.1.2 3D object detection

This section describes the theoretical concepts behind 3D object detection.

A 3D object detection model can take multiple inputs from different sensors, e.g., LiDAR point cloud from LiDAR and RGB image from the camera. In this thesis, the evaluation is done for the model using LiDAR data only. Be it a LiDAR point cloud or an RGB image, the model's output is a 3D bounding box enclosing the detected object along with the class label of the detected object, e.g., car, pedestrian, or motorbike.

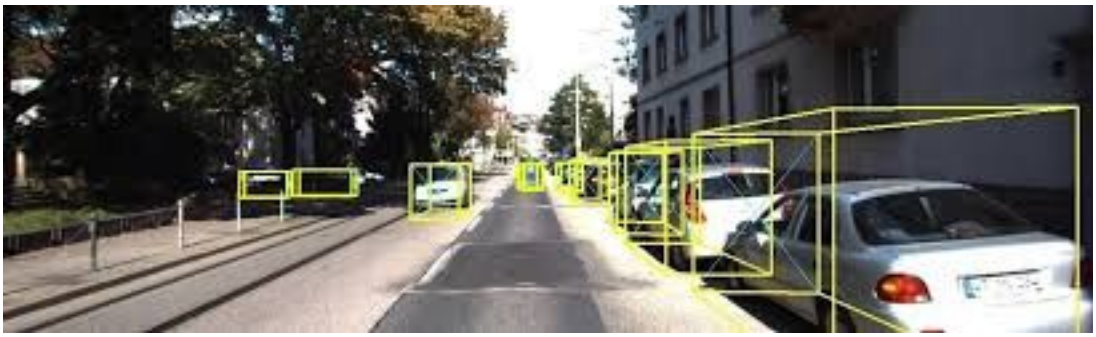

Figure 2.1.4: Example of a 3D Bounding Box [19]

Engelcke *et al.* proposed an approach in [20] for 3D object detection only on LiDAR point cloud using CNNs (convolutional neural networks). The author described the application of CNNs to predict detection scores from a sparse 3D grid given as input. The input LiDAR point cloud is discretized into a sparse 3D grid. Based on the statistics of the points in the non-empty cell, a feature vector is extracted, i.e., for the section that contains a non-zero number of points. And for these non-empty cells, the points are processed by implementing sparse 3D convolutional to this non-empty grid. Sliding-window search is employed with a fixed size window of N varied orientations to detect the objects. For evaluation, the author used the publicly available dataset [21] and projected 3D detections into the 2D image plane. This projection is made using calibration data, and the detections that fall out of the image are discarded. The evaluation is done on the projected 2D detections, making the assessment comparatively easier.

Another improved LiDAR-based model, ”VoxelNet,” was presented by Zhou *et al.* in [22], developed by Apple, as shown in figure 2.1.5. The author proposed a deep end-to-end network that conjoins feature extraction and bounding box prediction into a single stage, and the need for manual feature engineering for 3D point clouds is removed. The point cloud is divided into equally spaced 3D voxels (single data point on a three dimensional grid), and within each voxel, the points are grouped. VoxelNet transforms these groups of points utilizing the newly introduced VFE (voxel feature encoding) layer into a unified feature representation. Te VFE layer enables inter-point interaction within a voxel, combining point-wise features. The output from this layer, i.e., a sparse 4D array, is then fed through 3D convolutional layers and later reshaped into a 3D feature map. Finally, an RPN (Region proposal network) uses the volumetric representation and generates the detection results. The authors evaluated the model on the KITTI dataset [21] for both bird’s eye view detection and the full 3D detection.

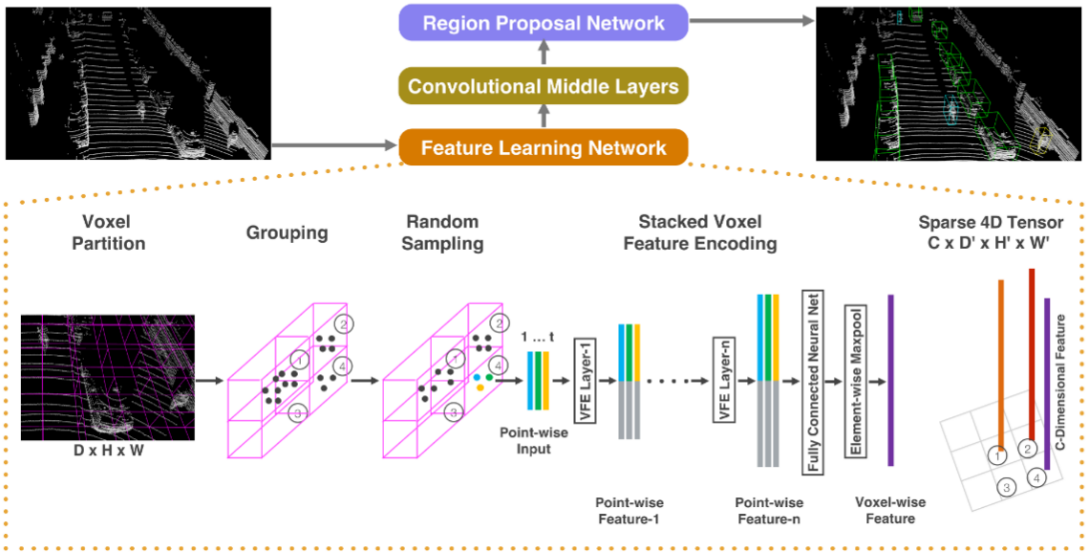

Figure 2.1.5: VoxelNet Architecture. The network takes raw point cloud as input and generates 3D detection [22]

"Complex-YOLO," a LiDAR-based 3D object detection, is proposed by Simon *et al.* in [6]. Complex-YOLO is a well-structured model that directly works on the LiDAR point cloud to estimate and localize 3D bounding boxes for multiple objects. The model needs no RGB image as an input; it is LiDAR-based only. The Complex-YOLO is accurate and capable of having a better computational speed. For point-cloud preprocessing and feature extraction, a multi-view idea (MV3D) [23] is used. In the model, multi-view fusion is neglected, and a single birds-eye-view[2] RGB map is generated, refer figure 2.1.6. The model is based on YOLOv2 [16], one of the advanced object detector version of YOLO. The model uses a new E-RPN to estimate the orientation of the objects coded by imaginary and real part for each bounding box. The model is capable of predicting the exact heading of the objects in real-time, ensuring accurate localization. For the model, special anchor-boxes are designed to capture the object, which is based on only a few points, e.g., pedestrians. Complex-YOLO is evaluated on the KITTI benchmark suite [21] and can predict all classes mentioned in the KITTI dataset.

[2]aerial view of the world

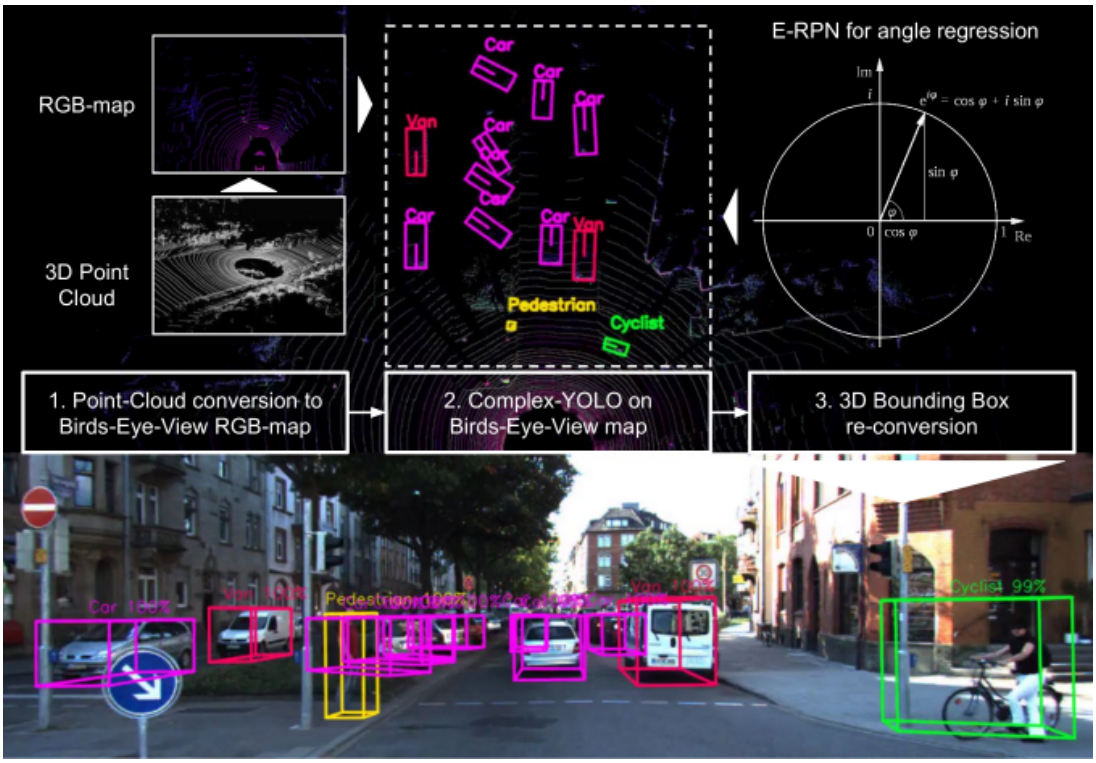

Figure 2.1.6: Complex-YOLO Architecture. The network takes only point cloud as input and generates 3D detection for each class. [6]

The models mentioned above are LiDAR-based only, i.e., the model only takes LiDAR point cloud as input and predicts a 3D bounding box. With time, certain modifications have been made on the model, e.g., using YOLOv3 and YOLOv4 combined with Complex-YOLO replacing the original YOLOv2 used.

2.2 Simulation of Autonomous Vehicles

This section will explain how the simulated environment can be used in autonomous driving research, mainly the CARLA simulator, how it can be used to create different scenarios for testing purposes, and extensive usage for generating a broad set of synthetic datasets.

2.2.1 CARLA Simulator

CARLA [24] is an open-source urban driving simulator for autonomous driving research. It supports the development, testing, and validation of autonomous urban driving models, including perception and control. It is built on the Unreal gaming engine platform and provides flexible API (Application Programming Interface) for users to modify the environmental condition and sensor suites as per the usage. CARLA allows users to choose between different open-source urban layouts, buildings, and vehicles, both dynamic and static. Also, it is possible to create a custom high definition map with the needed assets. The main advantage of CARLA over other simulators is the high degree of similarity with the real world.

CARLA is mainly designed as a server-client system, i.e., client API (implemented in Python) and server, see figure 2.2.1. The server is responsible for creating an environment and actors: cars, motorbikes, pedestrians, sensors, etc. Apart from formulation, it is also responsible for the simulation of physics. The client provides an interface for the user to configure and control the assets spawned in the simulation.

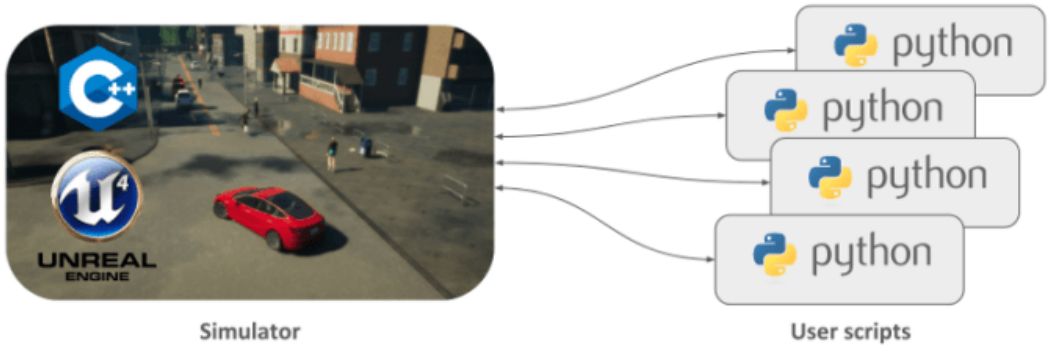

Figure 2.2.1: CARLA Architecture [24]

In CARLA, actors include not only vehicles and pedestrians but also the sensors. The client enables the user to select the sensor's position on the vehicle and set the configuration. The client receives the data from sensors and vehicles for every frame. The CARLA sensors are mainly classified into three categories: Cameras, Detectors, and Others, which are further divided

into sub-categories.

Cameras include a depth camera, RGB camera, and semantic segmentation. Detectors constitute a Collision sensor, a Lane invasion sensor, and an obstacle sensor. Others involve GNSS, IMU, LiDAR, Radar, RSS, and Semantic LiDAR sensors [24]. In this thesis, mainly the Depth camera, RGB camera, and LiDAR sensors are used for the dataset generation, which will be discussed in the following chapter.

RGB camera acts as a regular scene capturing camera, gathering images of the environment based on the configured field of view. The depth camera determines each pixel's depth in the image and creates a depth map of the surroundings. There are two options in CARLA to get a depth view, i.e., Logarithmic depth and depth, as shown in figure 2.2.2. In the simulation, LiDAR sensors create a 3D map of the surrounding using ray-casting. Ray-cast calculates the depth of each pixel for every frame. For each pixel in the scene, the method cast a light ray through the pixel to identify the visible surface [25]. The points are computed based on the vertical field of view, and the point cloud is estimated by doing ray-cast for every laser.

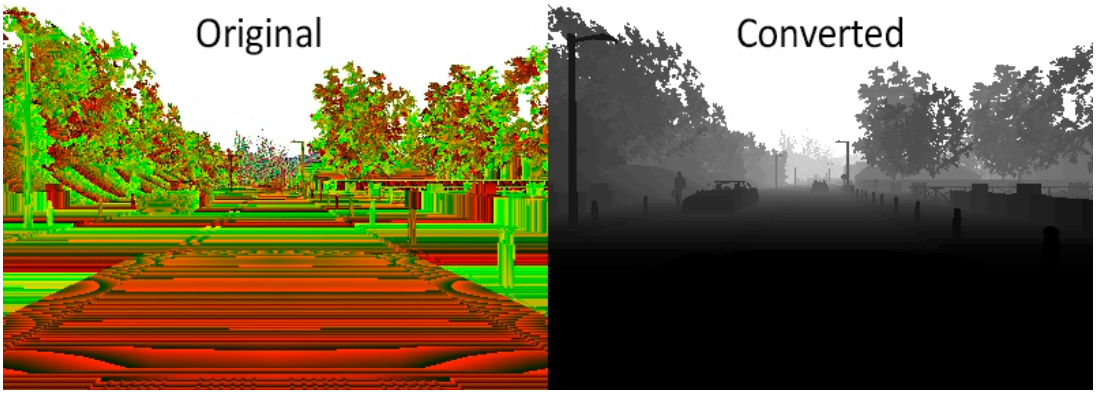

Figure 2.2.2: CARLA Depth camera. On Left: original depth image. On Right: Converted depth image into Logarithmic depth [24]

2.2.2 Applications of simulation environment

Recent advancements in autonomous driving simulators have provided a great source of data for testing and validating an algorithm by incorporating various

scenarios. The use of CARLA is gaining the researcher and engineers' interest to generate a diverse and large dataset. To facilitate generating a large dataset, CARLA has provided multiple scenarios for the user to choose from. There are around 15 weather conditions and a variety of illumination regimes. These differ in the sun's orientation and color, the color of the sky, atmospheric fog, cloudiness, and precipitation. Besides, there are five towns available for simulation, providing plenty of options for the user to create different scenarios and collect large datasets for testing and validation.

Figure 2.2.3: Example of four different weather conditions in CARLA[24]

2.3 Evaluation Metrics

Quantitative evaluation of the object detection algorithms is essential to compare the performance between two algorithms. This section will cover the main performance metrics available to evaluate an object detection algorithm. Different challenges and competitions attempt to examine object detection algorithms by using a variety of performance metrics.

The most common and widely used metric is Average Precision (AP) and mean Average Precision (mAP) for all classes, which will be explained later in this section. The threshold value for calculating AP has been changed accordingly

in each of the competitions to increase the difficulty. Before moving forward on how to calculate Average Precision, other concepts need to be reviewed.

Intersection Over Union (IoU):

IoU is a measure based on the area of intersection between two bounding boxes. It requires a ground truth bounding box (denoted by BB_{gt}) and a predicted output bounding box (denoted by BB_p) from the detection algorithm. IoU is calculated by dividing the intersection area between BB_{gt} and BB_p by area of union between them [26].

$$IoU = \frac{area(BB_p \cap BB_{gt})}{area(BB_p \cup BB_{gt})} \tag{2.1}$$

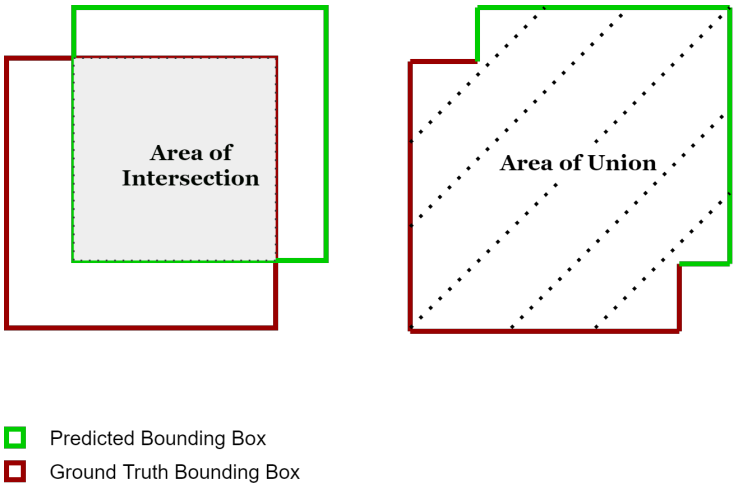

Figure 2.3.1: Left: Intersection Area, Right: Union Area

Applying IoU against the set threshold can then determine valid detections. The threshold is generally set to 50%, 75%, or 90%. The Lyft 3D Object detection challenge [27] for autonomous vehicles uses around ten thresholds for Average Precision. As there are many bounding boxes in the image that should not be detected, the True Negative (TN) result is not applied.

- **True Positive (TP)**: A correct detection. Detection with IoU $\geq$ Threshold.

- **False Positive (FP)**: An incorrect detection. Detection with IOU $<$ Threshold
- **False Negative (FN)**: Ground truth bounding box not detected.

Precision & Recall:

- Precision: It is a measure of the identification of relevant objects in the given image. In simpler words, it signifies the accurate detections the model has made. It is given by:

$$Precision = \frac{\#TP}{\#TP + \#FP} \tag{2.2}$$

- Recall: It is a measure of the identification of all the ground truth bounding boxes, i.e., how well the model has detected the True Positive in the image [28]. It is given by:

$$Recall = \frac{\#TP}{\#TP + \#FN} \tag{2.3}$$

- A precision score of closer to 1.0 that means there is a high likelihood that whatever the classifier predicts as a positive detection is, in fact, a correct prediction. Recall score close to 1.0, which means the model will positively detect almost all objects in the given dataset.
- High recall but low precision implies that all ground-truth objects have been detected, but most detections are incorrect (many False Positives).
- Low recall, but High precision implies that all predicted boxes are correct, but most ground truth objects have been missed (many False Negatives).
- With high precision and high recall, the ideal detector has most ground truth objects detected correctly.

Precision vs Recall Curve (PR curve):

Precision vs Recall curve is a plot between precision and recall for all the detected bounding boxes with different confidence scores predicted by the model. For a model, the precision will be high if the False Positive rate is low, and there are many cases where the model misses many True Positive, which results in a high False Negative rate, and hence the model has a low recall. A model is considered accurate if it finds all the ground-truth objects in the image (high recall) while classifying all the relevant object classes (high precision). Hence, an accurate object detection model should have high precision and high recall by keeping a trade-off between them for all the threshold values [28].

On plotting Precision vs. Recall, the Area Under the Curve (AUC) should be high, indicating high precision and recall. The PR curve generally follows a zig-zag pattern with increasing recall and decreasing precision.

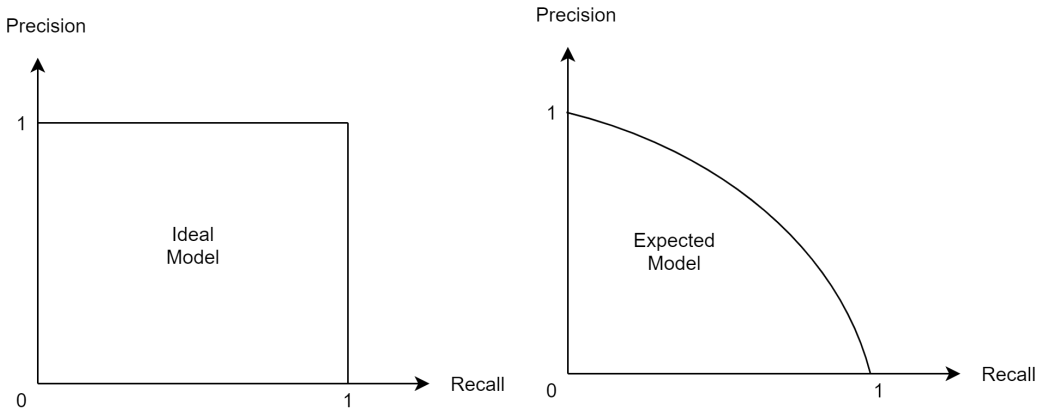

Figure 2.3.2: Precision vs. Recall Curve for ideal and expected model

This Precision vs. Recall curve can be summed by a single metric called Average Precision (AP). Salton *et al.* proposed this metric for the first time in [29]. There are three approaches to calculate Average Precision; in this thesis, two approaches are targeted, 11-point interpolated method and a 40-point interpolated method, and finally 40-point interpolation method is used to calculate the Average Precision, discussed in next chapter. It estimates the

form of precision vs. recall curve as:

$$AP|_R = \frac{1}{|R|} \sum_{r \in R} P_{interp}(r) \tag{2.4}$$

where,

$$P_{interp}(r) = \max_{r':r' \geq r} P(r')$$

Here $P(r')$ addresses the precision through recall r, meaning that, rather than averaging over actual observed precision values through point r, the maximum precision value at recall value greater than equal to r is taken [28].

F-score:

F-score is a measure of the model's performance accuracy by combining precision and recall. It is a harmonic mean of precision and recall. F-score is needed to maintain a balance between precision and recall of a model [30]. The generalized F-score is given by:

$$F_\beta = (1 + \beta^2) \frac{Precision \times Recall}{(\beta^2 \times Precision) + Recall} \tag{2.5}$$

where β is variable weight. In practical applications, it is likely to adjust β as per the desired output, such as to yield a result with more importance to precision than recall. To give more preference to recall than precision, β can be set to 2, keeping recall twice as important as precision. The standard used F-score is F_1 score ($\beta = 1$), and is given by:

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.6}$$

A perfect model has F_1 score of 1. The main advantage of using F_1 score is keeping the balance between precision and recall when there is a large number of uneven class distributions.

Mean Average Precision (mAP):

Mean Average Precision is simply the average of Average Precision over all the specified classes. It is a measure to determine the model's accuracy over all the categories [28]. It is the final measure of the model's performance because it takes the average of the performance of the model over all the classes, and in a sense reflects the mean performance, and is given by:

$$mAP = \frac{1}{|N|} \sum_{n \in N} AP_n \tag{2.7}$$

where N is the total number of classes.

Chapter 3

Methodology

This chapter presents a detailed description of the methods used for the thesis work. The chapter begins with the intricate architecture of the pipeline used in the thesis, continues with the explanation of the occlusion filter to filter out the obstructed objects and scenarios used to create the simulated dataset. The chapter ends with the method used for calculating Average Precision and evaluation method for 2D & 3D object detection modules and comparing different modules.

3.1 System Architecture

The architecture of the pipeline used in this thesis consists of several blocks, as shown in figure 3.1.1. The first block in the pipeline is the organization's [1] CARLA server and client, consisting of smaller modules for a specific function. In the company's CARLA client, the bounding box block is responsible for drawing bounding boxes around the objects and displaying them to the user. The spawn vehicles module is accountable for spawning actors (vehicles and pedestrians) in the environment, and the traffic manager inside vehicle control regulates the action of the actors. In the server, the environment model is to be set up accordingly, i.e., different weather conditions or town maps used for the thesis and sensor model block describes the model of the used sensors (Camera

[1]MAN Truck & Bus SE

& LiDAR).

Once the complete setup is ready, the dataset is generated from CARLA, from which the information of ground truth bounding boxes is saved, and the dataset is then passed through the pre-trained detection modules (2D and 3D detection). The detection modules generate the predicted bounding box information as output, which is later fed into the perception module evaluator. This operation is done for all the modules that are needed to be evaluated. The pipeline's output is the final value of evaluation metrics to analyze the individual object detection module's performance.Based on the values of the performance metrics, the modules are compared.

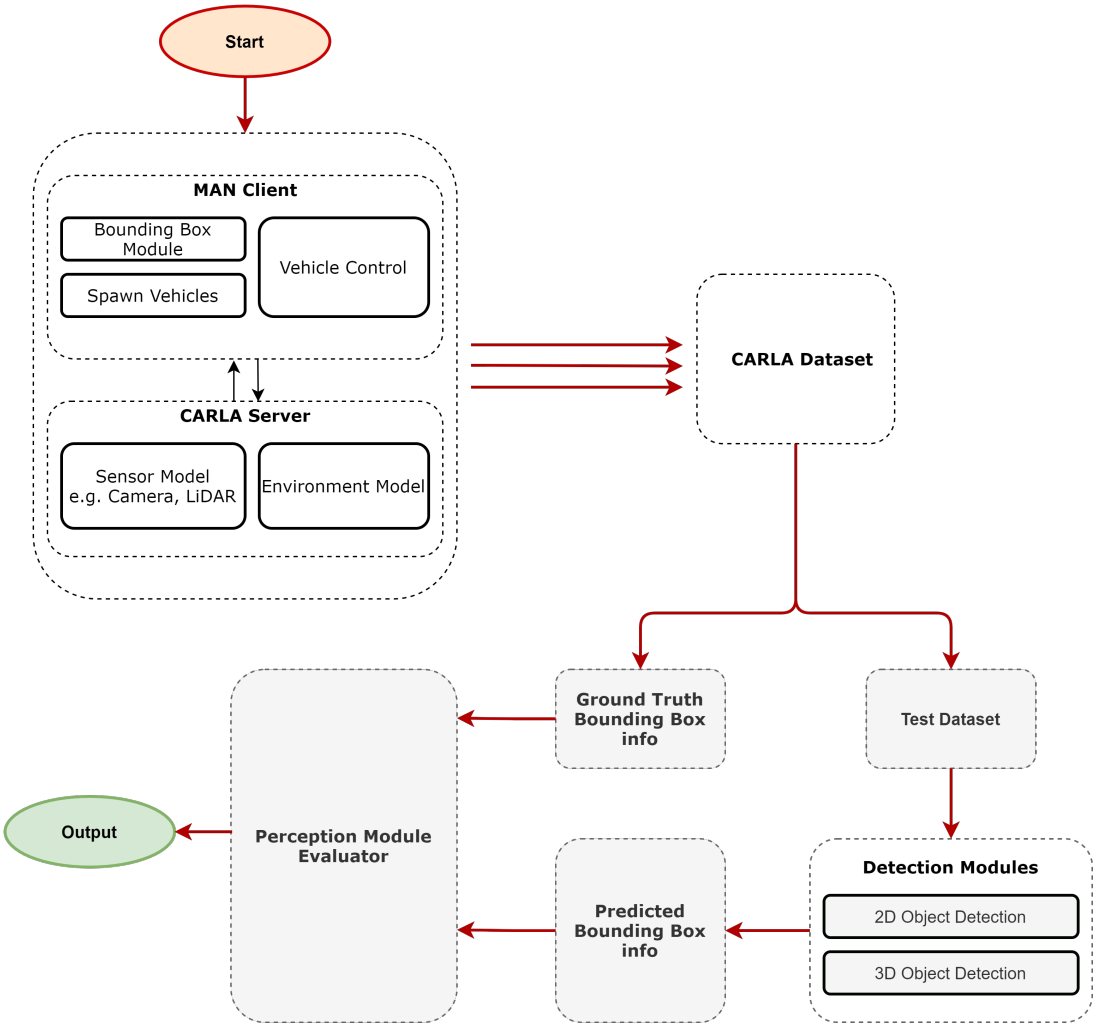

Figure 3.1.1: Architecture of the Pipeline

3.2 CARLA Dataset

As an autonomous driving development simulator, CARLA provides various modules for the user to a particular use. One of the modules is a bounding box generator that allows the user to generate 3D bounding boxes of the vehicle and pedestrians. Vehicle class includes car, truck, motorbike, and bicycle. For this thesis work, along with 3D bounding box ground truth, 2D bounding box ground truth is also needed. There is no direct function available in the simulator for the generation of 2D bounding box ground truth; therefore, a different method is used to generate 2D bounding box for objects in the CARLA environment.

Also in CARLA, there is an issue of occluded objects when generating bounding boxes. When an object is in the front of the camera but blocked by a building or tree, the client draws a bounding box on the vehicle, as shown in the figure 3.2.1. Three filters are used to rectify this problem of occluded objects:

- Distance Filter
- Field of View Filter
- Occlusion Filter

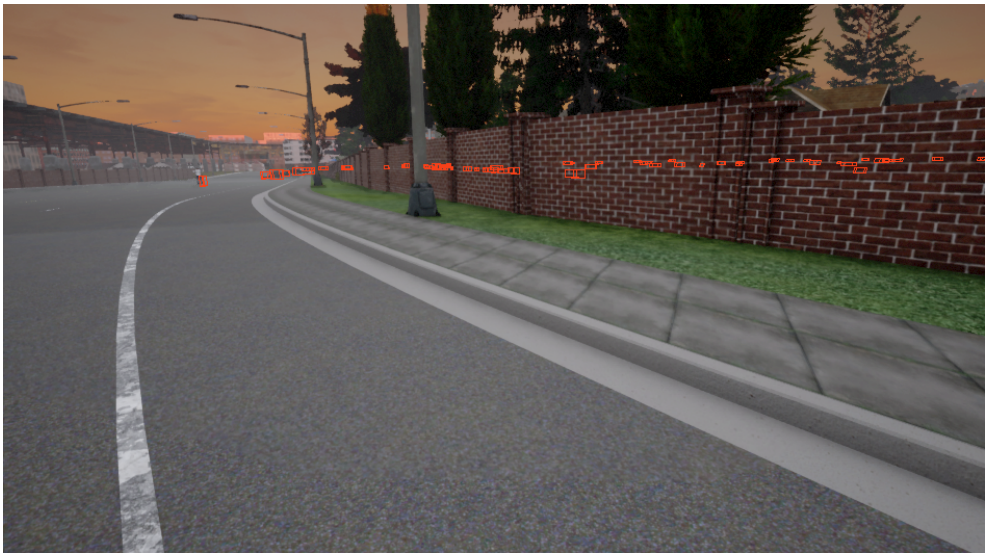

Figure 3.2.1: Occluded objects bounding box

Distance Filter

The distance filter removes the actors (vehicles and pedestrians) that are far from the maximum distance. The user can set the maximum distance as per the usage. The maximum distance is the distance between the camera and the actor.

In the figure 3.2.2, the actor at a distance greater than the maximum distance is rejected, and only actor within the maximum distance range is approved for drawing a bounding box. This filter is essential for unbiased testing as the client will draw bounding boxes on the actor irrespective of the distance and save the information of the ground truth bounding box. This false data will affect the evaluation of the module for which this data is used. Therefore, to create a valid dataset, it is vital to consider the distance filter.

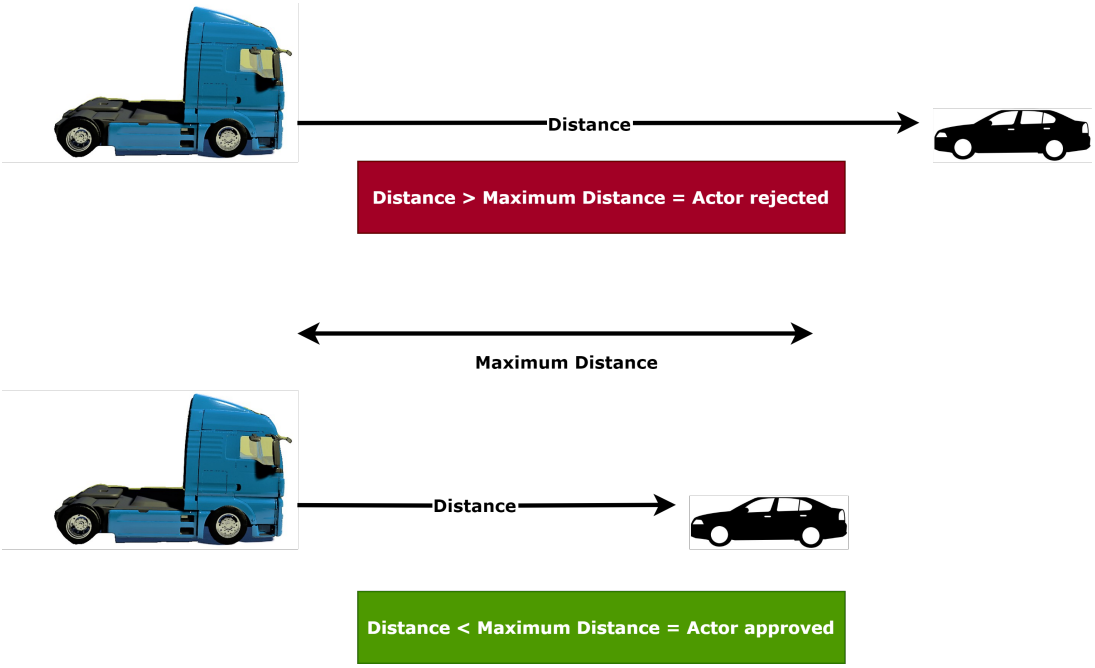

Figure 3.2.2: Distance filter

Field of View (FoV) Filter

The field of view of a camera represents the area of the surroundings the lens can see. The field of view filter removes the actors outside the camera's field of view. The user can set the camera's field of view as per the usage, either the

wide-range field of view or narrow range. In the figure 3.2.3, the actors outside the camera's field of view are rejected, and only actors within the camera's field of view are approved for drawing a bounding box. For the thesis purpose, the camera's field of view is 120 degrees.

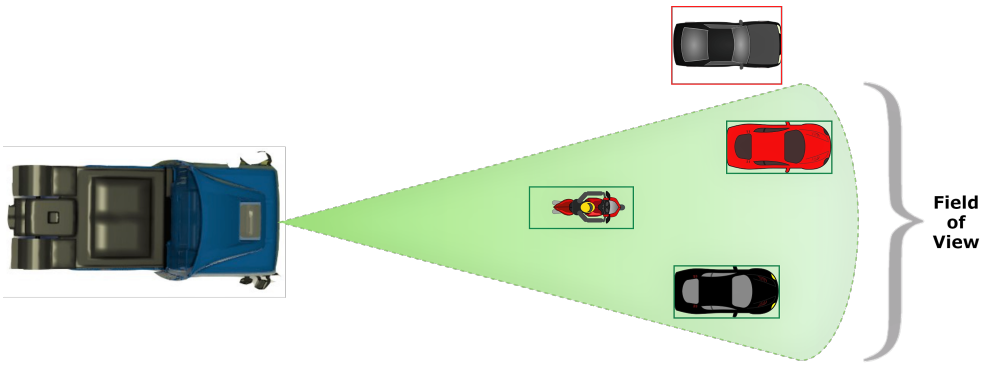

Figure 3.2.3: Field of view filter

Occlusion Filter

The Occlusion filter removes actors that are occluded by other objects such as buildings, trees, or other vehicles between the target actor and camera. The filtering is based on comparing the distance measurement between camera and actor to the actor's depth measurement from the depth camera. If the actor's depth measurement is less than the actor's actual distance from the camera, then that particular actor is occluded. For example, if a tree or building occludes an actor, then the depth measurement will be calculated from the camera to the tree, and this depth will be less than the actual distance between the camera and the actor, refer figure 3.2.4

Since CARLA provides eight coordinates of the bounding box for each actor, the above logic will be applied to each of the points, i.e., for each point, the comparison between depth and distance to the particular coordinate will be made. If all the eight points are not occluded means, the actor is not occluded, as shown in algorithm 1.

Algorithm 1: Filter out Occluded actors

Input: All bounding boxes $\longrightarrow 8 \times 3$ matrix
Output: Filtered bounding boxes

```
/* num visible points: number of visible points required to
   draw the bounding on the vehicle                          */
```

begin
  for *points* **in** *bounding_boxes* **do**
    $count = 0$
    V = [] `// empty list containing visible points`
    for *bbox* **in** *points* **do**
      $x = int(bbox[0, 0])$
      $y = int(bbox[0, 1])$
      $z = int(bbox[0, 2]) \longrightarrow$ distance from camera to (x, y)
      calculate depth of (x, y)
      if *depth* $> z$ **then**
        $count = count + 1$
      else
    end
    if *count* $>$ *num visible points* **then**
      append points to a V
    else
  end
  return visible points
end

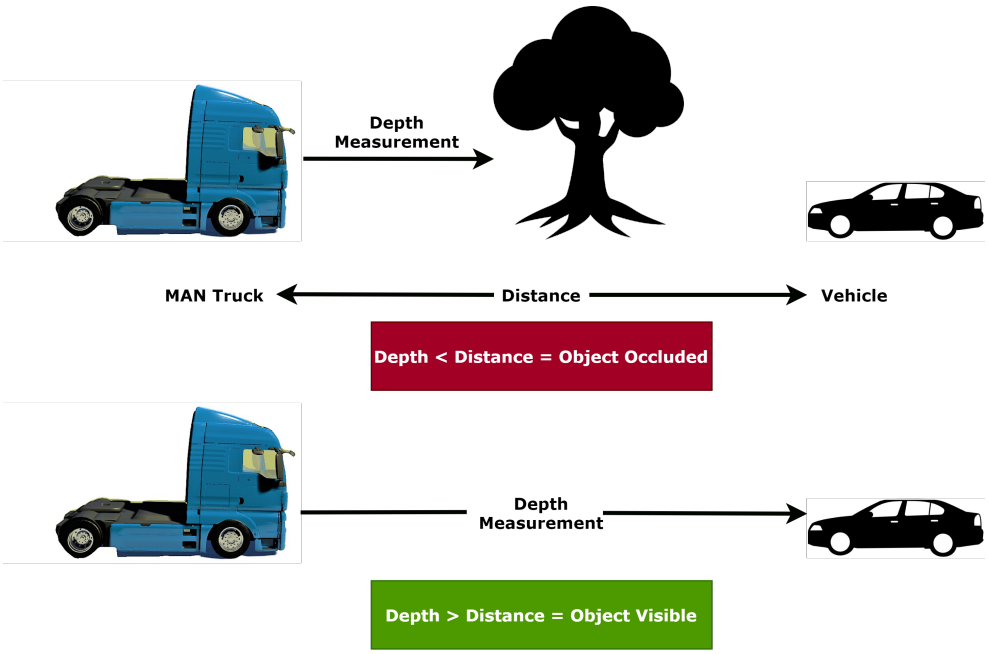

Figure 3.2.4: Occlusion Filter

3.3 Test Scenarios

This section explains the different weather conditions used to evaluate the detection modules.

The most critical problem in autonomous vehicle development is bad performance under unfavorable weather conditions such as heavy rain, fog, bad lighting [31]. For getting a better and clear picture of the performance of detection modules, it hence becomes necessary to consider such scenarios. Therefore, four weather conditions are considered in this thesis for evaluation, i.e., clear weather, hard rainy weather, night time, and fog as shown in figure 3.3.1, 3.3.2, 3.3.3, and 3.3.4 respectively

The parameters for the amount of rain, the visibility on the roads, the fog density are set accordingly. The scenarios considered for evaluating modules cover datasets from five towns in CARLA with different weather conditions. Along with the weather parameters, traffic parameters are also set accordingly to vary the vehicles and pedestrians in simulation.

Figure 3.3.1: Clear Weather

Figure 3.3.2: Foggy weather

Figure 3.3.3: Night time weather

Figure 3.3.4: Hard Rain weather

3.4 Perception Module Evaluator

This section explains the evaluation method for 2D and 3D object detection.

In 2D and 3D object detection, Intersection Over Union has been widely employed as an evaluation metric to calculate the intersection between ground truth and detections predicted by the model.

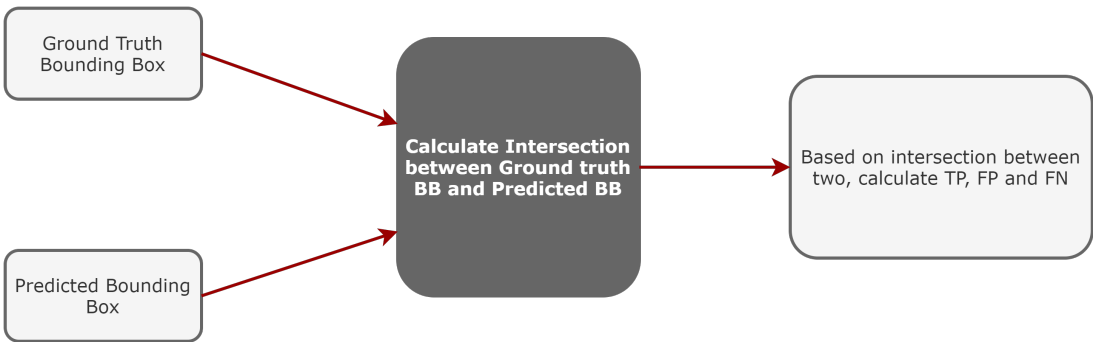

Figure 3.4.1: Evaluation Framework

3.4.1 2D Object Detection Evaluation

In 2D object detection, the object detection model draws a rectangular bounding box around the predicted object on the given input image in the image reference frame. Along with the 2D bounding box coordinates, the model also gives the object's confidence score. For the thesis work, the predictions have been made for the classes: Pedestrian, Car, and bicycle. 2D bounding boxes are parameterized by four coordinates $(x_{min}, y_{min}, x_{max}, y_{max})$, as shown in figure 3.4.2. The coordinates system for the ground truth bounding box is the same as the predicted bounding box, i.e., image reference frame.

For each image passed through the model, there is a corresponding ground truth, as shown in figure 3.4.3, with which the detections are compared in the end to calculate the performance of the model. By comparing the intersection between detections and ground truth[2], Intersection over Union (IoU) is calculated, which gives TP, FP, and FN. Later, based on TP, FP, and FN, key performance metrics values are calculated.

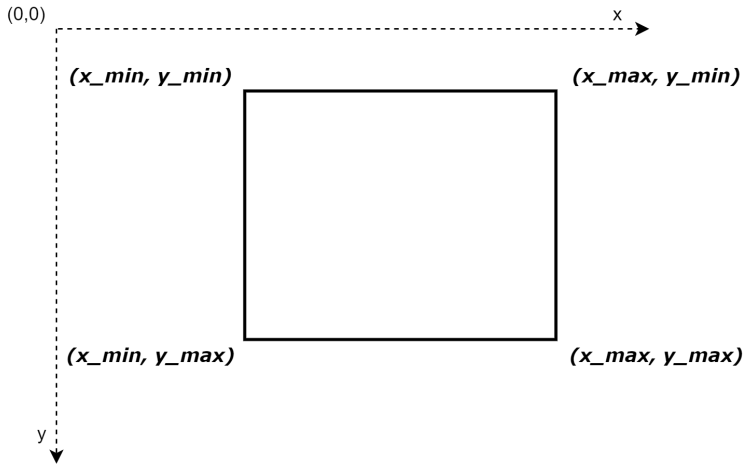

Figure 3.4.2: Reference for 2D Bounding Box Coordinates

[2]ground truth is obtained from CARLA

Figure 3.4.3: Multiple 2D Ground Truth

3.4.2 3D Object Detection Evaluation

In this thesis work, for 3D object detection, unlike RGB images, the LiDAR point cloud is used as an input to the model. In 3D object detection, the model draws a 3D bounding box around the object and gives the 3D bounding box's coordinates along with the class label as the output. As compared to rectangular 2D bounding boxes, the model's output is a rectangular cuboid bounding box.

Unlike 2D object detection, where the detection is in the image frame, in 3D object detection, the detection is in 3D space or world reference frame. Seven points represent a 3D bounding box: $(x, y, z, w, h, l, \theta)$, where (x, y, z) is the center of the 3D bounding box, (w, h, l) is the box's width, height, and length. θ is the box rotation, i.e., the yaw angle (the roll and pitch angle are very very small or assumed to be zero), refer figure 3.4.4

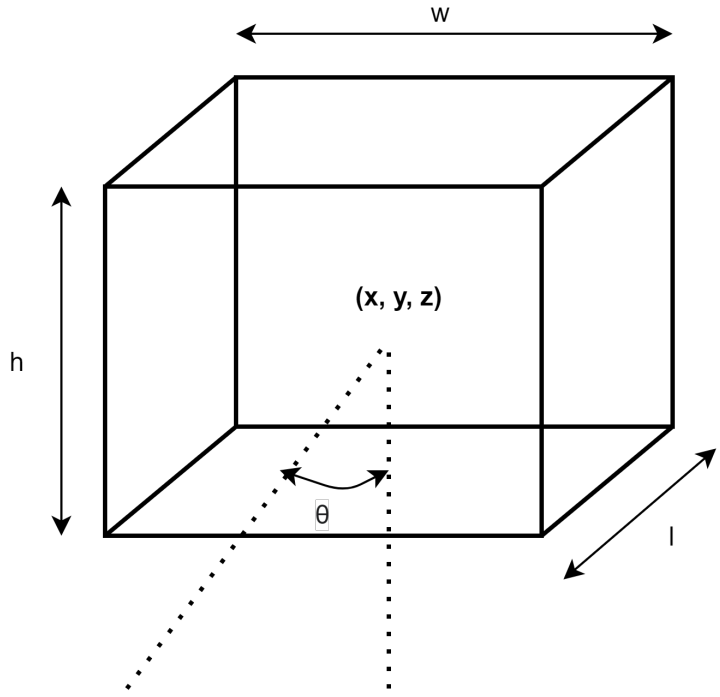

Figure 3.4.4: Reference for 3D Bounding Box

As seen in the previous subsection in 2D object detection evaluation, the comparison is much easier as only the areas are compared without any orientation. Therefore, in that case, the intersection is just a rectangle, enabling easy calculation of performance metrics. In 3D object detection, the evaluation is quite different and complicated as the bounding box's orientation is also considered while calculating Intersection over union. IoU, in this case, is defined for Bird's eye view (IOU_{BEV}) and 3D detection task (IoU_{3D}). The two factors that play a crucial role while calculating intersection are:

- **Bounding Box rotation:** The IoU is positively affected by the rotation of the bounding boxes. Therefore, its consideration is essential during the calculation; however, the roll and pitch angle is assumed to be zero, and only yaw is considered.
- **Bounding Box Volume:** Since the 3D bounding boxes are cuboidal, the intersection is between the volume of ground truth and the predicted box. Therefore, IoU calculation is challenging in the 3D detection task as compared to 2D.

IOU_{BEV} is comparatively easier to calculate, since the bounding boxes drawn over the vehicles are 2D in Bird's eye view map, and a visual representation from 3D Bird's Eye View (BEV) is shown in figure 3.4.5. The green bounding box is used to calculate the IoU overlap in the computation of the average precision. Note: Orientation is opposite, but during computation of IoU, both the boxes would be considered same [32].

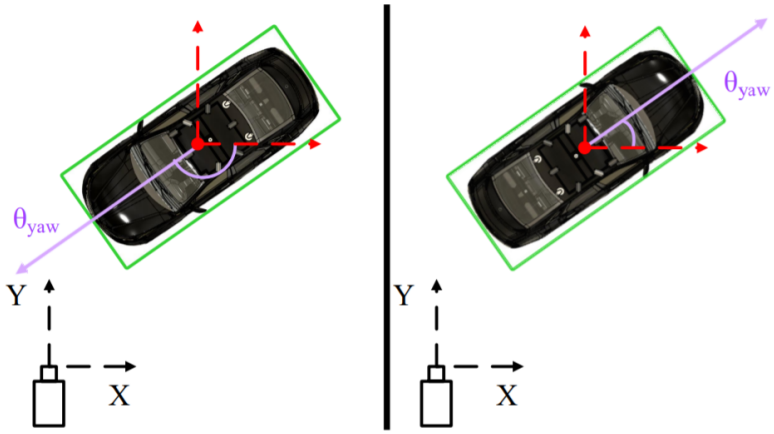

Figure 3.4.5: Bird's Eye View visual representation

Figure 3.4.6: Multiple 3D Ground Truth

3.5 IoU and Average Precision

This section explains the algorithm used to calculate Intersection over Union (IoU), which will further enable the calculation of Precision and Recall, and the method used to calculate AP in this thesis work.

Intersection over Union (IoU):

As explained in the previous chapter, to calculate the IoU, the intersection and union of the ground truth bounding box (BB_{gt}) and predicted bounding box (BB_p) is divided.

It is initially checked if there is any intersection between BB_p and BB_{gt}; if not, then that measurement is ignored. Once there is a valid measurement, the area for both BB_p and BB_{gt} is calculated, and IoU is determined, as shown in algorithm 2.

Algorithm 2: IoU calculation

Input: Ground Truth Bounding Box (BB_{gt}) and Predicted Bounding Box (BB_p)
Output: IoU value

begin
  check if BB_{gt} and BB_p intersect or not
  if BB_{gt} *and* BB_p *do not intersect* **then**
    return 0
  end
  Calculate box area for BB_{gt}
  Calculate box area for BB_p
  Calculate union area between BB_{gt} and $BB_p \longrightarrow$ denoted by Union
  Calculate intersection area between BB_{gt} and $BB_p \longrightarrow$ denoted by Intersection
  $IoU = \frac{Intersection}{Union}$
  return IoU
end

Average Precision (AP):

There are three approaches to compute the Average Precision: 11-point interpolation method, all-point interpolation, and 40-point interpolation method.

In the 11-point interpolation method, the Precision vs. Recall curve is summarized by average precision values at a set of 11 different recall levels, $R_{11} = \{0, 0.1, 0.2, 0.3, ..., 1\}$.

$$AP|_R = \frac{1}{|R|} \sum_{r \in R} P_{interp}(r) \tag{3.1}$$

where,

$$P_{interp}(r) = \max_{r':r' \geq r} P(r')$$

In the all-point interpolation method, the Precision vs. Recall curve is summarized by average precision values at all points instead of just 11-points.

$$AP|_R = \sum_n (r_{n+1} - r_n) P_{interp}(r_{n+1}) \tag{3.2}$$

where,

$$P_{interp}(r_{n+1}) = \max_{r':r' \geq r_{n+1}} P(r')$$

In 40-point interpolation method [33], the average precision is computed at 40 equally spaced recall points with R_{11} by $R_{40} = \{1/40, 2/40, 3/40, ..., 1\}$, thus calculating average precision. This method removes the issue encountered at lowest recall case, and allows the user to get a clear evaluation of the model. In this thesis work, 40-point interpolation method is used to compute average precision.

Chapter 4

Experiments and Results

This chapter contains the results of different experiments and the evaluation of the models on particular scenarios. The analysis of the models is based on the selected performance metrics. The evaluation is done on 2D & 3D object detection models.

4.1 2D Object Detection Models

This section represents the quantitative evaluation and analysis of 2D object detection models. The dataset is generated from the CARLA simulator with four different scenarios discussed in the third chapter for the assessment. The four models used for the evaluation are YOLOv3, YOLOv3-Tiny, YOLOv4, and YOLOv4-Tiny. The results for these models are presented for each weather condition in the upcoming sections.

Considering the main objective of the thesis work, the models are pre-trained and not disciplined on CARLA's dataset. The dataset is generated from MAN's CARLA client, and the vehicle used in the dataset generation is MAN truck. The dataset has $21,454$ randomly[1] generated images from 5 different towns in CARLA, including all four weather conditions.

[1] simulating the vehicle in auto-pilot mode

Clear Noon:

This section explains the performance of detection modules in the case of a clear day. This weather scenario consists of a total of $6,454$ images. Figure 4.1.1 shows the precision-recall curve for the model depicting best performance among all, and table 4.1.1 shows the value of Average Precision for three classes: car, bicycle, and person. The plot is only shown for the model that has performed best out of the existing models.

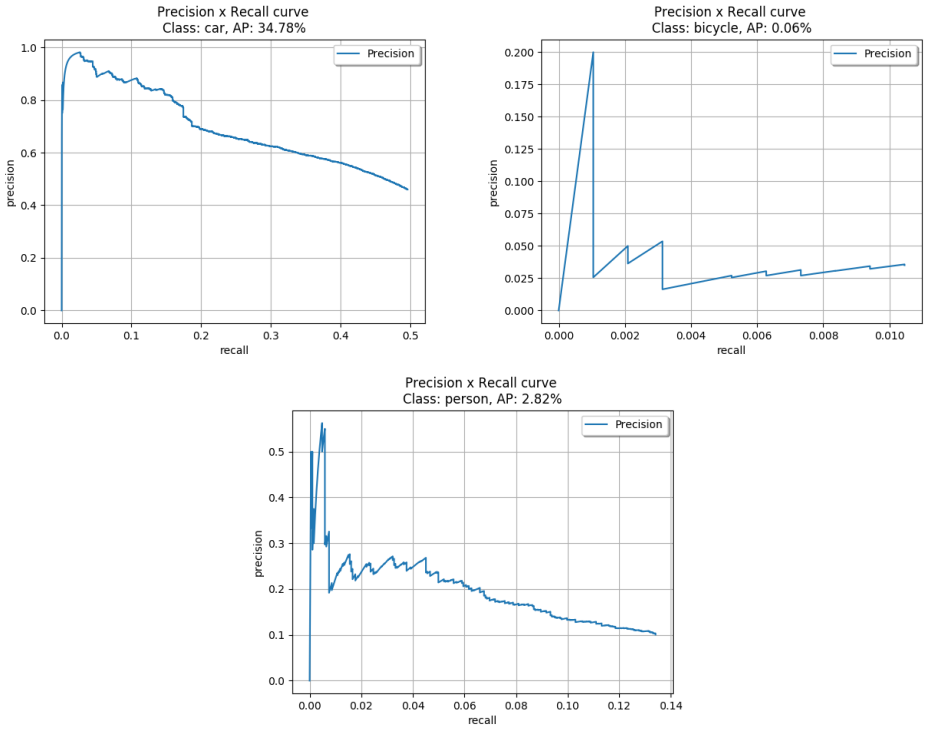

Figure 4.1.1: Precision-Recall curve for scenario: Clear Noon

Scenario: Clear Noon			
Modules	AP_{car}	$AP_{bicycle}$	AP_{person}
YOLOv3	34.78	0.06	2.82
YOLOv3-Tiny	3.86	4.42	0.15
YOLOv4	21.23	2.47	1.12
YOLOv4-Tiny	0.58	4.58	0.07

Table 4.1.1: Average Precision(in %) of detection algorithms for scenario: Clear Noon

The precision-recall plot (figure 4.1.1) is not as expected curve discussed in section 2.3 because the performance of the models on the simulated dataset is not good as they are pre-trained on real-time data.

From the table 4.1.1, it can be observed that the performance of model YOLOv3 is better than the others for the class car and person, signified by a high Average Precision value. However, for the class bicycle, YOLOv4-tiny has outperformed the others.

Fog:

This section explains the performance of detection modules in the case of a foggy weather. This weather scenario consists of a total of $5,122$ images. The figure 4.1.2 shows the precision-recall curve for one of the model, and the table 4.1.2 shows the value of Average Precision for three classes: car, bicycle, and person.

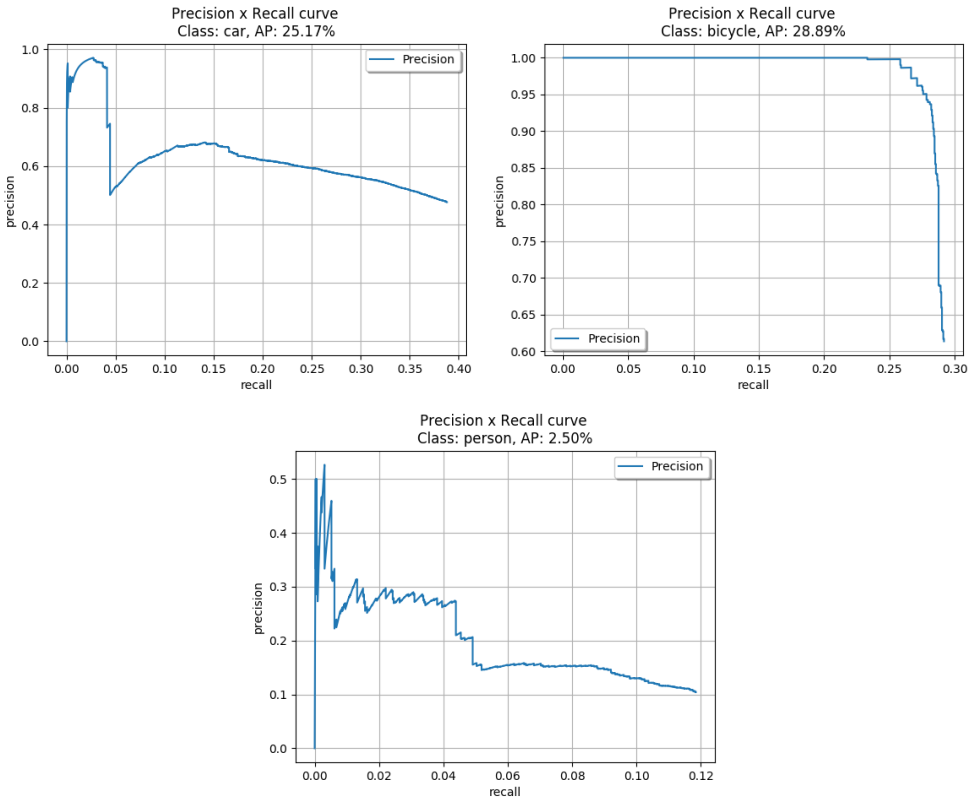

Figure 4.1.2: Precision-Recall curve for scenario: fog

Scenario: Fog			
Modules	AP_{car}	$AP_{bicycle}$	AP_{person}
YOLOv3	23.52	27.64	2.07
YOLOv3-Tiny	18.35	8.24	1.54
YOLOv4	25.17	28.29	2.50
YOLOv4-Tiny	16.93	0.03	1.58

Table 4.1.2: Average Precision(in %) of detection algorithms for scenario: Fog

From the table 4.1.2, it can be observed that the performance of the model YOLOv3 has dropped, which performed better in the case of Clear Noon weather scenario. For all the three classes, the model YOLOv4 has outperformed all the models.

Night:

This section explains the performance of detection modules in the case of a night scene. This weather scenario consists of a total of $5,138$ images. The figure 4.1.3 shows the precision-recall curve for one of the model, and the table 4.1.3 shows the value of Average Precision for three classes: car, bicycle, and person.

From the table 4.1.3, it can be observed that the performance of the model YOLOv4 is not the best for the class bicycle, but it performed better for the classes car and person. For the class bicycle, YOLOv4-Tiny has shown the best performance. The lousy performance of the models can be easily justified because of the lack of dataset available for the night time. Most of the open public dataset only provides images for the daytime; thus, these pre-trained models have performed poorly in this case.

However, it should be noted that the value of Average Precision for model YOLOv3-tiny and YOLOv4-Tiny is zero, which signifies that for these models, the value of both Precision and Recall was zero. These two models were not able to correctly classify any object in the given image.

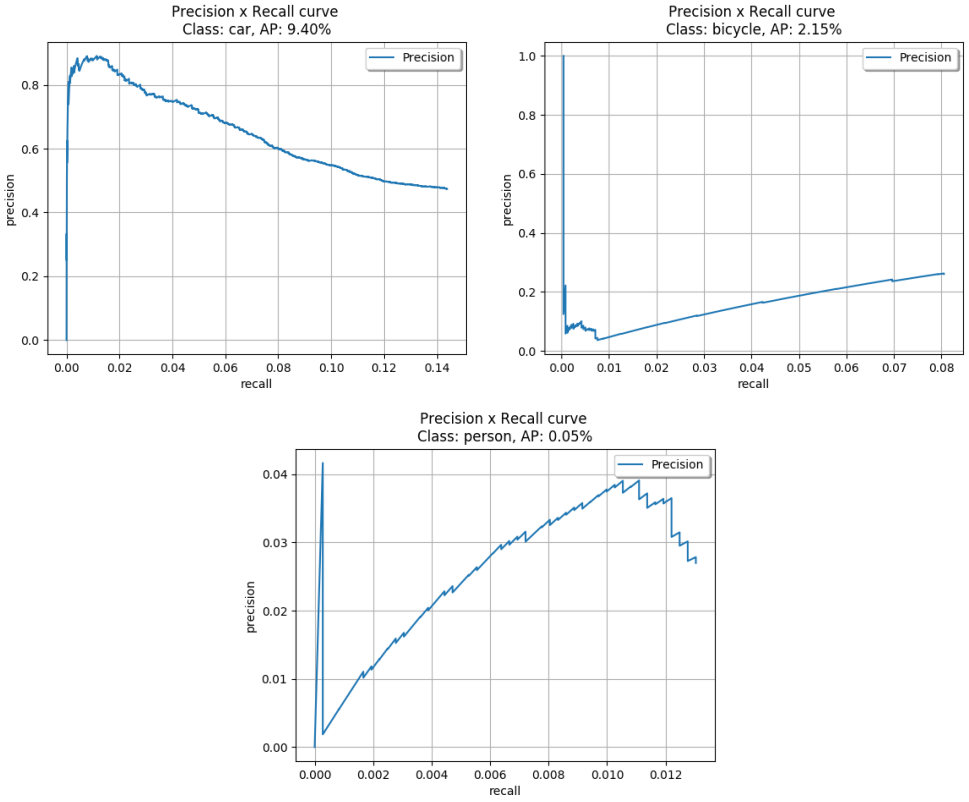

Figure 4.1.3: Precision-Recall curve for the scenario: Night

Scenario: Night			
Modules	AP_{car}	$AP_{bicycle}$	AP_{person}
YOLOv3	7.14	2.69	0.04
YOLOv3-Tiny	1.10	0.89	0.00
YOLOv4	9.40	2.15	0.05
YOLOv4-Tiny	0.40	3.41	0.00

Table 4.1.3: Average Precision(in %) of detection algorithms for scenario: Night

Hard Rain:

This section explains the performance of detection modules in the case of a night scene. This weather scenario consists of a total of $5,730$ images. The figure 4.1.4 shows the precision-recall curve for one of the model, and the table 4.1.4 shows the value of Average Precision for three classes: car, bicycle, and

person.

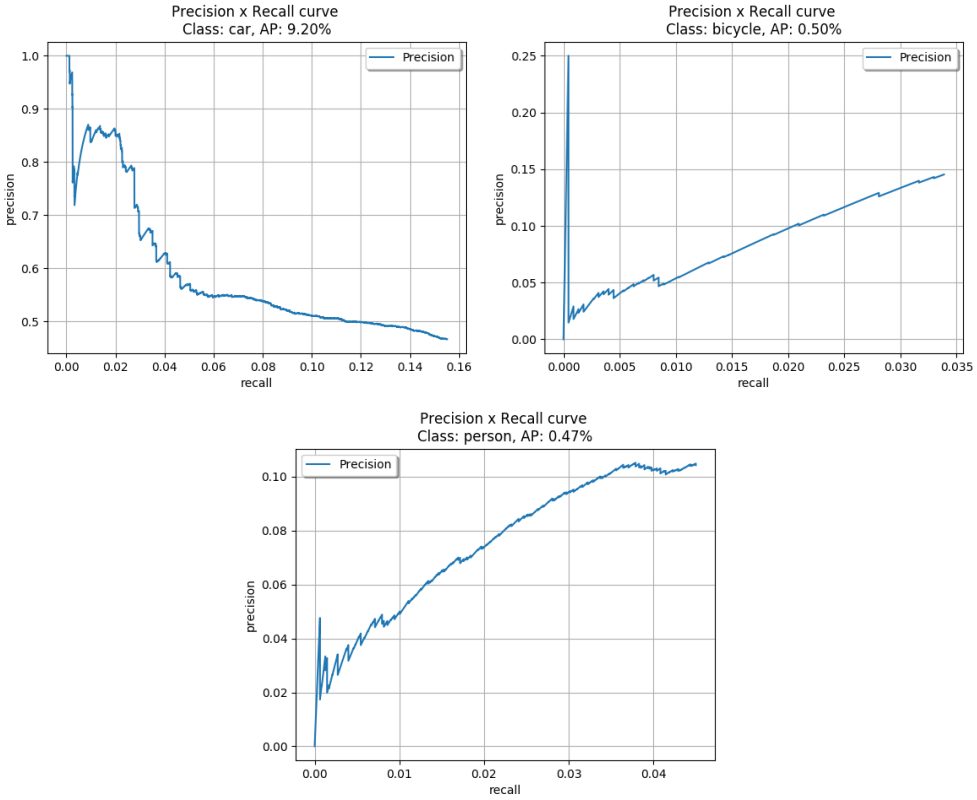

Figure 4.1.4: Precision-Recall curve for scenario: Hard Rain

Scenario: Hard Rain			
Modules	AP_{car}	$AP_{bicycle}$	AP_{person}
YOLOv3	8.72	0.49	0.13
YOLOv3-Tiny	5.23	0.36	0.08
YOLOv4	9.20	0.50	0.47
YOLOv4-Tiny	5.34	0.35	0.30

Table 4.1.4: Average Precision(in %) of detection algorithms for scenario: Hard Rain

From the table 4.1.4, it can be observed that the performance of the model YOLOv4 performed better for all the three classes. The performance of YOLOv3 is quite similar to YOLOv4 in this case.

Mean Average Precision (mAP)				
Modules or Scenario	Clear Noon	Fog	Night	Hard Rain
YOLOv3	12.55	17.74	3.29	3.11
YOLOv3-Tiny	2.81	9.37	1.33	1.89
YOLOv4	8.27	18.85	3.86	3.39
YOLOv4-Tiny	1.74	6.18	1.27	1.99

Table 4.1.5: Mean Average Precision(in %) of all models

The table 4.1.5 shows the value of Mean Average Precision for all the models for the four given scenarios. The mean average precision is the final key performance metric based on which an organization can decide about a particular model. The Mean Average Precision table allows the company to have an exact comparison between all the models. By evaluating all the models in different scenarios, the robustness of the developed pipeline is validated. From the table 4.1.5, it can be observed that the model YOLOv4 has performed better for all the scenarios except clear weather, where YOLOv3 has comparatively performed better.

Computational Time				
Modules/Scenario	Clear Noon	Fog	Night	Hard Rain
YOLOv3	3.48	3.94	2.71	3.07
YOLOv3-Tiny	2.15	2.68	2.03	2.93
YOLOv4	3.57	4.05	2.90	3.30
YOLOv4-Tiny	2.01	2.89	2.51	2.90

Table 4.1.6: Execution time (in s) of all models for all the scenarios

The table 4.1.6 shows the execution time needed by the evaluation framework to compute the performance of an individual model for all the scenarios. As per the results obtained, for each module, the execution time is approximately 3 seconds. This additional metric is added to analyze the performance of the

proposed architecture, which allows the organization to perform a comparison in the future with other available evaluation frameworks. The execution time metric, thus, quantifies the performance of the proposed pipeline in this thesis work. Hence, for an organization, along with the performance of the selected algorithms, it becomes requisite to also take into account the time taken by the framework to evaluate the performance of these modules.

4.2 3D Object Detection Models

This section represents the quantitative evaluation and analysis of 3D object detection models. The dataset used in assessing models is the KITTI dataset, released by Karlsruhe Institute of Technology and Toyota Technological Institute. The KITTI dataset has been used for benchmarking for so many years and consists of 7,481 training images and 7,518 testing images. Since it is the most used dataset for benchmarking and training, it is used in the pipeline developed for the evaluation.

The four models used for the evaluation are Complex-YOLOv3, Complex-YOLOv3-Tiny, Complex-YOLOv4, and Complex-YOLOv4-Tiny.

Average Precision			
Modules	AP_{car}	$AP_{bicycle}$	AP_{person}
Complex-YOLOv3	97.89	90.12	82.70
Complex-YOLOv3-Tiny	95.92	78.7	49.20
Complex-YOLOv4	97.91	89.04	81.6
Complex-YOLOv4-Tiny	93.42	79.29	48.31

Table 4.2.1: Average Precision(in %) of all models

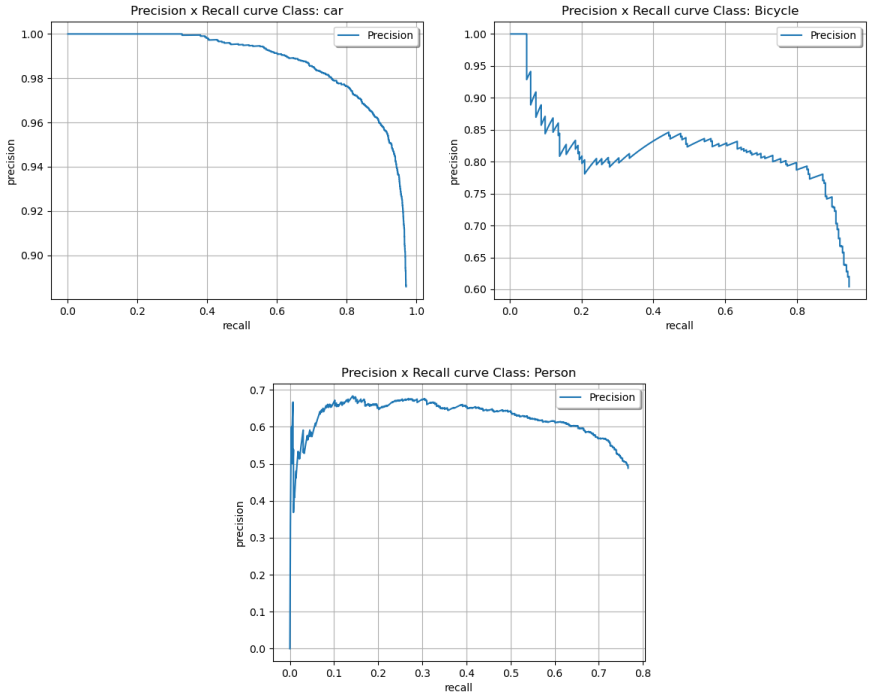

Figure 4.2.1: Precision-Recall curve

Mean Average Precision (mAP) and F-1 score		
Modules	mAP	F-1 score
Complex-YOLOv3	90.23	0.86
Complex-YOLOv3-Tiny	74.6	0.74
Complex-YOLOv4	89.51	0.84
Complex-YOLOv4-Tiny	73.6	0.73

Table 4.2.2: Mean Average Precision(in %) and F-1 score(Average over all three classes) of all models

Table 4.2.1 shows the value of AP for all three classes: car, person, and bicycle, and the figure 4.2.1 shows the precision-recall curve for the model depicting best performance. From the table, it can be observed that Complex-YOLOv3 has performed best for the three classes except for the class car. The table 4.2.2 shows the mAP for all the models, and out of which Complex-YOLOv3 performed the best. As explained in the previous section, this is the final metric value based on which the organization [2] can decide whether to use this

[2]MAN Truck & Bus SE

particular model or not.

Computational Time	
Modules	Execution Time
Complex-YOLOv3	6.28
Complex-YOLOv3-Tiny	5.90
Complex-YOLOv4	6.59
Complex-YOLOv4-Tiny	6.04

Table 4.2.3: Execution time (in s) of all models for 3D evaluation

The table 4.2.3 shows the execution time needed by the evaluation framework to compute the performance of an individual model for 3D object detection. As per the results obtained, for each module, the execution time is approximately 6 seconds, which is comparatively greater than the execution time needed for evaluating 2D object detection algorithms (as shown in table 4.1.6). The main reason behind this is the two complexities that are added while evaluating 3D modules ,i.e., volume of the 3D bounding box and the orientation of the bounding box in the world reference frame. This additional metric as previously mentioned is added to analyze the performance of the proposed architecture, which allows the organization to perform a comparison in the future with other available evaluation frameworks. The execution time metric, thus, quantifies the performance of the proposed pipeline in this thesis work for 3D object detection evaluation. Hence, for an organization, along with the performance of the selected algorithms, it becomes requisite to also take into account the time taken by the framework to evaluate the performance of both 2D and 3D object detection algorithms.

Based on the research and implementation for this thesis work, the possible metrics for evaluating the performance of an object detection algorithm is Average Precision, Mean Average Precision, and F-score. The intention of using these metrics is to enable an easy way to analyze the performance more efficiently. Using these metrics, the framework was designed to evaluate different object detection algorithms. The results section shows the

performance output for each module from the proposed framework. Also, to quantify the performance of the proposed pipeline, the results for the execution time metric was also mentioned.

Chapter 5

Conclusions

5.1 Conclusion

In this thesis work, the evaluation and analysis of 2D and 3D object detection modules have been discussed. This thesis proposes an end-to-end robust architecture to evaluate different detection algorithms, where we just need to use the weights and configuration file for each module and directly compares their performance. This thesis work also presents the possible key performance metrics that can be used by an organization or individual to assess a perception system. The work also benchmarked the performance of pre-trained 2D object detection algorithms on the $21,454$ images generated from the CARLA simulator.

The developed architecture is well capable of evaluating different detection algorithms in a more accessible and simplified way. To validate the robustness of the developed architecture, total eight (four for 2D object detection, and four for 3D object detection) algorithms have been tested and generated their performance results. The thesis work also proposes an easy way to generate a 2D and 3D dataset from the CARLA simulator. The dataset collected from the developed pipeline is well-structured and in a generalized format, which can be used for testing purposes and in the future to train neural networks.

5.2 Future Work

Since the pipeline developed in this thesis work is modular and robust for evaluating different algorithms, validated by the use of different algorithms and getting the complete performance analysis, it allows the user to evaluate an object detection algorithm for even more classes; also, the pipeline automatically considers the number of classes mentioned in the ground truth data, providing scalable architecture. Further, to improve detection algorithms' performance on the simulated dataset, the neural networks can be trained on the simulated dataset generated from the developed pipeline along with the real-world data. And the performance of such modules can then be easily evaluated on the developed pipeline. Apart from just evaluating the performance of detection modules on images, future work could be to use the same proposed architecture to evaluate the performance on video sequences. And, another metric can be added to quantify the performance of the pipeline such as versatility. Therefore, the scope of the proposed framework is not just limited to this thesis work and can be extended further.

Bibliography

[1] Yaqoob, Ibrar, Khan, Latif U, Kazmi, SM Ahsan, Imran, Muhammad, Guizani, Nadra, and Hong, Choong Seon. “Autonomous driving cars in smart cities: Recent advances, requirements, and challenges”. In: *IEEE Network* 34.1 (2019), pp. 174–181.

[2] Cheng, Hong. *Autonomous intelligent vehicles: theory, algorithms, and implementation*. Springer Science & Business Media, 2011.

[3] Khairul, I and Bhuiyan, A. *LIDAR sensor for autonomous vehicle*. Tech. rep. Technical report, Technische Universität Chemnitz, 2017.

[4] Naghavi,
Seyyed Hamed, Avaznia, Cyrus, and Talebi, Hamed. “Integrated real-time object detection for self-driving vehicles”. In: *2017 10th Iranian Conference on Machine Vision and Image Processing (MVIP)*. IEEE. 2017, pp. 154–158.

[5] Redmon, Joseph, Divvala, Santosh, Girshick, Ross, and Farhadi, Ali. “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.

[6] Simon, Martin, Milz, Stefan, Amende, Karl, and Gross, Horst-Michael. “Complex-yolo: Real-time 3d object detection on point clouds”. In: *arXiv preprint arXiv:1803.06199* (2018).

[7] Alessandro Lori, PhD. *Are Self-Driving Cars Safe?* Aug. 2019. URL: `https://www.verizonconnect.com/resources/article/are-self-driving-cars-safe/`.

[8] Woollaston, Victoria. *How Do Google's Driverless Cars Work?* Aug. 2014. URL: https://www.alphr.com/cars/7038/how-do-googles-driverless-cars-work.

[9] Schrank, David, Eisele, Bill, Lomax, Tim, and Bak, Jim. "2015 urban mobility scorecard". In: (2015).

[10] *2D Bounding Box Annotation Machine Learning: 3D Image Bounding Box*. Nov. 2020. URL: https://www.cogitotech.com/bounding-box-annotation/.

[11] Ren, S., He, K., Girshick, R., and Sun, J. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149.

[12] Girshick, R., Donahue, J., Darrell, T., and Malik, J. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587.

[13] Girshick, R. "Fast R-CNN". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448.

[14] Lin, T., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. "Feature Pyramid Networks for Object Detection". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 936–944.

[15] Lin, Tsung-Yi, Maire, Michael, Belongie, Serge, Hays, James, Perona, Pietro, Ramanan, Deva, Dollár, Piotr, and Zitnick, C Lawrence. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[16] Redmon, J. and Farhadi, A. "YOLO9000: Better, Faster, Stronger". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6517–6525.

[17] Redmon, Joseph and Farhadi, Ali. *YOLOv3: An Incremental Improvement*. 2018. arXiv: 1804.02767 [cs.CV].

[18] Bochkovskiy, Alexey, Wang, Chien-Yao, and Liao, Hong-Yuan Mark. "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: *arXiv preprint arXiv:2004.10934* (2020).

[19] Choe, Jaesung, Joo, Kyungdon, Rameau, Francois, Shim, Gyumin, and Kweon, In So. "Segment2Regress: Monocular 3D Vehicle Localization in Two Stages." In: *Robotics: Science and Systems*. 2019.

[20] Engelcke, M., Rao, D., Wang, D. Z., Tong, C. H., and Posner, I. "Vote3Deep: Fast object detection in 3D point clouds using efficient convolutional neural networks". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017, pp. 1355–1361.

[21] Geiger, Andreas, Lenz, Philip, and Urtasun, Raquel. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[22] Zhou, Y. and Tuzel, O. "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4490–4499.

[23] Qi, C. R., Liu, W., Wu, C., Su, H., and Guibas, L. J. "Frustum PointNets for 3D Object Detection from RGB-D Data". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 918–927.

[24] Dosovitskiy, Alexey, Ros, German, Codevilla, Felipe, Lopez, Antonio, and Koltun, Vladlen. "CARLA: An Open Urban Driving Simulator". In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.

[25] Roth, Scott D. "Ray casting for modeling solids". In: *Computer Graphics and Image Processing* 18.2 (1982), pp. 109–144.

[26] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., and Savarese, S. "Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression". In: *2019 IEEE/CVF Conference on*

Computer Vision and Pattern Recognition (CVPR). 2019, pp. 658–666. DOI: 10.1109/CVPR.2019.00075.

[27] *Lyft 3D Object Detection for Autonomous Vehicles*. URL: https://www.kaggle.com/c/3d-object-detection-for-autonomous-vehicles/.

[28] Padilla, R., Netto, S. L., and da Silva, E. A. B. "A Survey on Performance Metrics for Object-Detection Algorithms". In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*. 2020, pp. 237–242.

[29] Dillon, Martin. *Introduction to modern information retrieval: G. Salton and M. McGill. McGraw-Hill, New York (1983)*. 1983.

[30] Goutte, Cyril and Gaussier, Eric. "A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation". In: vol. 3408. Apr. 2005, pp. 345–359. ISBN: 978-3-540-25295-5. DOI: 10.1007/978-3-540-31865-1_25.

[31] Zang, Shizhe, Ding, Ming, Smith, David, Tyler, Paul, Rakotoarivelo, Thierry, and Kaafar, Mohamed Ali. "The Impact of Adverse Weather Conditions on Autonomous Vehicles: Examining how rain, snow, fog, and hail affect the performance of a self-driving car". In: *IEEE Vehicular Technology Magazine* PP (Mar. 2019), pp. 1–1. DOI: 10.1109/MVT.2019.2892497.

[32] Ku, Jason, Mozifian, Melissa, Lee, Jungwook, Harakeh, Ali, and Waslander, Steven. *Joint 3D Proposal Generation and Object Detection from View Aggregation*. 2018. arXiv: 1712.02294 [cs.CV].

[33] Simonelli, A., Rota Bulo, S., Porzi, L., Lopez Antequera, M., and Kontschieder, P. "Disentangling Monocular 3D Object Detection: From Single to Multi-Class Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), pp. 1–1. DOI: 10.1109/TPAMI.2020.3025077.

TRITA-EECS-EX-2020:926