Mathematisch-
Naturwissenschaftliche
Fakultät

Lernbasierte Computer Vision

Masterarbeit

On Offline Evaluation of 3D Object Detection for Autonomous Driving

Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Lernbasierte Computer Vision
Tim Schreier, tim.schreier@uni-tuebingen.de

Bearbeitungszeitraum: 01.12.2022-28.06.2023

Betreuer/Gutachter: Prof. Dr. Andreas Geiger, Universität Tübingen
Zweitgutachter: Prof. Dr. Zeynep Akata, Universität Tübingen

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Masterarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Tim Schreier (Matrikelnummer 5634978), July 7, 2023

Abstract

One of the central challenges in the development of autonomous driving is the reliable sensor-based perception of the environment. Accurate 3D object detection is of pivotal importance for the required holistic scene understanding. Currently, most research on object detection for self-driving focuses on public detection benchmarks that evaluate object detectors using the average precision metric. However, average precision is a task-agnostic metric that does not consider driving-related priorities like speed prediction or correct heading estimation. It further assigns equal importance to all objects in the environment, even though some perception errors are trivial while others can have fatal consequences. Consequently, multiple authors have suggested more task-specific alternatives to average precision, such as the NuScenes Detection Score (Caesar et al., 2020), the Average Orientation Similarity (Geiger et al., 2012), or the Planning-KL divergence (Philion et al., 2020b). In this work, we conduct the first empirical evaluation testing how predictive different detection metrics are of actual driving performance when detection systems are integrated into a full self-driving stack. We conduct extensive experiments on urban driving in the CARLA simulator using 16 object detection models. We find that the NuScenes Detection Score (Caesar et al., 2020) has a higher correlation to driving performance than the standard average precision metric. Importantly though, our results call for caution on the exclusive reliance on an emerging class of evaluation techniques we refer to as planner-centric detection metrics.

Acknowledgments

I am deeply indebted to my direct supervisors, Kashyap Chitta and Katrin Renz, for their excellent advice and practical assistance throughout this project. Furthermore, I want to express profound gratitude to my family for their unwavering support.

Contents

1 Introduction

The dream of vehicles that can steer themselves without human intervention has fascinated researchers, engineers and enthusiasts for decades. Recent advances in the field of autonomous driving have brought us closer to a future with safer and more efficient transportation. According to estimates from the *World Bank* (2023), 1.35 million people are killed annually in crashes involving cars, buses, motorcycles, bicycles, or trucks, and more than half of those who die are pedestrians, motorcyclists, or cyclists. In fact, traffic accidents are the leading cause of death among children and young adults between the ages of 5 and 29. The vast majority of these accidents are not due to technical failure but are caused by human errors such as speeding, substance abuse, or distraction. Autonomous vehicles and fast communication between vehicles thus hold the potential to prevent most of the fatalities associated with driving. Furthermore, autonomous vehicles can make mobility more accessible, for example, for people who are unable to drive due to age or disability.

The journey toward autonomous vehicles dates back several decades (Pomerleau, 1988; Thorpe et al., 1987). Early research in the field was driven by military applications, aiming to create unmanned vehicles capable of operating in hazardous environments (Albus, 2002). Over time, advancements in sensor technology, computing power, and artificial intelligence algorithms have paved the way for the realization of autonomous vehicles in the consumer market (Carfrae, 2010). Car companies and technology giants have invested heavily in autonomous driving research, significantly accelerating progress in the domain (Niel, 2012).

One of the fundamental challenges in realizing the full potential of autonomous vehicles lies in their ability to perceive and interpret their environment. Computer vision and, more specifically, object detection thus plays a pivotal role in enabling autonomous vehicles to understand and navigate their surroundings safely. The need for accurate detection and classification of objects, such as pedestrians and vehicles, has driven rapid progress in the field of 3D object detection (Qian et al., 2022; Gupta et al., 2021; Mao et al., 2023).

This progress is typically measured on public benchmarks. These challenges consist of large datasets that research groups use to train their detection models. Models are then submitted and evaluated on a hidden test set by the benchmark provider to enable a fair comparison. Ever since the first detection benchmark challenges like the PASCAL VOC (Everingham et al., 2006) became popular, average precision has been used as the standard metric for evaluating the performance of a detection model.

The average precision metric stems from research on information retrieval (Choueka et al., 1971). It measures how well a model identifies all relevant objects (e.g., traffic participants) while punishing models for false positive results (i.e., hallucinating additional objects).

Ideally, detection systems should be evaluated with regard to their impact on driving performance. However, closed-loop tests, where detection systems are integrated into vehicles and tested on driving tasks, are expensive, time-consuming, and dangerous. Average precision thus serves as a proxy metric that does not have these downsides. Recently, an increasing number of researchers in the field of autonomous driving have criticized average precision for its task-agnostic design and proposed task-specific modifications of the metric (B. Deng et al., 2021; Caesar et al., 2020; A. Bansal et al., 2021; Sun et al., 2020). One aspect of average precision that authors criticize is that the metric assigns equal importance to all objects. However, giving equal importance to objects in a scene does not reflect real-world priorities for self-driving. Failing to detect a pedestrian in front of the vehicle can result in a fatal collision, but misjudging the number of parked cars in a car park 30 meters behind the vehicle is practically irrelevant. In search for more task-specific metrics several authors further suggested a planner-centric approach for the evaluation of object detection (Lambert, 2013; Ivanovic & Pavone, 2022; W.-X. Li & Yang, 2023). Planner-centric metrics move away from the average precision paradigm and focus on measuring the effects of perception errors on the downstream path planning module.

This work investigates how the different metrics stack up against each other and how we should compare them. More precisely, we empirically investigate the question of what predictive power the individual metrics have for the actual downstream driving performance. Holding the self-driving pipeline and the evaluation benchmark fixed allows us to study the impact of the detector on driving performance by successively integrating different detection models. The CARLA Driving Score Dosovitskiy et al. (2017) as a measure of driving outcomes serves as a highly task-relevant performance indicator for detection models that allows us to compare them on a common metric. This is achieved by analyzing how strongly average precision and the proposed alternatives correlate with CARLA Driving Scores. In summary,

- we conduct the first empirical investigation that compares object detection metrics according to their correlations with driving performance.
- we train 16 3D object detection models, integrate them into our modular self-driving pipeline, and extensively evaluate the driving performance of the resulting agents in the CARLA simulator.
- we measure the offline detection performance on the driven routes using nine detection metrics and test how strongly these metrics correlate to Driving Scores.

We find that even though average precision is highly correlated with driving

performance, the NuScenes Detection Score (Caesar et al., 2020), a task-specific variation, is even more predictive of driving outcomes. Furthermore, our results show that the planner-centric detection metrics we examine are significantly less indicative of driving performance.

The rest of the thesis is structured as follows. Chapter 2 introduces the relevant background of autonomous driving, object detection, and performance metrics. Chapter 3 then describes the self-driving pipeline we used for online evaluations. In Chapter 4, the metrics that are compared are introduced in detail. In Chapter 5, We present the conducted experiments and our results. In Chapter 6 we present our conclusions and discuss the limitations of our analysis.

2 Related Work

In this chapter, we review related research on autonomous driving (Section 2.1), 3D object detection (Section 2.2), and performance metrics (Section 2.3).

2.1 Autonomous Driving

Research into autonomous driving has come a long way since its beginning in the 1980s. Pioneering work in the space was done by projects like Navlab at Carnegie Mellon University (Thorpe et al., 1987) and the ALVINN (Autonomous Land Vehicle in a Neural Network) project (Pomerleau, 1988). These initial systems used camera input to predict appropriate steering commands. This approach to autonomous driving has later been termed End-to-End learning. In End-to-End learning, sensory input data is directly mapped to driving commands. Self-driving (SD) is thus tackled in a single step. This allows the researcher to directly optimize the model for driving and attain training data easily by linking camera-based recordings to a human expert's driving behavior. While there has been steady progress in research on self-driving over the decades, retrospectively, we can identify three significant revolutions that have enabled the field to make great strides toward increasing autonomy.

The first of these revolutions happened around the year 2000 when GPS sensors and Inertial Measurement Units (IMUs) were developed, making it possible to determine the position of a vehicle with great accuracy. Combined with digital maps, a vehicle could now be localized and thus navigate effectively. In 2005, research into self-driving was also accelerated by the DARPA Grand Challenges, where autonomous vehicles competed for a million dollars in a race across the Mojave Desert. In the first year, no vehicle completed the course, but the subsequent annual challenges attracted much research interest and drove progress in the field. Notably, many winning systems like BOSS (Urmson et al., 2008) or Stanford's Junior (Montemerlo et al., 2008) and Stanley (Thrun et al., 2006) systems, were not based on End-to-End learning but tackled environment perception and motion planning separately. In this modular approach to self-driving, the task is split into consecutive components. There are various ways to break down the task of driving. However, they are often structured such that a perception module first uses the sensor information to parse the scene, recognize objects like vehicles, lanes, and buildings, and estimate their positions. Afterward, a planning module makes behavioral decisions and predicts

the optimal route. Finally, a controller module translates the predicted route into low-level steering commands.

A second revolution in the space occurred around 2006 through improvements in sensor technology. Perception is crucial for autonomous vehicles, and increasing camera resolutions and high-resolution LiDAR systems (see Section 3.3.1) enabled more detailed environment perception. At this time, many larger companies became interested in the space and started making significant investments. Google, for example, started its self-driving efforts in 2009 and developed a prototype called "Waymo." In the subsequent years until 2015, Google invested 1$ billion dollars into R&D for self-driving.

Starting in 2012, a third technological revolution occurred through the uprise of deep learning. Sparked by success like the AlexNet image recognition system (Krizhevsky et al., 2017), rapid progress was made in computer vision and other fields crucial for self-driving. Soon afterward, more investment followed by companies like Uber, Otto, and Tesla. Research has further been accelerated by developing high-fidelity simulators for self-driving (Tideman & Van Noort, 2013; Rong et al., 2020; Benekohal & Treiterer, 1988). Research into AV is inherently expensive and dangerous. Software like the widely used CARLA simulator (Dosovitskiy et al., 2017) allows research group to test their systems at a fraction of the costs and iterate quickly on ideas. While important contributions from industry often rely on real cars or proprietary data (Zeng et al., 2019) and real cars (M. Bansal et al., 2018), many recent central contributions from academia are based on the CARLA simulator (Renz et al., 2022; D. Chen & Krähenbühl, 2022; Zhang et al., 2021; D. Chen et al., 2020; Chitta et al., 2022; Shao et al., 2023)

Today, consumer vehicles that can drive autonomously in many circumstances are already available. However, human operators still need to be in the driver's seat and attentive, ready to intervene when the system fails. Notable exceptions are Mercedes Benz's DRIVE PILOT (*Mercedes-Benz DRIVE PILOT*, 2023), which attained level 3 autonomy certification this year, meaning the driver does no longer need to be attentive to the street in many circumstances, and Baidu's driverless Appollo busses, that entered mass production in China in 2018 (*BBC*, 2018; H. Fan et al., 2018).

What most industry leaders for AV like Waymo, Uber, and Tesla have in common is their reliance on the modular pipeline approach. Modularity has proven to provide some important benefits over End-to-End approaches. Modular systems can be developed in parallel by a large team, making them very amenable to industrial research. They also have the advantage of human-understandable intermediate representations, such as semantic segmentation for perception (Feng et al., 2020), that allow for inspection of module performance and offline analysis of failure cases. Modularity further allows developers to benefit from other established research fields like object detection and tracking. Due to these benefits, this work provides an empirical analysis relevant to the modular approach to self-driving.

2.2 Object Detection

Object detection is one of the most fundamental tasks in computer vision and deals with the classification and localization of semantic objects, mostly in digital images or video. Without prior information about the number of objects in an image, a system is tasked with identifying all relevant objects from a large number of predefined categories and specifying their relative position in the image. Object detection is widely used in applications such as image annotation, content-based image retrieval, medical image analysis, video surveillance, and robot vision (Zaidi et al., 2022). Moreover, it is also often a preliminary step in more advanced computer vision tasks such as object tracking (F. Chen et al., 2022) or action recognition (Zhu et al., 2020). Object predictions are specified using bounding boxes. These tightly fitted axis-aligned rectangular boxes fully encompass the objects (see Section 3.3.2).

Methods for object detection can be categorized into two approaches. The first approach uses feature extraction algorithms and then applies traditional machine learning methods to these features, while the modern deep learning-based approach operates directly on the image pixels. The traditional approach has been the focus of research for decades before the rise of deep learning. Here, features of an image need to be extracted in the first step on which further processing is based. Carefully hand-crafted algorithms like the Scale-invariant feature transform (SIFT) (Lowe, 1999) or the Histogram of oriented gradients (HOG) (Dalal & Triggs, 2005) exploit features like color histograms and edge orientations to create features describing the image. Afterward, machine learning methods like the support vector machines (Cortes & Vapnik, 1995) are used to classify objects and regress their locations. Today, this traditional approach is out-competed by modern methods. Since breakthrough results in applying deep learning to image classification achieved by Krizhevsky et al. (2017), the computer vision community has made immense progress using neural networks. State-of-the-art methods for object detection on images are now based on large convolutional neural networks (CNN) (Matsugu et al., 2003).

One of the first successful deep networks for object detection was the Region-based Convolutional Neural Network (R-CNN) proposed by (Girshick et al., 2015). It used an off-the-self region proposal algorithm to generate an around 2000 box proposals per image. These boxes are cropped and warped to fit the fixed-size input of the CNN. The CNN backbone of the RCNN architecture uses two heads. One acts as the classifier and predicts a class label and a confidence score, while the other one is the box regressor that refines the box location for the predictions. RCNN performed well but suffered from high per-region computational costs and a slow region proposal method with limited recall. The same group quickly removed these constraints through Fast-RCNN (Girshick, 2015), which strongly reduced the per region computation by relying on a single forward pass through the CNN backbone and then Faster-RCNN (Ren et al., 2015), which improved upon the speed and accuracy of the region proposal generation by introducing a region proposal neural network that exploits the features of the backbone. At this point, it is worth noting

that there are methods, such as YOLO (You only look once) (Redmon et al., 2016) and SSD (Single Shot Detection) (Liu et al., 2016), that use a single stage. Although these methods can be significantly faster than the two-stage approaches, they often do not perform as well in practice. But, 2D object detection has proved insufficient for applications like scene understanding (J. Fan et al., 2022) and object tracking (Xu et al., 2019; Luo et al., 2021). It has become clear that richer three-dimensional representations are needed to drive progress in robotics and self-driving.

2.2.1 3D Object Detection for Self-Driving

An autonomous vehicle needs to understand its environment fully. This includes tracking objects in the scene and understanding absolute distances between objects. Introducing a third dimension to estimating the location and extent of objects allows for estimating absolute distances, but it also adds significant complexity to the task. The research community still faces a significant performance gap between 2D and 3D methods (Arnold et al., 2019). Even though significant progress has been made on camera-based depth estimation (Qian et al., 2022; Gupta et al., 2021; Mao et al., 2023), depth estimation from images remains a difficult and fundamentally ill-posed problem. Objects of different sizes can produce the same visual impressions, and triangulation is error-prone at larger distances. There has thus been a strong interest from academia and industry in LiDAR-based 3D object detection for self-driving (Fernandes et al., 2021; Elhousni & Huang, 2020; Y. Li et al., 2020). LiDAR sensors (see Section 3.3.1), use active light sensing to attain accurate depth estimation. The accurate 3D information simplifies the task of 3D object detection, but the LiDAR-based approach has unique challenges, like sparsity and feature extraction from point clouds. The following Section discusses how modern approaches to LiDAR-based 3D detection are structured on a high level.

2.2.2 Operational Pipeline of LiDAR-based 3D Object Detection

Despite architectural differences among the various state-of-the-art detectors, all models share the same operational pipeline. Instead of images, LiDAR-based object detectors use unstructured sets of 3D points as input. Here, a common framework for understanding modern LiDAR-based 3D object detectors is introduced to enable a more structured discussion of individual models and their differences in Section 3.3.3. The operational pipeline consists of three main steps:

First, the LiDAR sensor data is converted to a structured representation. These downstream representations include point-based (S. Shi et al., 2019), voxel(volume pixel)-based (Zhou & Tuzel, 2018), projection-based (front view or bird's eye view) (Beltrán et al., 2018), and pillar-based (G. Shi et al., 2022). Point-based representations leave the original point cloud unchanged or keep the format while down-sampling the data. Voxel-based representations use 3D pixels to encode volumetric information.

Pillar-based approaches first discretize the ground plane into a grid. 3D points from the point cloud are assigned to their closest grid cells to yield the pillar representation. Projection-based methods use various strategies for transforming the point cloud into a 2D representation.

As a second module, there is the feature extraction stage. The feature extraction stage extracts semantic features from the scene representation using deep learning methods. In voxel-based representations encoder or encoder-decoder architectures of 3D convolutional neural networks are used to create high-dimensional feature maps in birds eye view (BEV). In point-based representations, encoder or encoder-decoder backbones based on Pointnet++ (Qi, Yi, et al., 2017) are used for point-wise feature extraction and foreground segmentation tasks. In most projected representations, standard fully convolutional neural networks are applied to extract features.

When the feature extraction step is complete, the core object detection module is applied. Detector networks process the extracted features and predict 3D bounding boxes and associated confidence scores. There are anchor-based and anchor-free detection heads. The anchor-based systems use a per object class fixed size 3D bounding box anchor (S. Shi, Guo, et al., 2020). Anchor-free networks (Q. Chen et al., 2020) do not use anchors and leverage semantic information such as foreground points or centerpoints (Yin et al., 2021) to regress box sizes. Systems from both categories either operate on 2D or 3D features. Another difference arises between single-stage detectors and two-stage detectors. In single-stage detectors (Yan et al., 2018), the final step is to predict bounding boxes from the previously extracted features. In a two-stage detector, predictions of a region proposal network (RPN) are fed into a second stage (S. Shi et al., 2023). The second stage refines the box proposals from the RPN and calculates offsets further to regress the bounding box to the ideal locations. Proposals by the RPN with low confidence scores are rejected. All detection heads output bounding boxes, corresponding class labels, and confidence scores between 0 and 1.

Besides these three stages, there is another aspect that all modern 3D object detectors have in common: the need for large amounts of annotated data during training. The following Section thus briefly introduces important datasets in research on 3D object detection.

2.2.3 Datasets for 3D Object Detection for Self-Driving

As large-scale, specialized datasets are required for algorithmic advancement, many high-quality annotated datasets for self-driving have been published in recent years (Caesar et al., 2020; Chang et al., 2019; Sun et al., 2020; Kesten et al., 2019; Geiger et al., 2012). These datasets are recorded on public roads by cars that are equipped with LiDAR sensors and cameras. The datasets often serve as benchmark challenges, where various 3D detection models can be trained on the same data and evaluated in a fair comparison. Performance on these benchmark datasets is evaluated on a

hidden test set. Evaluation abides by a standardized protocol that specifies object classes, difficulty levels, evaluation metrics, measures of inference times, and similar characteristics. Here, we briefly introduce three noteworthy examples of these benchmark datasets.

The KITTI benchmark dataset (Geiger et al., 2012) comprises 22 video scenes. Contained are real-world driving scenes ranging from country roads to cities and highways. Sensor-wise, it features stereo-vision, LiDAR point clouds, and GPS coordinates that are synchronized in time, as well as relevant sensor-calibration information. Annotations in the form of 3D bounding boxes are categorized into three difficulty levels based on object size, occlusion, and truncation level. A drawback of the dataset is that it covers a limited variety of weather conditions, as most scenes have been recorded during sunny conditions.

The NuScenes object detection benchmark (Caesar et al., 2020) is newer and more extensive compared to the pioneering KITTI dataset. It features 100x as many images from 1000 video scenes and covers scenes recorded at night and in bad weather conditions. It features LiDAR sweeps, six camera views, and 360° radar data. A drawback is that NuScenes features very little geographical diversity, covering an effective area of only five square kilometers (Drobnitzky et al., 2022).

The Waymo Open dataset (Drobnitzky et al., 2022) provides more geographical diversity, having been recorded in three different cities and with changing weather conditions. It includes 1,150 annotated videos recorded by five cameras and their corresponding LiDAR sweeps.

In this work we do not rely these datasets, but generate our training data using the CARLA simulator. For our experiments, we train 16 LiDAR-based 3D object detection models and include architectures based on voxel- (Yan et al., 2018; J. Deng et al., 2021) and pillar-representations (G. Shi et al., 2022), as well as fused approaches that also incorporate point-based information (S. Shi, Guo, et al., 2020; Lang et al., 2019; Yin et al., 2021).

2.3 Performance Metrics for 3D Object Detection

In order to gauge algorithmic progress on object detection, different models are compared on dataset-based competitions like the ones mentioned in Section 2.2.3. Most comparisons are made using confusion matrix-based, scalarized performance measures such as the popular mean average precision metric (Qian et al., 2022). A detailed explanation of mean average precision (mAP) can be found in Section 4.2.

Metrics like mAP reflect the objective of maximizing the amount of true positive detection while minimizing the number of false positives, i.e., recognizing all objects in the scene without hallucinating additional objects. Mean average precision is a task-agnostic metric used to benchmark performance on many information retrieval

tasks. In the object detection setting, it has proved to be a good metric to understand how well a detector learns to recognize and localize objects. In this work, we want to study whether mAP is the appropriate metric to benchmark 3D object detection for self-driving or whether alternative approaches are better suited for the context.

2.3.1 Task-specific Detection Metrics for Self-Driving

While mean average precision is a flexible and informative metric, there are also drawbacks to its task-agnostic design. When it comes to self-driving, many authors have criticized that the metric gives equal importance to all annotated objects in the dataset and does not take the egocentric nature and task-specific characteristics of driving into account (Lambert, 2013; Sun et al., 2020; W.-X. Li & Yang, 2023; Ivanovic & Pavone, 2022). Not detecting a cyclist in front of the ego vehicle is a catastrophic failure, while online, detecting four of the five parked cars on the opposite lane 20 meters behind the ego is a trivial mistake. Yet, mAP weights both error the same. Detectors with identical mAP scores can thus vary in their capacity to detect close and safety-critical objects on the ego vehicle's trajectory and thus produce vastly different outcomes when integrated into a self-driving pipeline. Task-agnostic metrics do, therefore, not accurately reflect what is essential when evaluating perception for AV. To address this problem, multiple authors have proposed novel, task-specific object detection metrics for self-driving. For our experiments, we select a representative span of these metrics, covering all central ideas. The metrics we do not include in our analysis are briefly mentioned below.

Task-specific Variations of mAP

One approach to crafting task-specific detection metrics for SD that multiple authors have pursued is to take the standard mAP as a point of departure and adapt it to the self-driving application by modifying different aspects of the metric.

Sun et al. (2020) argue that correct heading estimation is crucial for motion forecasting and behavior planning. They thus suggest modifications of mAP that reflect the importance of heading estimation (mAPH). In the design of mAPH all true positive detections are weighted by the accuracy of their heading prediction when calculating mAP. The weighting is done by scaling the contribution of true positives with $\min(|\tilde{\theta} - \theta|, 2\pi - |\tilde{\theta} - \theta|)/\pi$, where $\tilde{\theta}$ is the predicted heading and θ is the GT heading in radians within $[-\pi, \pi]$. The resulting mAPH is used as the principal metric for the Waymo Open Dataset 3D object detection challenge (Sun et al., 2020). mAPH is very similar to the Average Orientation Similarity (AOS) proposed by Geiger et al. (2012) that we use for our analysis (see Section 4.2.3).

B. Deng et al. (2021) make the case that mAP should evaluate detections in a way that considers the ego vehicle's relative orientation. Focusing on the shape of predictions, they compute the distance the boundary of the predicted bounding box has to the

boundary of the associates ground truth annotation. They then use the maximal distance in longitudinal or lateral orientation to the ego (whichever is larger) as the support distance error (SDE) and use a threshold on the SDE to determine whether a particular detection is correct. While highlighting shape and translation errors that are particularly safety-relevant, the metric neglects other aspects of prediction quality, such as speed and heading estimation.

Planner-centric metrics

A different and novel approach to detection evaluation for self-driving that does not involve average precision is focused on evaluating the effects of perception noise on the planning module in a modular pipeline.

Philion et al. (2020b) propose a planner-based metric that focuses on the change in downstream planner outputs that is induced by perception errors as a measure of perception quality. More specifically, the planning-KL-divergence (PKL) measures the KL-divergences between distributions of waypoint locations conditioned on noisy perception or ground truth object annotations. The focus on the effect of detection noise on downstream planning makes the metric highly task-relevant, but there are also drawbacks to the approach. Not all perception noise implies behavior changes, even though it can detract from situational awareness. The metric, which Guo et al. (2020a) have found to be only weakly correlated to mAP, also relies on optimized neural networks to be evaluated, making comparisons across contexts difficult (Philion et al., 2020a).

W.-X. Li & Yang (2023) introduce a similar framework but emphasize the importance of understanding the internal reasoning of a planner as opposed to looking at the final outputs like in PKL. This allows them to consider the effects of perception noise that do not cause immediate action by the planner. A drawback of the approach is that it focuses on model-based planners with explicit goal formulations and can not be applied to deep learning-based planners.

In another recent work, Ivanovic & Pavone (2022) describe an approach where local gradients of a planning function are used to assign weights to objects in a scene. However, the approach of probing the effects of the input signal on a cost function also relies on hand-crafted, differentiable planning functions that break down in complex environments.

With Risk Ranked Recall (RRR), A. Bansal et al. (2021) want to test detection systems fit for safety-critical tasks like self-driving. They proposed categorizing detections into three tiers of priority of safety and evaluating recall values identity for every rank. However, RRR is less expressive than mAP, and the authors can not describe a robust procedure for categorizing objects into tiers.

What we find striking about research on novel detection metrics is that none of the authors have provided quantitative results demonstrating that their metrics

are more closely related to driving performance measures than the standard mAP metric. We seek to address this literature gap by comparing offline evaluations with online tests.

2.3.2 Online and Offline Testing

There are two different ways in which deep learning-based systems are tested and evaluated. In offline (or 'open loop') testing, a deep neural network (DNN) is tested individually using a test dataset that has been obtained without using the system under test (SuT) and which has also not been used in the training process. Prediction errors or prediction accuracy is measured on the prerecorded test set. Offline tests can typically be done very rapidly, yet it is often unclear how the resulting measurements relate to system-level functionality in embedded applications.

In online (or 'closed loop') testing, the neural network is often integrated with other components (which might also be DNNs) into a larger system, which is then embedded into a dynamic application environment. Performance is evaluated on holistic metrics that reflect system-level capabilities and real-world outcomes. In closed-loop testing, the SuT's outputs feed back into and alter the environment in a dynamic loop. Online testing provides an ecologically valid evaluation method of system-level performance but is tied to higher costs and sometimes safety concerns.

In the context of self-driving, a system's performance is best evaluated by a large fleet of vehicles on public roads, gathering realistic performance estimates. However, costs, time commitments, and safety risks render real-world evaluations unfeasible for most researchers. Most testing is thus done in simulators that can account for system-level integration and dynamic environment interactions without the downsides associated with real-world tests (Kaur et al., 2021). But even simulation-based testing is associated with vastly greater computational costs and engineering efforts than offline tests. Thus, some initial research has investigated whether offline evaluations can predict the outcomes of online tests.

Haq et al. (2021) evaluate the correlations between online and offline performance of a camera-based end-to-end lane-keeping model using the PreScan simulator (Tideman & Van Noort, 2013). The authors could not reliably predict safety critical lane deviations from offline prediction errors and concluded that offline evaluations could not be used for safety testing in the context of steering prediction. In a similar experiment Codevilla et al. (2018) compare online driving performance to offline prediction accuracy for a camera-based end-to-end driving system in CARLA. They also found only weak correlations between the online outcomes and offline measures of steering errors.

Contrary to the research efforts above, we do not focus on steering prediction but aim to evaluate the effects of perception errors on downstream driving outcomes. More specifically, we investigate task-specific and task-agnostic offline detection

metrics with regard to their correlation to online driving performance. We aim to provide the first quantitative evidence comparing detection metrics for self-driving to help the community choose the most relevant metrics. The following chapter details the self-driving pipeline we use to conduct these experiments.

3 Modular Pipeline for Autonomous Driving

In this chapter, we present our self-driving agent. At first, we introduce the problem setting. Afterward, the modular structure of the proposed agent is given. Section 3.3.3 then explains LiDAR-based 3D object detection, including task, sensor, and output representations. Further, the specific detector architectures utilized for experimentation are detailed Finally, the proposed object tracking approach 3.4 and the planning module 3.5 are described.

3.1 Problem Setting

The modular pipeline discussed in this Section addresses the task of urban driving. The agent's objective is to navigate safely along a predetermined route while adhering to traffic regulations. The agent is equipped with various sensors, including LiDAR, GPS, a speedometer, and an inertial measurement unit (IMU). The agent's outputs (i.e., steering, throttle, and brake controls) are integrated into the simulator and applied at the subsequent timestep. Our agent has privileged access to simulator information concerning the ego lane and encounters with potential red lights. The benchmark described in Section 5 defines the starting points and destinations, and a navigation unit computes the high-level paths.

3.2 Setup

All experiments are conducted using a modular self-driving pipeline. Figure 3.1 presents the high-level structure of the setup. At every timestep, a roof-mounted 360° LiDAR sensor (see Section 3.3.1) scans the environment around the ego vehicle. The resulting 3D point cloud information is then processed by a 3D object detector which produces a set of bounding box predictions (see Section 3.3.2) about the position and characteristics of objects in the scene. Detections are tracked over time by the object tracking unit (see Section 3.4) to yield consistent predictions and speed estimates for detected objects. Given its high-level navigation goal, the planner (see Section 3.5.1) then uses these refined predictions about objects in the scene to decide on an appropriate course of action. The resulting target waypoints encode the planned trajectory. The PID controller processes the waypoints to compute appropriate lateral

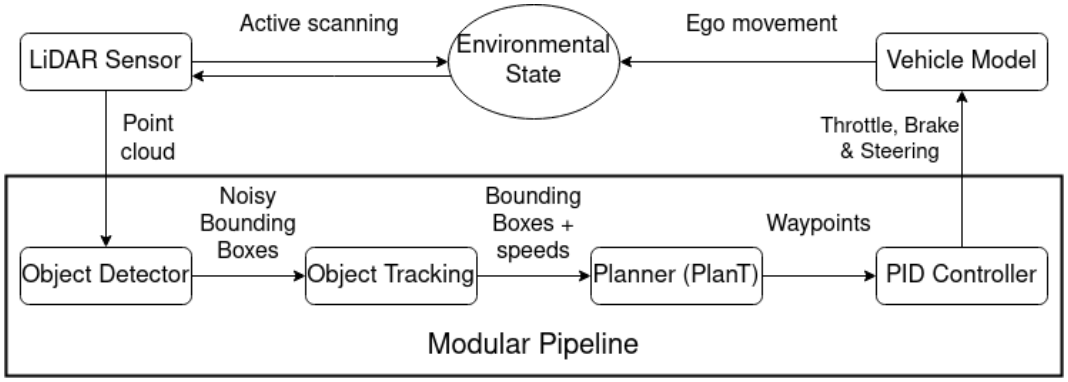

Figure 3.1: A schematic representation of our self-driving pipeline

and longitudinal controls to be executed by the actuators. In order to study the effects of the performance of specific object detectors on the downstream task of driving, all elements of the pipeline except the detector are constant between experiments.

3.3 LiDAR-based 3D Object Detection

3D object detection (3DOD) is one of the most central tasks in computer vision, especially for the robotics community. The task of 3DOD is to detect all relevant object instances in the sensor's field of view. This Section introduces LiDAR sensors, explains their benefits over camera-based systems, and elaborates on the bounding box representation. Afterward, we describe the concrete object detection architectures used for experimentation while highlighting notable differences.

3.3.1 LiDAR Sensors

The abbreviation LiDAR stands for 'light detection and ranging'. A LiDAR sensor measures an emitted laser pulse's time of flight (toF). Given the resulting measure of distance and the horizontal and vertical angles of the laser emitter, a 3D point on an object's surface is detected. Modern LiDAR devices for self-driving applications are equipped with multiple sensors per unit that scan the environment at high frequencies. The object detection models discussed in Section 3.3.3 use the resulting 3D point cloud as input. A common LiDAR system such as the HDL-64E3 outputs 120,000 points per frame. Given a 10 Hz frame rate, this produces a coverage of 1,200,000 points per second (Arnold et al., 2019).

LiDAR sensors offer a variety of benefits compared to cameras. One of the crucial advantages of LiDAR-based detection is that LiDAR sensors provide precise 3D information that represents the absolute scale of objects. This is important for obstacle avoidance and path planning tasks, where accurate depth perception is crucial. In

contrast, camera-based systems rely on visual cues and struggle with accurate depth estimation, especially at longer distances. A full 360° field of view is an additional benefit of LiDAR-based detection. LiDAR sensors are more robust to changes in lighting conditions than camera-based alternatives, which sometimes do not operate reliably when faced with glare or low-light conditions. A final advantage of LiDAR sensors is privacy-friendly data as they do not capture raw visual information.

3.3.2 3D Bounding Boxes for Object Detection

The 3D bounding box representation describes an object's position in 3D space. The box encoding consists of a centerpoint in 3D space $[x, y, z]$, a 3D shape representation of box width, length, and height $[w, l, h]$, and an angle that encodes the yaw orientation θ. The remaining two degrees of freedom, namely the other two rotation angles (pitch and roll), are ignored here, as all objects are assumed to be positioned on the ground plane. Bounding boxes are defined to fully enclose their target object and fit tightly around the object's surfaces. Every bounding box prediction is assigned a class label and a confidence score. Additional attributes that are predicted in the context of self-driving include, e.g., speed, information on whether a car is parked, information on whether a car is flashing emergency lights, and more.

3.3.3 Detector Architectures

For the experiments found in Section 5, eight different modern LiDAR-based 3D object detectors are trained and evaluated. The design of these architectures differs significantly among the aspects of the operational pipeline discussed above. Here, an overview of the detectors is introduced briefly, focusing on the differences and main contributions. Detailed architectural information can be found in the respective papers.

Point-RCNN

S. Shi et al. (2019) propose Point-RCNN. It is a 3D LiDAR detection model whose architecture resembles the camera-based 2D faster RCNN detection architecture (Ren et al., 2015). LiDAR points are first fed through a point cloud encoder and decoder based on Pointnet++ that yield features for each point. Pointnet++ is an elaboration of the pioneering Pointnet (Qi, Su, et al., 2017) architecture. These models have become very popular for processing point clouds as they allow deep learning to apply directly to unstructured sets of points. Pointnet applies multi-layer perceptrons (MLP) per point to create feature vectors. Pointnet++ improved upon this idea by designing the architecture to consider a point's local context explicitly. Pointnet++ does so by hierarchical sampling using Farthest Point Sampling (FPS), grouping points in neighborhoods to establish local context using a metric ball query and then applying a small Pointnet on the neighborhoods in centroid-relative coordinates.

After the Pointnet++ has extracted the features, they are fed into two parallel modules. The first is the 3D RPN that aims to generate an ideally overcomplete set of bounding boxes. In parallel, the authors implemented a foreground-background segmentation module trained using a bin-based loss. The top 100 boxes from the RPN are passed to the second stage of the detector head, which takes point coordinates, semantic features, foreground masks, and the 3D region of interest (ROI) from the RPN as input. Here, offsets that refine the bounding boxes and confidence scores for the predictions are computed. The Point-RCNN architecture is not included in the experiment but is mentioned to provide context to the evolution of detection algorithms.

Voxel-RCNN

J. Deng et al. (2021) propose Voxel-RCNN, which uses a two-stage approach very similar to Point-RCNN. The significant difference is that Voxel-RCNN transforms the point cloud into a voxelized grid representation. The authors argue that this is computationally much more efficient than the point-based approach. The feature extractor is a 3D-CNN-based backbone that processes the voxelized input into feature volumes. These volumes are stacked along the height dimension to create feature maps in BEV. The following 2D-CNN operates as the RPN and feeds its prediction to a second stage which refines the prediction, similar to the Point-RCNN architecture.

SECOND

SECOND (Sparsely Embedded Convolutional Detection) is another voxel-feature-based 3D object detector (Yan et al., 2018). SECOND uses a single stage and fixed-size anchors. After points are assorted to voxels, the Voxel Feature Encoding (VFE) extracts voxel-wise features. The VFE applies a Pointnet architecture to points in voxel relative coordinates to create feature vectors representing individual voxels.

Afterward, spatially sparse 3D convolutions are applied to the extracted features to create 2D feature maps at various resolutions of down-sampling. The design of these sparse convolutions exploits the sparsity of the voxels for computational efficiency. The exploitation process is a significant contribution of the paper and has since been adopted by many voxel-based designs (Mao et al., 2021; S. Shi, Guo, et al., 2020; S. Shi et al., 2023; S. Shi, Wang, et al., 2020). In SECOND, a 2D CNN that uses feature pyramid networks (Lin et al., 2017) processes the resulting 2D feature maps further and a RPN computes proposals for the second stage.

PV-RCNN & PV-RCNN++

S. Shi, Guo, et al. (2020) elaborates on the architectures discussed in Sections 3.3.3

and 3.3.3. They propose a two-staged detector called Point-Voxel-RCNN (PV-RCNN). PV-RCNN merges point and voxel-based approaches and makes use of sparse convolutions. It combines the flexible receptive field and accurate point locations of Pointnet-based methods and the efficient and high-quality 3D proposals of voxel-based sparse convolutions. In the first stage, the authors use a voxelized representation and a SECOND-inspired sparse 3D CNN backbone to extract voxel-wise features. After the 3D CNN, a 2D CNN and an anchor-based RPN further process the features and propose boxes.

PV-RCNN uses sampled keypoints from the point cloud. PV-RCNN selects a set of points using furthest point sampling (FPS), which summarizes the information of the scene while retaining information about accurate locations. The CNN-generated voxel features surrounding keypoints are aggregated by grouping features in the neighborhood via a Pointnet++-like architecture. The scene is encoded by associating the keypoints to the multi-scale features from the 3D backbone.

Two more feature vectors are generated per keypoint. The first one originates from feeding the original points through a Pointnet while the second feature vector is extracted from the feature map of the 2D CNN through interpolation. Keypoints are also re-weighted using a foreground/background segmentation module. The enriched keypoints and the ROIs predicted by the RPN are then fed into the RoI-grid pooling module. Here features of the proposed regions of interest are generated by aggregating features from keypoints in the vicinity. The resulting information is then passed through an MLP for confidence prediction and box refinement.

In a later publication, S. Shi et al. (2023) further improved PV-RCNN and released PV-RCNN++. One of these improvements is the introduction of the sectorized proposal centric keypoint sampling algorithm (SPC). It samples a fixed amount of points in the vicinity of the proposed 3D boxes so that the selected keypoints become more relevant to object detection instead of focusing on background objects. This is especially important because the number of background points significantly outweighs the number of foreground points in LiDAR sweeps for self-driving.

The authors also argue that not all voxels should be weighted equally and thus include a self-attention mechanism to the Pointnet structure to integrate and weight voxel feature information more adaptively. PV-RCNN's ball query has been replaced with a computationally more efficient Manhattan distance-based neighborhood search.

Centerpoint

Yin et al. (2021) propose Centerpoint, an anchor-free two-stage detector. Anchor-based 3D detection makes assumptions about the shapes, sizes, and aspect ratios of objects it encounters. As the world does not abide by these constraints, the authors propose working with the rotationally invariant centerpoints instead of fixed anchors

and regressing object properties from there.

Centerpoint employs a SECOND, VOXELNET or PointPillar (Zhou & Tuzel, 2018; Yan et al., 2018; Lang et al., 2019) based backbone to extract volumetric features from the point cloud. These are flattened to create a dense feature map. Standard CNNs then predict object centers. In the second stage, the additionally extracted point-wise features of these center locations are used to regress all other object properties like orientation, size, and shape. The first stage introduces centerpoints that have no intrinsic orientation. The authors argue that this allows the backbone to learn the rotational invariance of objects.

Pillarnet

Pillar-based methods are computationally much more efficient than point-based or 3D voxel-based convolutions. Unefficiency is a drawback of other approaches, as self-driving heavily relies on real-time processing and limited onboard computational resources. G. Shi et al. (2022) focus on a pillar-based approach that is computationally more efficient, as it only uses 2D convolutions, yet Pillarnet features competitive performance .

In the encoder phase of pillar-based methods, 3D point clouds are projected into a 2D BEV feature map. A 2D grid in the x-y plane is created, and points are sorted to the closest grid cell. The point encoding is then transformed to entail the distance to the center of the grid cell. Afterward, stacked sparse and dense 2D convolutional blocks are combined to extract deep pillar features. Lastly, a standard anchor based detect head is applied to predict class membership and further regress the box.

PointPillars

PointPillars, introduced by Lang et al. (2019), uses a pillar-based representation and point-wise features to create a single-stage object detector. Firstly, points are assorted into an evenly spaced grid in the $x-y$ plane. As in Pillarnet, points are augmented with information about their relative position to the pillar mean and offsets from the $x-y$ center. The authors exploit the sparsity of the pillar representation to compress the information into a tensor of stacked pillars. Let P be an imposed limit on the number of non-empty pillars, N the number of points per pillar, and D the length of the augmented point encodings. The resulting tensor then has dimensions $DxPxN$. A point-net architecture transforms point encodings into point-wise features, creating a $CxPxN$-dimensional representation. A max-pooling along the third dimension then flattens the tensor to dimension CxP. The features are then sorted to the original pillar locations to yield a feature map in BEV space. 2D CNNs then process the point-feature enriched BEV map before a standard single-stage detection head predicts bounding boxes.

PART-A^2

S. Shi, Wang, et al. (2020) base their two-stage detector on the observation that annotated 3D datasets provide well-separated, oriented information about object structures. All foreground points in annotated 3D objects provide information about their relative location within the object box. Part-A2 explicitly uses these intra-object locations to learn about 3D point distributions within objects.

The architecture utilizes a two-stage framework that makes use of a voxelized representation. A 3D CNN-based encoder decoder architecture, which uses sparse convolutions, extracts semantic features in BEV. Afterward, an intra-object part prediction module and a foreground-background segmentation module are applied in parallel. Ground truth information is extracted directly from the provided bounding boxes for both tasks. The authors investigated anchor-based and anchor-free designs for the detection head. They show that the anchor-based version produces better recall rates but requires more memory and computing power. In the more lightweight design, box proposals are predicted directly from features produced by the 3D backbone.

The second stage takes the proposed boxes and the output of both modules described above as input. The information is aggregated via pooling and sparse convolutional layers, and an MLP refines the predictions and assigns confidence scores.

3.3.4 Non-maximum Suppression

Region proposal networks predict multiple bounding boxes in close proximity for a single object. In order to avoid the excessive amounts of false positive detections non-maximum suppression is proposed. Non-maximum suppression (NMS) is a popular post-processing method for object detection modules that removes duplicate detections. NMS compares the confidence scores of overlapping predictions and retains only those with the highest confidence scores. If the IoU value between two bounding boxes exceeds a predefined threshold value, the bounding box with the lower confidence score is discarded. The utilized NMS version used in this thesis can be summarized as follows:

1. Sort predicted bounding boxes according to confidence scores.
2. Choose the prediction with the highest confidence score and save it to a list of finalized detections.
3. Remove all other predicted bounding boxes that share an IoU with the selected detection that exceeds a predefined threshold.
4. Repeat steps 2) and 3) until the initial list of predictions is empty.

3.4 Object Tracking

In this Section, the tracking of detected objects across frames in the modular pipeline is described. It is also explained how the speed predictions for those detections are computed.

Given a sequence of frames describing a scene with objects in them, the task of online multiple object tracking (MOT)(Ciaparrone et al., 2020; F. Chen et al., 2022) is to identify, locate and track instances of objects through time without prior information about the number of objects in the scene and when they appear or disappear. In the experiment's modular pipeline, the object detector handles detection and localization. Therefore, the remaining task is to associate detection through time and thereby maintain instance identities. For self-driving, establishing object tracks is crucial, as it enables estimating the speeds of other vehicles or pedestrians in the scene (Ravindran et al., 2020).

The object detectors outlined in the previous Section are applied on a frame-by-frame basis and thus do not yield sufficient information for the planning unit to understand what the other vehicles in the scene are doing. In the proposed setup, detections are associated across frames using bipartite graph matching which establishes tracks for detections that belong to the same physical object and provide speed estimates.

Let $B_t = \{b_{t,1}, ..., b_{t,n}\}$, $B_{t+1} = \{b_{t+1,1}, ..., b_{t+1,m}\}$ be the sets of bounding box detections at two consecutive frames. Consider the sets of detections as nodes. Then, use the IoUs of detections from different timesteps as similarity scores to represent the edges. The solution to the bipartite graph matching problem is the set of edges between the set of nodes in B_t and B_{t+1} that maximizes the sum over similarity scores. Finally, we compute the optimal solution using the Hungarian method (Kuhn, 1955).

Noisy object detectors generate many false positive predictions. These hallucinations lead to erratic and dangerous driving maneuvers. In order to prevent hallucinations, only detection that are successfully tracked for at least t timesteps are presented to the planning unit. Existing object tracks that are not associated with a new detection with an IoU over 0 are terminated.

The speed of tracked objects is approximated via a simple heuristic. Per object track, the bounding box centers of the last two timesteps is projected to the ground plane. Afterward, the L2 norm between these points is divided by the timestep length to approximate object speed.

3.5 Planning for Autonomous Vehicles

The planning task for self-driving consists of behavioral decisions and motion planning. The process takes the output of the perception stack as input and generates trajectories to be executed by a controller. Approaches to the planning task are

categorized into two camps. Model-based methods rely on explicit goal modeling, while data-driven models leverage deep learning and large amounts of recorded driving data.

Planning is a very challenging and complicated task. In busy highways, the behavior of other vehicles needs to be modeled and taken into account, and even uncommon traffic situations should not lead to unsafe driving behavior. Due to the task's complexity, most modern approaches to planning are based on deep data-driven models (Zeng et al., n.d.; Vitelli et al., 2022; Aradi, 2020). For our experiments, we chose the PlanT planning unit published by Renz et al. (2022). PlanT is a transformer-based planning architecture with state-of-the-art on the CARLA leaderboard challenge (*CARLA Leaderboard.*, n.d.)

3.5.1 Explainable Planning Transformers via Object-Level Representations (PlanT)

Most learning-based planners take dense, high-dimensional grid representations of their environment as input. Authors go to great lengths to produce accurate pixel-level renderings of scenes in BEV to create appropriate inputs for CNN-based planning modules. Renz et al. (2022) propose PlanT, a BERT-based transformer architectured for planning that differs from the mainstream approach. PlanT uses very compact representations of objects in the scene as inputs, making for 5.4x faster inference than equivalent pixel-based methods. The authors suggest that these representations are also more similar to the way human drivers operate in complex environments. Humans reason and prioritize in terms of important objects in the scene and ignore unimportant details when making behavioral decisions (Marr, 1982; Spelke & Kinzler, 2007; Johnson, 2018). The input to PlanT's transformer encoder consists of short vector encodings of object bounding boxes and encodings of the ego-lane. The authors demonstrate that PlanT achieves state-of-the-art performance on the CARLA leaderboard challenge and even matches the performance of the rule-based planner, which PlanT is trained to imitate. For training PlanT, the ground truth information about the route and objects in the scene is extracted directly from the simulator. PlanT's training objective has two components. The primary task is to predict waypoints at four future timesteps that should match the behavior of the expert policy. A GRU unit uses the transformer outputs to predict these waypoints. At every timestep, target waypoints $\mathcal{W}_t$ for the next four timesteps are predicted and are encoded as BEV points in ego-centered coordinates: $\mathcal{W}_t = (x_w, y_w)_{w=t+1}^{t+4}$ A PID controller takes these waypoints as input to determine appropriate low-level actions (i.e., throttle, brake, and steering).

For the auxiliary task, PlanT predicts the future locations of other vehicles in the scene for the next timestep. The authors argue that the latter task is a valuable objective as it trains the ability to reason about the behavior of other traffic participants, which is crucial for planning. Also, as the main task is to predict the desired future location

of the ego vehicle, the authors expect strong synergies, incentivizing the model to encode all information required to predict the future in the output representation.

3.5.2 PID Controller

In order to match the vehicle controls to accord with the waypoints for the next timesteps, we use the same PID controllers employed by Renz et al. (2022) for lateral and longitudinal control.

PID controllers (Astrom, 1995) are a widely used form of closed loop control. These controllers manage systems with feedback loops toward desired outcomes (e.g., a target speed). This task is translated into minimizing the error between a measured quantity and the desired quantity (e.g., actual speed and desired speed). PID stands for proportional-integral-derivative controller. Adjusting the control in proportion to the error leads to oscillating behavior. Thus PID controllers incorporate the integral and the derivative of the error over time to ensure smooth and efficient control toward the desired values.

4 Metrics

This chapter introduces the metrics we use in our experiments. We first explain the online metrics that quantify driving performance (Section 4.1. In Section 4.2, we explain the standard mean average precision metric and some task-specific variations. Afterward, we define two planner-centric metrics (see Section 4.3). In Section 4.4.1 we then introduce the NuScenes Detection Score, describe how it is computed, and outline how we further adapt it to self-driving.

4.1 Online Metrics

It is clear that online evaluations provide the most reliable estimates of system-level performance. However, it it is less obvious how one should quantify the results of these evaluations. Driving performance has many facets and any online metric has to strike a balance between safety and efficiency. Here, we measure online performance using two scalarized metrics: 1) the CARLA Driving Score, the official metric for the CARLA leaderboard, and 2) the number of collisions per route. Both metrics are explained below.

Driving Score The Driving Score (DS) is a composite metric that is calculated using the route completion and the infraction score. The route completion (RC) describes the percentage of the target route the agent completed. This score is irrespective of collisions that happened along the way but is not increased if the agent drives on the wrong lane or the sidewalk. The infraction score (IS) measures collisions or violations of traffic rules committed by the agent. An infraction score of one indicates zero infractions, while lower scores indicate worse performance. In every route, the agent starts with an infraction score of one but receives multiplicative penalties for various infractions with the following coefficients:

- Collision with a pedestrian: 0.50
- Collision with a vehicle: 0.60
- Collision with a static object: 0.65
- Running a red light: 0.70

The Driving Score is then calculated by aggregating RC and IS in the following manner:

$$\mathrm{DS} = \frac{1}{N}\sum_{i}^{N} \mathrm{RC_i IS_i} \tag{4.1}$$

Here, N represents the total number of routes, RC_i, and IS_i, the route completion and infraction score for route number i. As RC_i, $IS_i \in [0,1]$, the Driving Score is also bounded by zero and one, although we present it as a percentage score in the results.

Collision Count As a second metric, we count the total number of collisions per route in which the ego vehicle is involved. This metric exclusively focuses on safety, as agents who do not progress toward the target location can achieve the perfect score. We count collisions with other participants as well as collisions with static objects. We find that the agents with the worst detector systems continually engage in bumper-to-bumper accidents but still achieve very high route completion. We want to specifically punish this reckless driving more strongly than models that stop making progress after a single initial collision. Therefore, we use the absolute number of collisions instead of a collision count normalized by distance.

4.2 Average Precision based Metrics

In this Section, we first explain the basic classification metrics (Section 4.2.1), building up to a definition of average precision (Section 4.2.2). Afterward, we introduce the task-specific variations of average precision were investigated empirically (Section 4.2.4 and 4.2.3). We calculate the metrics for a single route and class. The final values for our benchmark are computed by averaging across both.

4.2.1 Basic Classification Metrics

In order to employ classification metrics in the context of object detection, conditions must be stipulated under which a particular prediction can be evaluated as correct or incorrect (i.e., a true positive criterion). In most research on 3D object detection, a predicted bounding box is accepted as a correct prediction when it overlaps with a ground truth bounding box to a significant degree. The intersection of both volumes over the union of both volumes formally specifies the degree of overlap as a percentage. In many benchmarks, a prediction is considered correct if it has an intersection over union (IoU) value with a ground truth bounding box of over 0.5 (50%). Of course every object in the ground truth annotations should only be recognized once. In the case of multiple predictions for the same object, we thus only consider the prediction with the highest IoU score correct.

With this true positive criterion, we can now define precision and recall. The recall metric describes how many of the objects, indicated by the ground truth (GT)

annotations, have been identified by the detector. On the other hand, the precision metric reports how many of the detector's predictions correspond to GT annotations and are not false positives. Both metrics are in the range from 0 to 1.

$$\begin{aligned} \text{Precision} &= \frac{TP}{TP+FP} \\ \text{Recall} &= \frac{TP}{TP+FN} \end{aligned} \qquad \begin{aligned} TP &= \text{ True positive} \\ FP &= \text{ False positive} \\ TN &= \text{ True negative} \\ FN &= \text{ False negative} \end{aligned} \tag{4.2}$$

With the equations above, we can plot the precision-recall curve (PR curve) for a set of detections to better understand a detector's performance. To compute the PR curve, we first evaluate if the detections are correct or false positive. Afterward, we sort the detections according to the confidence values the object detector provided. We then calculate precision and recall values for every detection as the aggregate precision and recall values for all detections up to themselves, starting at the highest confidence values.

Afterward, one can plot a curve that portrays the aggregated precision values on the y-axis and the corresponding aggregated recall values on the x-axis. This curve is called the precision-recall curve. PR-curves in standard detection scenarios often show how a detector's precision is very high for the first predictions with high confidence scores but then starts to decline when more detections with lower confidence scores are considered. While the precision values can zig-zag, the recall values are monotonically increasing. We remove this zig-zag pattern to render the precision values monotonically decreasing. We do so by substituting the observed precision values at recall r with the maximum precision value at any recall $\geq r$.

$$p_{interp}(r) = \max_{\tilde{r} \geq r} p(\tilde{r})$$

where $p(\tilde{r})$ computes the precision value at recall $\tilde{r}$ for object class c. We provide an exemplary PR-curve with interpolated precision values in Figure 4.1.

4.2.2 Average Precision

Average precision (AP) is the most popular metric for evaluating 2D and 3D object detection. It represents the gold standard by which the computer vision community measures progress. The average precision for class c is defined as the area under the PR-curve for that class:

$$\text{AP} = \int_0^1 p_{\text{interp}}(r)\, dr \tag{4.3}$$

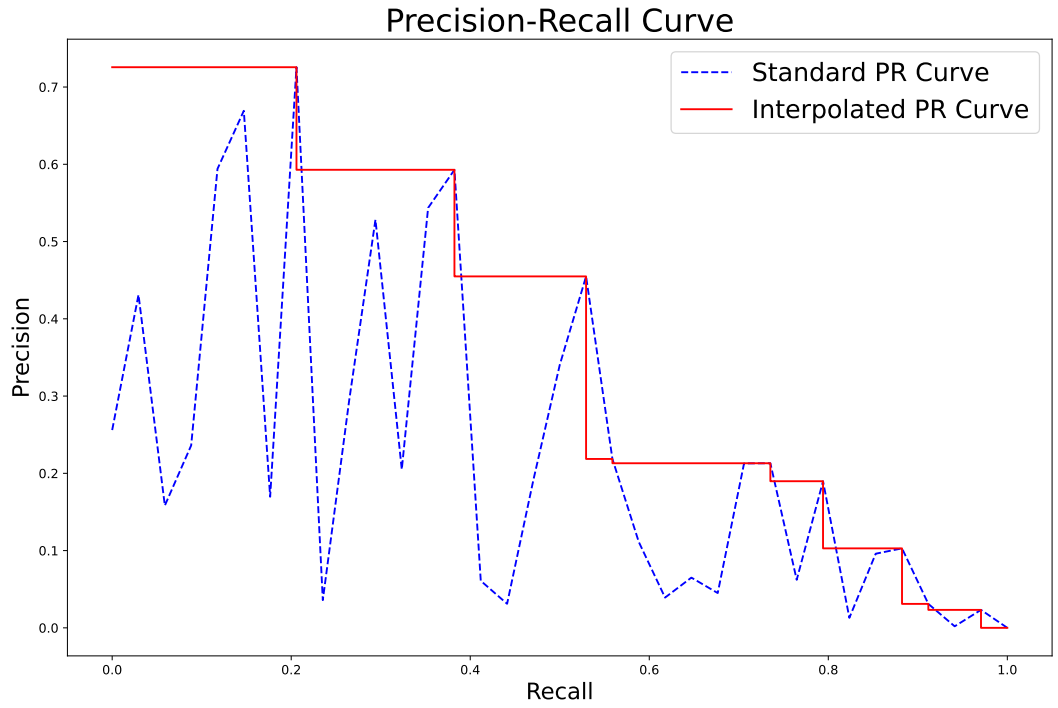

Figure 4.1: **Example of a precision-recall curve with interpolated precision values.**

Mean average precision (mAP) is the mean of the individual AP values for every object class in the data. We compute the integral using a 40-point interpolation with 40 equidistant recall values, as in the KITTI detection benchmark (Geiger et al., 2012). As all values of precision and recall are in the interval from 0 to 1, the same holds for AP and mAP. Following the KITTI protocol, we use a fixed[1] IoU threshold of 70% as the true positive criterion. Let C be the number of classes. We evaluate AP as follows:

$$\mathrm{AP} = \frac{1}{40} \sum_{r \in \{0, 0.025, 0.05, \ldots, 1\}} p_{\mathrm{interp}}(r) \tag{4.4}$$

4.2.3 Average Orientation Similarity

In the effort to adapt the mean average precision metric to the task of self-driving, multiple authors have stressed the importance of correct heading prediction (Geiger et al., 2012; Caesar et al., 2020; Sun et al., 2020). The heading angles of vehicles in the scene provide essential information for motion forecasting and behavior planning. As the original m-AP metric does not account for a notion of heading, we have

[1]This contrasts the evaluation protocol of Microsoft COCO benchmark (Lin et al., 2014), where the average over different IoU thresholds is calculated.

included the average orientation similarity metric (AOS) proposed by Geiger et al. (2012) in our analysis. AOS modifies AP in a way that weights detection according to the accuracy of their heading predictions. We compute the metric using the same 40 point interpolation:

$$AOS = \frac{1}{40} \sum_{r \in \{0, 0.025, 0.05, \ldots, 1\}} \max_{\tilde{r} \geq r} s(\tilde{r}) \tag{4.5}$$

Here, $s \in [0,1]$ at recall r is a variation on the cosine similarity:

$$s(r) = \frac{1}{|\mathcal{D}(r)|} \sum_{b_i \in \mathcal{D}(r)} \frac{1 + \cos \Delta_\theta^{(b_i)}}{2} \delta_{b_i}$$

where $\mathcal{D}(r)$ is the set of all detection at recall r, $\Delta_\theta^{(b_i)}$ is the heading error of detection i and δ_{b_i} is an indicator for true positive predictions. It is set to 1 if prediction i is correct and 0 if it is a FP.

4.2.4 Inverse Distance weighted AP

B. Deng et al. (2021) proposed another another idea for modifying AP to make it more relevant to the context of self-driving. They suggest weighting detections by their inverse distance to the ego vehicle. Closer objects are inherently more safety-critical than those far away. Thus, one should expect the weighting of AP by the inverse distance (ID-AP) to produce a metric that better reflects the ego-centric nature of detection for self-driving.

We define the inverse distance weighted TP count (ID-TP), FP count (ID-FP), and ground truth count (ID-G) as follows:

$$\text{ID-TP} = \sum_{b_i \in TP} 1/d_{b_i}^\beta \quad \text{ID-FP} = \sum_{b_i \in FP} 1/d_{b_i}^\beta \quad \text{ID-G} = \sum_{g_i \in G} 1/d_{g_i}^\beta \tag{4.6}$$

where d_x is the Euclidean distance of detection x to the center of the ego-vehicle in meter on the x-y plane, b_i are the bounding box detections, β is a hyperparameter controlling the emphasis on nearby objects, and G is the set of ground truth annotations.

We then calculate inverse distance weighted precision and recall using the definitions above:

$$\text{ID-Precision} = \frac{\text{ID-TP}}{\text{ID-TP} + \text{ID-FP}} \quad \text{ID-Recall} = \frac{\text{ID-TP}}{\text{ID-G}} \tag{4.7}$$

Both values remain within the [0, 1] interval. We then calculate ID-AP as the integral of the inverse distance weighted PR-curve using the 40-point interpolation method.

4.3 Planner-Centric Detection Metrics

Several authors have recently pursued a novel approach to detection metrics for self-driving: focusing on the planner instead of evaluating object detections directly (Philion et al., 2020b; W.-X. Li & Yang, 2023; Ivanovic & Pavone, 2022). The idea here is to isolate the detection task by measuring the impact of detection noise on downstream driving behavior. This approach naturally down-weights distant and less relevant objects in the metrics as they bear little significance to the planning task. On the other hand, (mis-)detections that affect the planner's decision-making strongly influence these metrics.

In this Section, we introduce two planner-centric metrics. To evaluate them, we first compute the planner's output for a scene given the ground truth object information. We then calculate the planner's output given the noisy output of the perception stack (detection + tracking) and analyze how the planner's predicted trajectories (i.e., the waypoints) differ. To compute the metrics, we use the PlanT planner (see Section 3.5.1) which achieves state-of-the-art results on our benchmark (see Section 5.4). The metrics we use for measuring these differences are also used in trajectory forecasting challenges like in the NuPlan benchmark (Caesar et al., 2021).

Average Displacement Error. We define the average displacement error (ADE) for a timestep as the average of the point-wise L2 distances between the predicted waypoints based on ground truth detections at frame t and the predicted waypoints based on noisy detections at the same timestep. We further define the ADE for a route as the mean of all ADEs in that route:

$$\text{ADE} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{L}\sum_{l=1}^{L}\|\tilde{w}_{t,l} - w_{t,l}\| \tag{4.8}$$

where T is the number of timesteps in the route, L is the number of predicted waypoints per timestep, w_{t_l} is the predicted waypoint number l at timestep t based on ground truth labels, and $\tilde{w}_{t_l}$ are the waypoints based on noisy detections.

Final Displacement Error We similarly define the final displacement error (FDE) as the L2 distance between the final predicted waypoint based on ground truth detections and the predicted waypoints based on noisy detections at the same timestep. Moreover, we define the FDE score for a route as the mean of all FDEs in that route:

$$\text{FDE} = \frac{1}{T}\sum_{t=1}^{T}\|\tilde{w}_{t_L} - w_{t_L}\| \tag{4.9}$$

By design, the ADE and the FDE are zero if the object detections perfectly align with ground truth annotations. We compute the final FDE and ADE scores for our benchmark by averaging scores across routes.

4.4 NuScenes Detection Score.

The NuScenes Detection Score (NDS) (Caesar et al., 2020) is a popular task-specific detection metric that uses mAP as a starting point. NDS is embedded as the primary metric on the NuScenes object detection challenge for self-driving (*NuScenes Challenge*, 2023) and has been picked up by many authors to evaluate object detection (Huang et al., 2021; Nagesh et al., 2022; Nabati & Qi, 2021; W.-X. Li & Yang, 2023). In this Section, we first explain the base metric and then introduce novel variations on the metric we tested.

4.4.1 Original Metric

NDS varies in two important ways from mAP. NDS relies on center distance in BEV for the TP-criterion instead of the standard IoU-based approach. Caesar et al. (2020) compute mAP with four different thresholds for the center distance (i.e., $0.5, 1, 2, 4$ meters) and average the resulting values to produce their final mAP score. In Section 4.2.2, we defined mAP in a way that does not average over different IoU thresholds. Here, we use a fixed center distance of 1 meter to not confound our results through the averaging. Analogous mAP, NDS only considers the TP with the smallest center distance whenever there are multiple predictions for the same annotation. By relying on the center distances, the authors decouple m-AP from object size and orientation, which they account for separately. They argue that center distance covers objects of different sizes more evenly because smaller volume objects like pedestrians can quickly achieve an IoU of zero if predictions have minor translation errors.

Besides this modified mAP value, NDS also includes explicit detection quality measures. The NDS measures detection quality in five aspects over all predictions that evaluate as true positives at a center distance threshold of two meters. The measures of detection quality are weighted equally and are, in total, given as much weight as the mAP metric in the final NDS. The five true positive metrics are given as positive scalars and include[2]:

- Average Translation Error (ATE): Euclidean center distance in BEV in meters.

[2]Our implementation is loosely based on code written by Dr. Jacob Lambert (2013)

```
def compute_ate(predictions, ground_truth):
    # Columns 0 and 1 encode the ground plane coordinates in meter.
    ate = np.mean(np.sqrt(
           (predictions[:, 0] - ground_truth[:, 0])**2 +
           (predictions[:, 1] - ground_truth[:, 1])**2))
    return ate
```

- Average Scale Error (ASE): Calculated as 1 - IOU after aligning centers and orientation.

```
def compute_ase(predictions, ground_truth):
    # This simplified IoU calculation assumes similar shapes.
    # Columns 3, 4 and 5 encode 3D object extend im meter.
    pred_vol = predictions[:, 3]*
        predictions[:, 4]*predictions[:, 5]
    gt_vol = ground_truth[:, 3]*
        ground_truth[:, 4]*ground_truth[:, 5]
    ase = np.mean(1 -
        np.minimum(pred_vol, gt_vol)/np.maximum(pred_vol, gt_vol))
    return ase
```

- Average Orientation Error (AOE): The smaller yaw angle between GT and prediction in radians.

```
def compute_aoe(predictions, ground_truth):
    # Columns 6 encodes heading angle in radians.
    err = ground_truth[:,6] - predictions[:,6]
    aoe = np.mean(np.abs((err + np.pi) % (2*np.pi) - np.pi))
    return aoe
```

- Average Velocity Error (AVE): Absolute velocity error in m/s.

```
def compute_ave(predictions, ground_truth):
    # Column 7 encodes speed in m/s.
    err = np.abs(ground_truth[:,7] - predictions[:,7])
    ave = np.mean(err)
    return ave
```

- Average Attribute Error (AAE) is defined as (1-Acc) for the prediction accuracy of additional attributes (e.g., a car being parked o not). In the experiments in Section 5, we do not predict additional attributes. We thus ignore this metric but nevertheless give equal weight to mAP and the sum of mTP-metrics.

The mean over the prediction classes is evaluated for each TP-metric to yield the mTP metrics. Afterward, all metrics, including the mAP, are merged into the NuScenes Detection Score:

$$\text{NDS} = \frac{1}{8}\left[4\text{mAP} + \sum_{\text{mTP}\in\mathbb{TP}} (1 - \min(1, \text{mTP}))\right] \tag{4.10}$$

where $\mathbb{TP}$ is the set of five true positive metrics.

4.4.2 Inverse Distance weighted NDS

Similarly to mAP, the original version of NDS does not account for an object's distance to the ego vehicle. As closer objects are, in most cases, more important for navigation than those in the distance, we also created an inverse distance-weighted version of the NDS (ID-NDS). For ID-NDS, we compute mAP according to the inverse distance weighted PR-curve described in Section 4.2.4. The detection quality metrics and the weighting remain unchanged.

4.4.3 Planning-aware NDS

In the literature on task-specific detection metrics, researchers have thus far proposed two different kinds of metrics: 1) Planner-centric metrics like the ones described in 4.3 and detection-centric metrics like mAP or NDS. As strong arguments support both approaches, we crafted a straightforward metric that fuses them. We define a planning-aware version of the NDS that incorporates the ADE.

$$\text{NDS-ADE} = 2 * \text{Z(NDS)} + \text{Z(ADE)} \tag{4.11}$$

where $Z(x)$ standardizes the metrics by subtracting the mean score over the routes and diving by the standard deviation over the routes.

5 Experiments

Our objective is to explore the relationship between the offline detection metrics described in the last Section and the online driving performance of our modular pipeline.

Ideally, we would always evaluate perception systems with regard to their effects on downstream driving. However, real-world tests of AVs are expensive and often dangerous. Even simulation-based evaluations require vastly more computational resources than offline tests and also significantly more engineering time to integrate the system under test into the simulated application environment. Thus simulation-based evaluations are often not a viable option for comparing many models or quickly testing hyper-parameter configurations in an optimization process. Offline metrics with high correlations to online performance could thus serve as proxies to reduce the cost of online testing and robust ranking from offline results.

Currently, most offline evaluations of 3D object detectors (3DODs) for self-driving are still done using the task-agnostic mAP metric. Recently various authors have proposed task-specific metrics for the evaluation of object detection (see Section 4). The problem is that these metrics differ significantly in design and are based on entirely different arguments about what aspects of detection are most relevant to self-driving. From intuition alone, it is hard to tell which metrics most predict online performance.

To inject some data into the discussion, we aim to establish quantitative results to help the research community choose the right offline metrics to compare and rank perception systems. To this end, we train 16 object detectors and evaluate their driving performance in simulation. During this driving evaluation, we log all object detections and the ground truth annotations to generate data for offline detection evaluation. This allows us to study the correlation of our detection metrics with online performance measures.

This Section discusses our setup for online evaluation and then explains how we train and configure the object detection models. Afterward, we present the results of our analysis.

5.1 Online Evaluation

5.1.1 CARLA Simulator

We base our experiments on the CARLA (CAR Learning to Act) simulator (Version:0.9.10) (Dosovitskiy et al., 2017). CARLA is an open-source, high-fidelity simulator designed for autonomous driving that many researchers in the field actively use. CARLA is highly configurable and allows us to extract bounding box object information for training the detection models. It uses a modern game engine and features reasonably realistic scene renderings. Several authors have been able to successfully transfer systems trained in CARLA to the real world (Müller et al., 2018; Gog et al., 2021; Wang et al., 2019)

5.1.2 Evaluation Benchmark

We gauge online performance for the detection models by integrating them into our modular pipeline to test driving performance. We evaluate performance on urban driving with goal-directed navigation within the CARLA simulator. Here, the agents must execute point-to-point navigation while reacting to other dynamic agents and obeying traffic rules.

We use the Longest6 benchmark introduced by Chitta et al. (2022) as an online evaluation protocol. Longest6 contains 36 ~1.5km long routes across six simulated towns within CARLA (town01-town06). Each route within Longest6 features a unique environmental condition that is composed of weather conditions (Cloudy, Wet, MidRain, WetCloudy, HardRain, SoftRain) and lighting conditions (Night, Twilight, Dawn, Morning, Noon, Sunset). We configure every route so that the ego-vehicle encounters a high density of dynamic agents in traffic. For a more detailed discussion of the benchmark, we refer the reader to Chitta et al. (2022).

5.1.3 Data Logging

While evaluating the agents on Longest6, we log detailed ground truth bounding box information about the objects in the scene and the perception stack's predictions for them. Besides the bounding boxes, we also record information about high-level navigation goals and physical sensors like IMU, GPS, and speedometer. We need this information to compute planning outputs frame-by-frame for offline evaluation of the planner-centric metrics. We sample from the simulator at a rate of two frames per second. Based on the resulting logs, we compute and average the offline detection metrics for every route. This enables us to compare the observed driving outcomes for a given route with the associated offline detection performance measures.

5.2 Detection Models

5.2.1 Training Setup

We train all detectors using the open-source LiDAR detection framework OpenPCDet (Open Point Cloud Detection) (OD-Team, 2020). OpenPCDet offers a unified platform where many state-of-the-art LiDAR detection models can be trained on custom point cloud data. We trained each of the eight architectures described in Section 3.3.3 for 72 hours on GPU nodes equipped with eight NVIDIA GeForce RTX 2080 Ti units. Our model choice focuses on architectural variety to attain a broader distribution of outcomes. Absolute model performance is less important for our purpose, and we did not optimize hyper-parameters for performance. We set all hyper-parameters used in training to the default model configurations provided by OpenPCDet. Our results should thus not be interpreted as a performance comparison between different detection architectures. We included two checkpoints per architecture in our experiments. We extract the first set of checkpoints after 36 hours of training and the second checkpoints after 72 hours of training. We indicate the different checkpoints via the number of epochs they trained for.

5.2.2 Training Data

We generate the training data for the object detectors according to the same protocol used to train the planer (Renz et al., 2022). We produce the dataset with the CARLA simulator by observing the driving behavior of an expert, rule-based algorithm with privileged access to ground truth object information. A detailed description of this expert policy can be found in Chitta et al. (2022). We evaluate the policy on 2500 unique routes through junctions with an average length of 100m and 1000 unique routes on curved highways with an average length of 400m. We randomly vary weather and lighting conditions during data generation at every frame. We log expert behavior for three different random traffic conditions on every route and sample at a rate of two frames per second, which yields 684k frames for training.

At every frame, we record data from a 360° roof-mounted LiDAR sensor positioned at 2.3m from the ground plane and located 1.3m in front of the center of the ego vehicle. The LiDAR is set to a rotation frequency of 20 Hz and records 1.200.000 points per second. We further record information from additional sensors such as an inertial measurement unit (IMU) to estimate the orientation of the ego vehicle, a GPS for localization, and a speedometer to get the current speed of the ego vehicle.

5.2.3 Detector Configurations

To curb the amount of false positive detections and attain more robust results, we set the minimum confidence threshold for object detections to 0.3. In order to further shield the planning unit from noisy detections, we only present object tracks to the

planning unit that we successfully tracked for four frames and apply non-maximum suppression with an IoU threshold of 0.2.

5.3 Metric Evaluation Protocol

After we evaluate the driving performance for all 16 detection models on the 36-route driving benchmark, we can compare the outcomes. We are interested in the varying degrees of correlation the offline metrics have to the online metrics.

We first average scores across the 36 routes to one data point per detector for every metric. For the 16 detector-wise data points, we then calculate Pearson's correlation coefficients between offline and online metrics. These correlation coefficients indicate how well the offline test can predict driving performance, given that the other parts of the modular pipeline remain fixed. Besides the correlation coefficients, we also use scatter plots to analyze the results visually.

We furthermore inspect correlations between metrics across the individual routes without taking the average over Longest6 per detector. In this route-wise setting, we can analyze 36 x 16 = 576 driven routes as data points, rendering the correlations more robust. However, route difficulties vary significantly, which influences route-wise driving performance. Thus less of the total variation between data points is explained by detection quality in this setting.

For ease of comparison and interpretation, we provide all Pearson's r correlation coefficients as absolute values in the results below. The error term-based metrics (NDS, FDE) are inversely correlated to DS and positively correlated to the number of collisions on Longest6, while score-based metrics are positively correlated to DS and inversely correlated to the number of collisions.

5.4 Results

5.4.1 Model Performance

Here, we give a high-level impression of model performances. Table 5.1 shows an overview of all detection models, their associated online performance, and a sample of offline metrics. The Table shows that the models feature different driving performance levels on Longest6 and that offline results cover a range of outcomes. We also see that, despite some correlation, metrics are not perfectly aligned, and different models shine on different metrics. Lastly, checkpoints of the same model have related performance, but their scores are dissimilar enough to be included as two data points in our analysis.

Some of the models achieve state-of-the-art Driving Score results on Longest6. Table 5.2 illustrates that our agent outperforms all previous perception-based models that

Model	DS↑	# Collisions↓	ADE↓	NDS↑	MAP↑
Centerpoint_21	**76.9**	20	24.1	76.3	54.4
Centerpoint_41	73.2	15	21.8	70.5	**63.2**
PART_A2_25	48.8	60	30.4	59.9	47.1
PART_A2_49	45.3	98	27.8	59.8	46.1
Pillarnet_21	75.6	15	21.8	**80.1**	62.9
Pillarnet_41	67.5	26	23.7	79.5	55.9
Point-Pillar_60	49.3	154	33.5	45.6	24.3
Point-Pillar_120	36.5	165	30.7	51.9	32.2
PV-RCNN_15	66.9	31	28.5	65.4	43.7
PV-RCNN_30	63.4	37	29.6	69.5	50.2
PV++_14	74.2	20	21.5	75.5	55.7
PV++_28	71.0	**8**	**20.1**	78.4	61.0
SECOND_30	72.2	17	21.2	73.5	59.6
SECOND_60	76.3	13	22.2	70.2	62.8
Voxel-RCNN_38	63.5	46	30.1	66.3	47.5
Voxel-RCNN_76	60.8	54	24.9	67.7	54.8

Table 5.1: **Detector performance overview.** The Table shows performances for all detection models on the Longest6 benchmark. The best performance on a metric is indicated in green, while the worst performance is marked in red.

reported results on the benchmark. The comparison to other perception agents is not entirely fair, as our model has access to the ground truth route and red light information from the simulator. In contrast, other agents infer red light (and route) information from onboard cameras. Equipped with the Centerpoint detector, our model slightly improves upon the rule-based agent used for data generation, even though the expert accesses ground truth object bounding boxes during inference.

5.4.2 Correlation Results

Using the results of the Longest6 evaluation shown in Figure 5.2, we compute Pearson's r correlation between the offline and the online metrics. Figure 5.3 presents the resulting correlation coefficients. The first clear result from the analysis is that our offline metrics correlate highly to driving outcomes. The NDS achieves the highest correlation values, with the standard mAP value in second place. The planner-centric metrics achieve less impressive results than their AP-based alternatives. The heading accuracy weighted version of AP (AOS) is also less correlated with online results than the NDS. NDS attains a correlation of 85% with the CARLA Driving Score on Longest6 and a correlation 90% with the number of collisions associated with a detector. The strength of the correlation indicates that offline metrics can provide decent heuristics for a detector's performance in online tests. Of course, online tests

Method	Input	DS ↑
LAV (D. Chen & Krähenbühl, 2022)	Cam. + Li.	32.74
TransFuser (Chitta et al., 2022)	Cam. + Li.	47.30
PlanT w/ perception	Cam. + Li.	**57.66**
PlanT w/perception	Cam. + Route	64.50
PlanT w/ Centerpoint_21 **(ours)**	Li. + Route + Red	**76.92**
AIM-BEV (Hanselmann et al., 2022)	Obj. + Map	45.06
Roach (Zhang et al., 2021)	Obj. + Map	55.27
PlanCNN	Obj. + Route	77.47
PlanT	Obj. + Route	**81.36**
Rule-based Expert	Obj. + Route + Act.	76.91

Table 5.2: **Results on Longest6 Benchmark**. Our best agent matches the driving performance of the rule-based expert and outperforms other published perception-based agents. Cam.=Cameras, Li.=LiDAR, Map=HD Maps, Obj.=Object bounding boxes, Act.=Actions of other vehicles, Red=Red lights. All Plan* models except ours were proposed by Renz et al. (2022).

Metric	Correlation w/ DS	Correlation w/ #Collisions
NDS (Caesar et al., 2020)	**0.852**	**0.907**
MAP (Hoiem et al., 2009)	0.805	0.903
ADE (Caesar et al., 2021)	0.784	0.770
AOS (Geiger et al., 2012)	0.742	0.894
FDE (Caesar et al., 2021)	0.703	0.653

Table 5.3: **Pearson correlation between online and offline metrics**. Absolute values are shown for clarity. Error terms (FDE,ADE) are inversely correlated to DS while scores are inversely correlated to the number of collisions.

are the gold standard for evaluating system requirements. However, given the strong relationship, offline metrics seem to be reliable tools for rougher comparisons among models and quick hypotheses testing.

This conclusion can save research groups a significant amount of time as the difference in evaluation times between the metrics are immense. While the offline metrics for a detector's performance on Longest6 can be computed on a single CPU in 5 minutes, testing a single model on the 36-route benchmark in CARLA using an NVIDIA GeForce RTX 2080 Ti takes three days.

5.4.3 NuScenes Detection Score

Table 5.3 shows that the NuScenes Detection Score correlates more to the CARLA Driving Score than the standard mAP metric (0.85 vs. 0.80). Both metrics have similar correlations to the number of collisions (0.90).

In the original NDS metric, all detection quality measures are given equal importance, and their sum is given as much weight as mAP. We conduct a small ablation study to test whether all NDS components are required for its strong correlation to driving performance. We systematically switch on and off various components and study the effects on correlation scores.

We find that all four error terms are vital, and none of them can be removed without causing a slight drop in correlation. Furthermore, we find that NDS even correlates more with DS than mAP when we remove the detection quality metrics (0.82 vs. 0.80), suggesting that center distance is a more appropriate TP-criterion than IoU. These findings give further credence to the design of the NDS. However, we find that we can further drive correlation up by reconfiguring the weighting scheme. We empirically derive a reweighted version of NDS by putting more emphasis on the translation and scale errors compared to the orientation and velocity errors while also slightly deemphasizing mAP:

$$\text{re-NDS} = \frac{1}{14}\left[6 * \text{mAP} + \sum_{\text{mTP} \in \mathbb{TP}} (1 - \min(1, \text{mTP}))\right] \tag{5.1}$$

$$\text{mTP} = \begin{bmatrix} 1 & 3.5 & 3 & 0.5 \end{bmatrix} \begin{bmatrix} mAVE \\ mATE \\ mASE \\ mAOE \end{bmatrix}$$

where mTP are the re-weighted versions of the averaged detection quality errors (velocity error, translation error, scale error, and orientation error) as defined in Section 4.4.1. The re-weighting elevates the correlation to DS from 0.85 to 0.87 and the correlation to the number of collisions from 0.91 to 0.94. This result suggests that

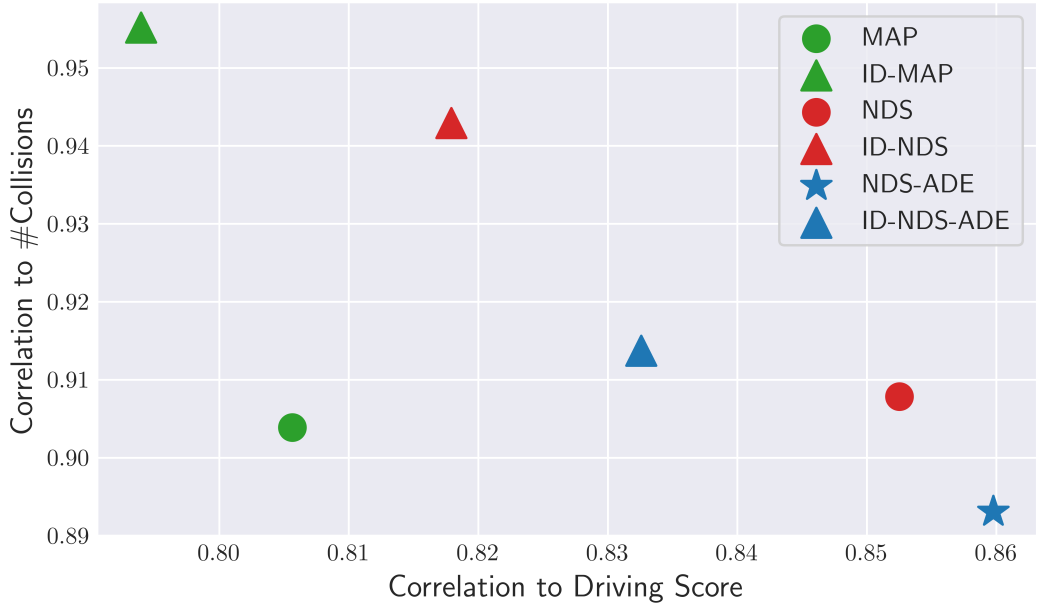

Figure 5.1: **Effects of inverse distance weighting.** Correlation scores achieved by the inverse distances weighted metrics are marked by triangles. The star symbol marks metrics that incorporate planner outputs in their design.

a precise prediction of scale and translation could matter more than exact orientation and velocity estimation for the task-specific evaluation of detection quality.

Unlike mAP, NDS also uses a velocity error term. One could thus argue that mAP covers scale, translation and orentation accuracy by its IoU based approach, and NDS only improves upon that by exploiting additional information and consecutive frames. Upon investigation, we find that after removing the velocity term, NDS is still higher correlated to DS on Longest6 than mAP (0.84 vs. 0.80). Judging from our results, NDS thus is the preferable metric, even when speed estimation does not apply.

5.4.4 Inverse Distance Weighting

We include inverse distance weighted variations for mAP, NDS, and NDS-ADE. We observe that the correlation to Driving Score drops for all three metrics. In contrast, the correlation to the number of collisions increases compared to the original metrics (see Figure 5.1). The fact that inverse distance weighting increases predictive power for collisions is to be expected. Many collisions likely occur when the perception stack misses traffic participants directly in front of the ego. ID-MAP is especially tightly connected to the collision count, with a Pearson's r coefficient of 0,955. A scatterplot of the data points behind this result is shown in Figure 5.2. We find

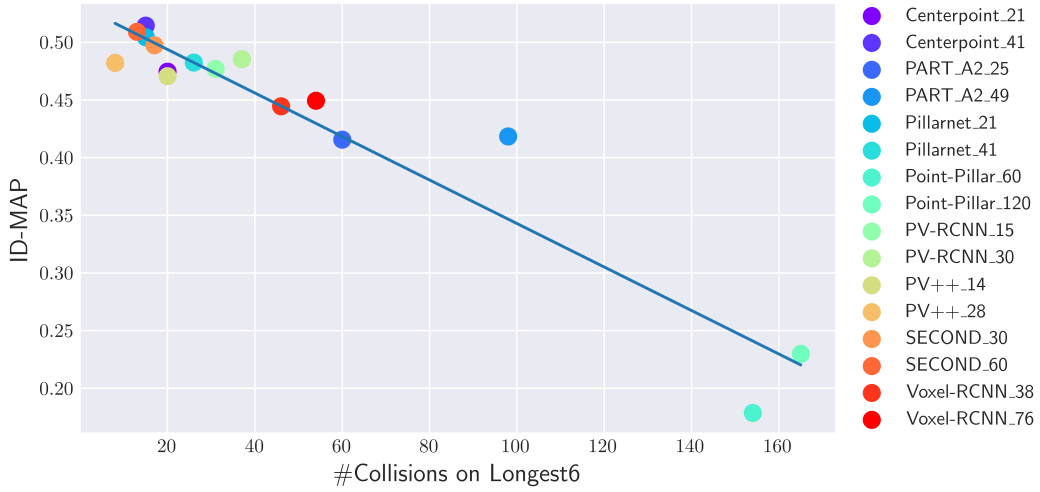

Figure 5.2: **ID-MAP vs #Collisions on Longest6.** The inverse distance weighted version of mAP correlates with the collision count with a Pearson's r of 0.955

that PV++_28 achieves the best result with 8 collisions while Point-Pillar_120 was involved in 165 collisions. We suspect that the increase in correlation to collision count is more pronounced for mAP than for NDS, as the inverse distance weighting does not affect the detection quality measure, effectively reducing the overall impact of the weighting.

On the other hand, the drop in correlation to Driving Score is more surprising. We hypothesize that due to the dense traffic conditions on Longest6, the inverse distance weighted metrics give much weight to closeby vehicles in slow traffic, which have limited implications for safety. In stop-and-go traffic, objects behind the ego or right next to the ego on opposite lanes rarely cause safety-critical situations. The drop in correlation to DS might be caused by the focus on these detections at the expense of safety-critical objects at a distance (e.g., a car about to cross the same intersection from another side.).

5.4.5 Planner-Centric Metrics

Our results show that the ADE marks a better indicator for driving performance than the FDE. While the planner-centric ADE correlates to Driving Score and collision count (0.784 & 0.770, respectively), these correlations pale in comparison to those of the average precision-based metrics (see Figure 5.3). The significant discrepancies in correlation scores we observe between the approaches, therefore, contra-indicate a reliance on planner-centric metrics alone when evaluating object detection for self-driving.

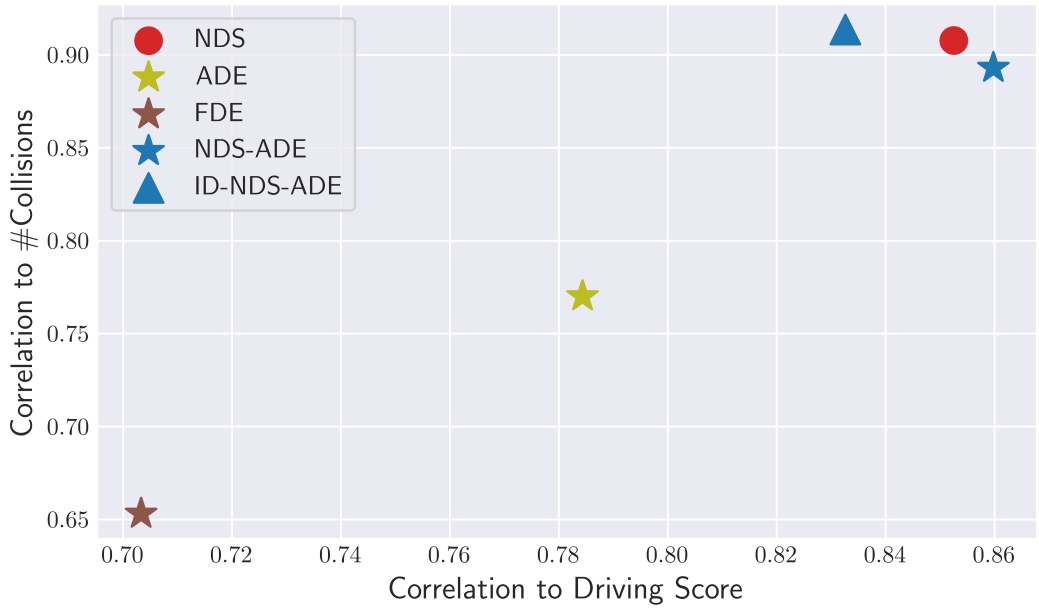

Figure 5.3: **Correlation analysis for planner-centric metrics.** The plot marks correlation coefficients for planner-centric metrics with a star symbol. The inverse distance weighted version of NDS-ADE is marked as a trangle. NDS is added as reference.

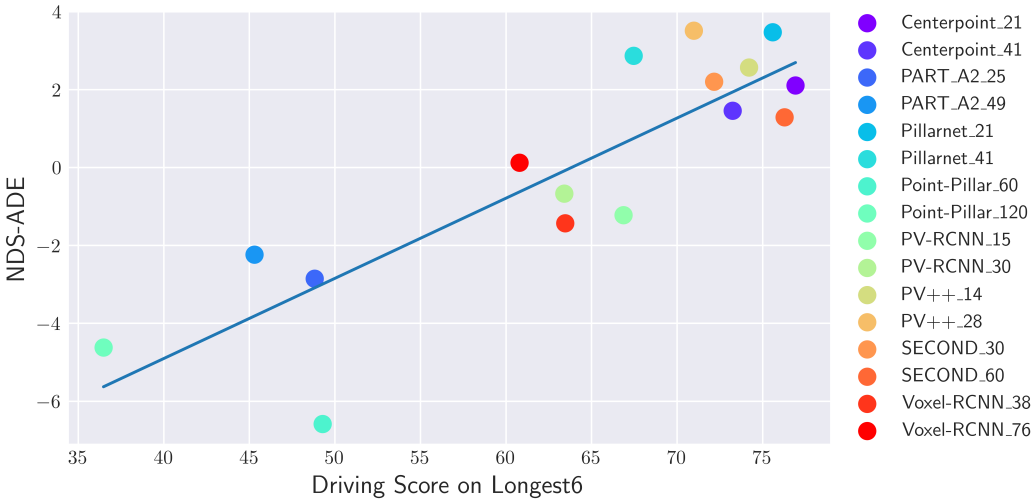

Figure 5.4: **NDS-ADE vs Driving Score on Longest6.** The planning aware modification of the NDS (NDS-ADE) achieves the highest correlation to DS in our detector-level analysis.

Figure 5.5: **Summary of correlation results.** The plot shows all detector-level correlation results on the Longest6 benchmark. The star symbol indicates planner-related metrics and, the triangle marks metrics with inverse distance weighting.

This being said, we want to highlight, that NDS-ADE, our planning aware modification of the NDS (see Section 4.4.3), achieves the highest correlation to DS (0.86) of all metrics included in the detector-wise analysis. We include a scatterplot illustrating the relationship in Figure 5.4. This result suggests that even though planner-centric approaches should not be relied on exclusively, they still seem able to provide additional information to metrics like the NDS. To our knowledge, no research group has yet combined planner-based and AP-based metrics for detection evaluation. As we base our ADE-NDS metric on a very straightforward procedure, we suspect that there is value in exploring this approach further.

5.4.6 Summary and Comparison to Route-Level Analysis

We compactly present all detector-level correlation results in Figure 5.5. Key results illustrated in the figure include the superior correlation associated with NDS over mAP, the drop in correlation to DS caused by inverse distance weighting, and the lower correlation results for the purely planner-centric metrics. All of these insights are confirmed by the route-level analysis we conduct. Here, the metrics are not averaged per detector, but scores for all 576 routes are used to compute the correlations. The route-level correlations to Driving Score displayed in Figure 5.6 underline the conclusions stated above and further support the utility of the NDS.

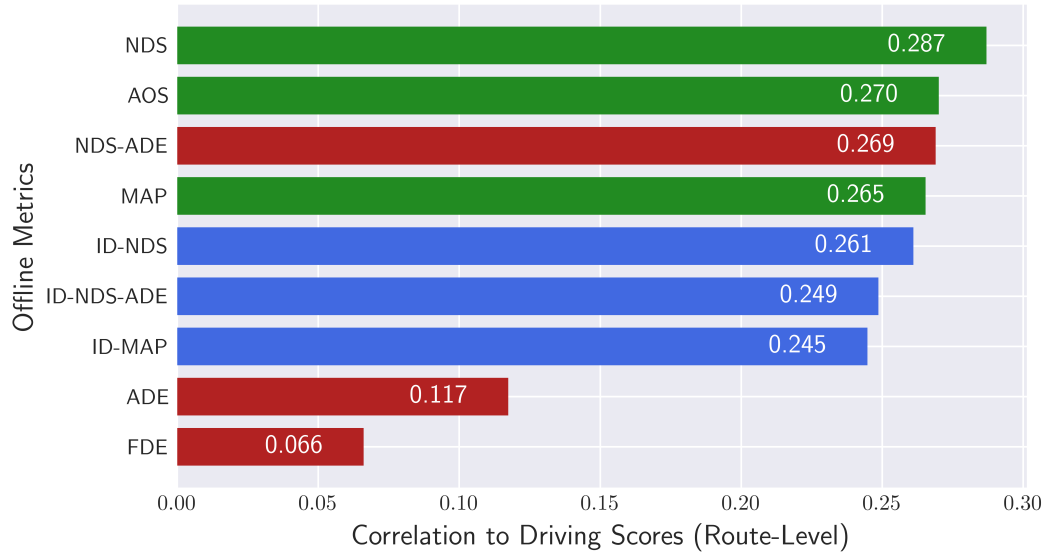

Figure 5.6: **Correlations to Driving Score on route-level.** In the route-level analysis, we do not use the detector-wise averages but compute the correlations using all 576 routes simultaneously. Standard mAP-based metrics are marked in green, inverse distance weighted metrics in blue and all metrics that incorporate planner information are marked red.

Interestingly, our planning-aware modification of NDS is associated with a lower correlation than the standard NDS in these results.

6 Conclusion

In this thesis, we demonstrate that common metrics for 3D object detection are highly correlated with online driving performance. These findings contrast with prior results that indicated that standard offline measures of steering accuracy are only weakly related to online outcomes (Haq et al., 2021; Codevilla et al., 2018). Online tests are often associated with immense costs and safety implications. Offline evaluations offer cost-efficient heuristics for benchmarking object detection models. We present the first quantitative examination comparing different detection metrics with regard to their correlations to online performance. Our extensive evaluation shows that the NuScenes Detection Score is more predictive of closed-loop outcomes than the standard mean average precision score. Our data thus supports the trend toward utilizing task-specific detection metrics when benchmarking 3D object detection for self-driving. Having observed weaker correlations with the other task-specific metrics, we especially recommend the NuScenes detection score. While we find the standard mAP score to yield reasonable correlation, our results invoke skepticism regarding detection benchmarks that exclusively rely on planner-centric approaches (Guo et al., 2020b) or use a strong focus on heading accuracy (Sun et al., 2020).

While our results discourage exclusively planner-centric approaches, we attain strong results for our fused metric. To our knowledge, no metric that combines detection-centric and planner-centric approaches has yet been proposed, yet our results indicate that there is potential for this direction. There are additional research directions that appear interesting based on our results but were out of this project's scope. One of them is an expanded analysis encompassing more metrics and investigating more configurations, such as the averaging AP over different thresholds of the TP-criterion. A further compelling idea is the application of our methodology for evaluating metrics to other tasks in the self-driving pipeline, like object tracking and motion forecasting.

There are important limitations that constrain the degree to which one can generalize from our results, three of the most important which we want to highlight. First, we base all our experiments on the same neural planning architecture. In our experiments, PlanT acts as a mediator between the detection performance and the driving outcomes. Different planners might focus on other object cues for motion forecasting and behavior planning. They might further portray contrasting reactions to perception noise. It is thus unclear how strongly our conclusions apply to experiments utilizing a different planning model.

Second, we rely on the CARLA simulator for evaluation. The complexity of the challenges on Longest6 is limited compared to the real world, and using any simulator naturally raises questions about transferability of the results to real-world applications. However, here we particularly want to highlight the limitations of the Driving Score metric. The metric is a relatively simple heuristic for evaluating overall driving performance, and more accurate online metrics might yield different correlation outcomes. The fact that we use the Driving Score as our primary online metric may strongly influence our conclusions.

Third, we primarily focus on the Pearson correlation coefficient to study the relationship between online and offline measures. While correlation gives valuable information about the connection between two variables, a more fine-grained study of the connection between the metrics could provide a deeper understanding of their interactions.

To summarise, our results indicate that the NuScenes Detection Score is a more relevant metric than the IoU-based average precision. We are excited to see if other groups can replicate this finding using different planners and online metrics. When more research confirms our findings, we recommend that the community embrace the NDS more broadly.

References

Albus, J. S. (2002). 4d/rcs: a reference model architecture for intelligent unmanned ground vehicles. In *Unmanned ground vehicle technology iv* (Vol. 4715, pp. 303–310).

Aradi, S. (2020). Survey of deep reinforcement learning for motion planning of autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, *23*(2), 740–759.

Arnold, E., Al-Jarrah, O. Y., Dianati, M., Fallah, S., Oxtoby, D., & Mouzakitis, A. (2019). A survey on 3d object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems*, *20*(10), 3782–3795.

Astrom, K. J. (1995). Pid controllers: theory, design, and tuning. *The International Society of Measurement and Control*.

Bansal, A., Singh, J., Verucchi, M., Caccamo, M., & Sha, L. (2021). Risk ranked recall: Collision safety metric for object detection systems in autonomous vehicles. In *2021 10th mediterranean conference on embedded computing (meco)* (pp. 1–4).

Bansal, M., Krizhevsky, A., & Ogale, A. (2018). Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*.

Bbc. (2018). `https://www.bbc.com/news/technology-44713298`. (Accessed: 2023-06-20)

Beltrán, J., Guindel, C., Moreno, F. M., Cruzado, D., Garcia, F., & De La Escalera, A. (2018). Birdnet: a 3d object detection framework from lidar information. In *2018 21st international conference on intelligent transportation systems (itsc)* (pp. 3517–3523).

Benekohal, R. F., & Treiterer, J. (1988). Carsim: Car-following model for simulation of traffic in normal and stop-and-go conditions. *Transportation research record*, *1194*, 99–111.

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., ... Beijbom, O. (2020). nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 11621–11631).

Caesar, H., Kabzan, J., Tan, K. S., Fong, W. K., Wolff, E., Lang, A., ... Omari, S. (2021). nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*.

Carfrae, J. (2010). *An automated adventure at the wheel of a driverless bmw – the national.* `Thenational.ae`. (Accessed: 2023-06-20)

Carla leaderboard. (n.d.). `leaderboard.carla.org`. (2020)

Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., ... others (2019). Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 8748–8757).

Chen, D., & Krähenbühl, P. (2022). Learning from all vehicles. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 17222–17231).

Chen, D., Zhou, B., Koltun, V., & Krähenbühl, P. (2020). Learning by cheating. In *Conference on robot learning* (pp. 66–75).

Chen, F., Wang, X., Zhao, Y., Lv, S., & Niu, X. (2022). Visual object tracking: A survey. *Computer Vision and Image Understanding*, *222*, 103508.

Chen, Q., Sun, L., Wang, Z., Jia, K., & Yuille, A. (2020). Object as hotspots: An anchor-free 3d object detection approach via firing of hotspots. In *Computer vision–eccv 2020: 16th european conference, glasgow, uk, august 23–28, 2020, proceedings, part xxi 16* (pp. 68–84).

Chitta, K., Prakash, A., Jaeger, B., Yu, Z., Renz, K., & Geiger, A. (2022). Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Choueka, Y., Cohen, M., Dueck, J., Fraenkel, A. S., & Slae, M. (1971). Full text document retrieval: Hebrew legal texts (report on the first phase of the responsa retrieval project). In *Proceedings of the 1971 international acm sigir conference on information storage and retrieval* (pp. 61–79).

Ciaparrone, G., Sánchez, F. L., Tabik, S., Troiano, L., Tagliaferri, R., & Herrera, F. (2020). Deep learning in video multi-object tracking: A survey. *Neurocomputing*, *381*, 61–88.

Codevilla, F., Lopez, A. M., Koltun, V., & Dosovitskiy, A. (2018). On offline evaluation of vision-based driving models. In *Proceedings of the european conference on computer vision (eccv)* (pp. 236–251).

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*, 273–297.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 ieee computer society conference on computer vision and pattern recognition (cvpr'05)* (Vol. 1, pp. 886–893).

Deng, B., Qi, C. R., Najibi, M., Funkhouser, T., Zhou, Y., & Anguelov, D. (2021). Revisiting 3d object detection from an egocentric perspective. *Advances in Neural Information Processing Systems*, *34*, 26066–26079.

Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., & Li, H. (2021). Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 35, pp. 1201–1209).

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). Carla: An open urban driving simulator. In *Conference on robot learning* (pp. 1–16).

Drobnitzky, M., Friederich, J., Egger, B., & Zschech, P. (2022). Survey and systematization of 3d object detection models and methods. *arXiv preprint arXiv:2201.09354*.

Elhousni, M., & Huang, X. (2020). A survey on 3d lidar localization for autonomous vehicles. In *2020 ieee intelligent vehicles symposium (iv)* (pp. 1879–1884).

Everingham, M., Zisserman, A., Williams, C. K., Van Gool, L., Allan, M., Bishop, C. M., ... others (2006). The 2005 pascal visual object classes challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment: First pascal machine learning challenges workshop, mlcw 2005, southampton, uk, april 11-13, 2005, revised selected papers* (pp. 117–176).

Fan, H., Zhu, F., Liu, C., Zhang, L., Zhuang, L., Li, D., ... Kong, Q. (2018). Baidu apollo em motion planner. *arXiv preprint arXiv:1807.08048*.

Fan, J., Zheng, P., & Li, S. (2022). Vision-based holistic scene understanding towards proactive human–robot collaboration. *Robotics and Computer-Integrated Manufacturing*, *75*, 102304.

Feng, D., Haase-Schütz, C., Rosenbaum, L., Hertlein, H., Glaeser, C., Timm, F., ... Dietmayer, K. (2020). Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, *22*(3), 1341–1360.

Fernandes, D., Silva, A., Névoa, R., Simões, C., Gonzalez, D., Guevara, M., ... Melo-Pinto, P. (2021). Point-cloud based 3d object detection and classification methods for self-driving applications: A survey and taxonomy. *Information Fusion*, *68*, 161–191.

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 ieee conference on computer vision and pattern recognition* (pp. 3354–3361).

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the ieee international conference on computer vision* (pp. 1440–1448).

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2015). Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, *38*(1), 142–158.

Gog, I., Kalra, S., Schafhalter, P., Wright, M. A., Gonzalez, J. E., & Stoica, I. (2021). Pylot: A modular platform for exploring latency-accuracy tradeoffs in autonomous

vehicles. In *2021 ieee international conference on robotics and automation (icra)* (pp. 8806–8813).

Guo, Y., Caesar, H., Beijbom, O., Philion, J., & Fidler, S. (2020a). The efficacy of neural planning metrics: A meta-analysis of pkl on nuscenes. *arXiv preprint arXiv:2010.09350*.

Guo, Y., Caesar, H., Beijbom, O., Philion, J., & Fidler, S. (2020b). The efficacy of neural planning metrics: A meta-analysis of pkl on nuscenes. *arXiv preprint arXiv:2010.09350*.

Gupta, A., Anpalagan, A., Guan, L., & Khwaja, A. S. (2021). Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array*, *10*, 100057.

Hanselmann, N., Renz, K., Chitta, K., Bhattacharyya, A., & Geiger, A. (2022). King: Generating safety-critical driving scenarios for robust imitation via kinematics gradients. In *Computer vision–eccv 2022: 17th european conference, tel aviv, israel, october 23–27, 2022, proceedings, part xxxviii* (pp. 335–352).

Haq, F. U., Shin, D., Nejati, S., & Briand, L. (2021). Can offline testing of deep neural networks replace their online testing? a case study of automated driving systems. *Empirical Software Engineering*, *26*(5), 90.

Hoiem, D., Divvala, S. K., & Hays, J. H. (2009). Pascal voc 2008 challenge. *World Literature Today*, *24*.

Huang, J., Huang, G., Zhu, Z., Ye, Y., & Du, D. (2021). Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. *arXiv preprint arXiv:2112.11790*.

Ivanovic, B., & Pavone, M. (2022). Injecting planning-awareness into prediction and detection evaluation. In *2022 ieee intelligent vehicles symposium (iv)* (pp. 821–828).

Johnson, S. P. (2018). Object perception. In *Oxford research encyclopedia of psychology.*

Kaur, P., Taghavi, S., Tian, Z., & Shi, W. (2021). A survey on simulators for testing self-driving cars. In *2021 fourth international conference on connected and autonomous driving (metrocad)* (pp. 62–70).

Kesten, R., Usman, M., Houston, J., Pandya, T., Nadhamuni, K., Ferreira, A., ... others (2019). Lyft level 5 av dataset 2019. *urlhttps://level5. lyft. com/dataset*, *1*, 3.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, *2*(1-2), 83–97.

Lambert, J. (2013). *Project 3d_lidar_detection_evaluation.* `github.com/jacoblambert/3d_lidar_detection_evaluation`. GitHub. (Accessed: 2023-05-20)

Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., & Beijbom, O. (2019). Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 12697–12705).

Li, W.-X., & Yang, X. (2023). Transcendental idealism of planner: Evaluating perception from planning perspective for autonomous driving. *arXiv preprint arXiv:2306.07276*.

Li, Y., Ma, L., Zhong, Z., Liu, F., Chapman, M. A., Cao, D., & Li, J. (2020). Deep learning for lidar point clouds in autonomous driving: A review. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(8), 3412–3432.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2117–2125).

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *Computer vision–eccv 2014: 13th european conference, zurich, switzerland, september 6-12, 2014, proceedings, part v 13* (pp. 740–755).

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In *Computer vision–eccv 2016: 14th european conference, amsterdam, the netherlands, october 11–14, 2016, proceedings, part i 14* (pp. 21–37).

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the seventh ieee international conference on computer vision* (Vol. 2, pp. 1150–1157).

Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., & Kim, T.-K. (2021). Multiple object tracking: A literature review. *Artificial intelligence*, *293*, 103448.

Mao, J., Shi, S., Wang, X., & Li, H. (2023). 3d object detection for autonomous driving: A comprehensive survey. *International Journal of Computer Vision*, 1–55.

Mao, J., Xue, Y., Niu, M., Bai, H., Feng, J., Liang, X., ... Xu, C. (2021). Voxel transformer for 3d object detection. In *Proceedings of the ieee/cvf international conference on computer vision* (pp. 3164–3173).

Marr, D. (1982). Computational investigation into the human representation and processing of visual information. *Freeman*.

Matsugu, M., Mori, K., Mitari, Y., & Kaneda, Y. (2003). Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, *16*(5-6), 555–559.

Mercedes-benz drive pilot. (2023). `group.mercedes-benz.com/innovation/product-innovation/autonomous-driving/drive-pilot-nevada.html`. (Accessed: 2023-06-20)

Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., ... others (2008). Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, *25*(9), 569–597.

Müller, M., Dosovitskiy, A., Ghanem, B., & Koltun, V. (2018). Driving policy transfer via modularity and abstraction. *arXiv preprint arXiv:1804.09364*.

Nabati, R., & Qi, H. (2021). Centerfusion: Center-based radar and camera fusion for 3d object detection. In *Proceedings of the ieee/cvf winter conference on applications of computer vision* (pp. 1527–1536).

Nagesh, S., Baig, A., Srinivasan, S., Rangesh, A., & Trivedi, M. (2022). Structure aware and class balanced 3d object detection on nuscenes dataset. *arXiv preprint arXiv:2205.12519*.

Niel, D. (2012). *The driverless car is coming. and we all should be glad it is. – wall steet journal.* `wsj.com`. (Accessed: 2023-06-20)

Nuscenes challenge. (2023). `https://www.nuscenes.org/object-detection`. (Accessed: 2023-06-20)

OD-Team. (2020). *Openpcdet: An open-source toolbox for 3d object detection from point clouds.* `github.com/open-mmlab/OpenPCDet`. (Accessed: 2023-02-20)

Philion, J., Kar, A., & Fidler, S. (2020a). Implementing planning kl-divergence. In *Computer vision–eccv 2020 workshops: Glasgow, uk, august 23–28, 2020, proceedings, part vi 16* (pp. 11–18).

Philion, J., Kar, A., & Fidler, S. (2020b). Learning to evaluate perception models using planner-centric metrics. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 14055–14064).

Pomerleau, D. A. (1988). Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, *1*.

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 652–660).

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, *30*.

Qian, R., Lai, X., & Li, X. (2022). 3d object detection for autonomous driving: a survey. *Pattern Recognition*, *130*, 108796.

Ravindran, R., Santora, M. J., & Jamali, M. M. (2020). Multi-object detection and tracking, based on dnn, for autonomous vehicles: A review. *IEEE Sensors Journal*, *21*(5), 5668–5677.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 779–788).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, *28*.

Renz, K., Chitta, K., Mercea, O.-B., Koepke, A., Akata, Z., & Geiger, A. (2022). Plant: Explainable planning transformers via object-level representations. *arXiv preprint arXiv:2210.14222*.

Rong, G., Shin, B. H., Tabatabaee, H., Lu, Q., Lemke, S., Možeiko, M., ... others (2020). Lgsvl simulator: A high fidelity simulator for autonomous driving. In *2020 ieee 23rd international conference on intelligent transportation systems (itsc)* (pp. 1–6).

Shao, H., Wang, L., Chen, R., Li, H., & Liu, Y. (2023). Safety-enhanced autonomous driving using interpretable sensor fusion transformer. In *Conference on robot learning* (pp. 726–737).

Shi, G., Li, R., & Ma, C. (2022). Pillarnet: Real-time and high-performance pillar-based 3d object detection. In *Computer vision–eccv 2022: 17th european conference, tel aviv, israel, october 23–27, 2022, proceedings, part x* (pp. 35–52).

Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., & Li, H. (2020). Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 10529–10538).

Shi, S., Jiang, L., Deng, J., Wang, Z., Guo, C., Shi, J., ... Li, H. (2023). Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. *International Journal of Computer Vision*, *131*(2), 531–551.

Shi, S., Wang, X., & Li, H. (2019). Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 770–779).

Shi, S., Wang, Z., Shi, J., Wang, X., & Li, H. (2020). From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE transactions on pattern analysis and machine intelligence*, *43*(8), 2647–2664.

Spelke, E. S., & Kinzler, K. D. (2007). Core knowledge. *Developmental science*, *10*(1), 89–96.

Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., ... Caine, B. (2020). Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 2446–2454).

Thorpe, C., Hebert, M., Kanade, T., & Shafer, S. (1987). Vision and navigation for the carnegie-mellon navlab. *Annual Review of Computer Science*, *2*(1), 521–556.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., ... others (2006). Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, *23*(9), 661–692.

Tideman, M., & Van Noort, M. (2013). A simulation tool suite for developing connected vehicle systems. In *2013 ieee intelligent vehicles symposium (iv)* (pp. 713–718).

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M., ... others (2008). Autonomous driving in urban environments: Boss and the urban challenge. *Journal of field Robotics*, *25*(8), 425–466.

Vitelli, M., Chang, Y., Ye, Y., Ferreira, A., Wołczyk, M., Osiński, B., ... others (2022). Safetynet: Safe planning for real-world self-driving vehicles using machine-learned policies. In *2022 international conference on robotics and automation (icra)* (pp. 897–904).

Wang, D., Devin, C., Cai, Q.-Z., Yu, F., & Darrell, T. (2019). Deep object-centric policies for autonomous driving. In *2019 international conference on robotics and automation (icra)* (pp. 8853–8859).

World bank. (2023). `data.worldbank.org/indicator/SH.STA.TRAF.P5`. (Accessed: 2023-06-20)

Xu, Y., Zhou, X., Chen, S., & Li, F. (2019). Deep learning for multiple object tracking: a survey. *IET Computer Vision*, *13*(4), 355–368.

Yan, Y., Mao, Y., & Li, B. (2018). Second: Sparsely embedded convolutional detection. *Sensors*, *18*(10), 3337.

Yin, T., Zhou, X., & Krahenbuhl, P. (2021). Center-based 3d object detection and tracking. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 11784–11793).

Zaidi, S. S. A., Ansari, M. S., Aslam, A., Kanwal, N., Asghar, M., & Lee, B. (2022). A survey of modern deep learning based object detection models. *Digital Signal Processing*, 103514.

Zeng, W., Luo, W., Suo, S., Sadat, A., Yang, B., Casas, S., & Urtasun, R. (n.d.). End-to-end interpretable neural motion planner. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 8660–8669).

Zeng, W., Luo, W., Suo, S., Sadat, A., Yang, B., Casas, S., & Urtasun, R. (2019). End-to-end interpretable neural motion planner. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition* (pp. 8660–8669).

Zhang, Z., Liniger, A., Dai, D., Yu, F., & Van Gool, L. (2021). End-to-end urban driving by imitating a reinforcement learning coach. In *Proceedings of the ieee/cvf international conference on computer vision* (pp. 15222–15232).

Zhou, Y., & Tuzel, O. (2018). Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4490–4499).

Zhu, Y., Li, X., Liu, C., Zolfaghari, M., Xiong, Y., Wu, C., ... Li, M. (2020). A comprehensive study of deep video action recognition. *arXiv preprint arXiv:2012.06567*.