Evaluation of 3D Object Detection Models in Autonomous Driving Software

Callum Turnbull

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2024

Abstract

3D object detection is an important task in an autonomous driving system. This work evaluates the 3D object detection capabilities of Autoware, an open source autonomous driving software stack. The focus is on evaluation of a LiDAR-based detector. Experimental data collection is carried out using a real autonomous vehicle in a typical urban European city environment, and this dataset is used in the evaluation. Additionally, evaluation is carried out on the Waymo Open dataset, a standard large open source dataset in the field. It is found that the selected detector performs reasonably well on the small experimental dataset assessing a particular scenario, but surprisingly performs poorly on the Waymo Open dataset. This is attributed to a domain gap from the training data.

Research Ethics Approval

This project obtained approval from the Informatics Research Ethics committee.
Ethics application number: 552416
Date when approval was obtained: 2024-08-06
The participants' information sheet is included in the appendix.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Callum Turnbull*)

Acknowledgements

I would like to thank my supervisor, Professor Subramanian Ramamoorthy, as well as Alejandro Bordallo, Hector Cruz, Servinar Cheema, and Jim Duffin from the University of Edinburgh AV team, for their invaluable support and advice throughout the project. I would also like to thank fellow Masters student Julius Schulte for his collaboration on shared logistics for both of our projects.

Table of Contents

List of Figures

Acronyms

2D 2 dimensional

3D 3 dimensional

AOE Average Orientation Error

AOS Average Orientation Similarity

AP Average Precision

APH Average Precision weighted by Heading

ASE Average Scale Error

AV Autonomous Vehicle

BEV Bird's Eye View

CNN Convolutional Neural Network

FN False Negative

FP False Positive

HDMap High-Definition Map

IoU Intersection over Union

JSON JavaScript Object Notation

LiDAR Light Detection and Ranging

mAP Mean Average Precision

NMS Non-Maximum Suppression

NDS nuScenes Detection Score

RADAR Radio Detection and Ranging

RGB Red-Green-Blue

RoI Region of Interest

ROS Robot Operating System

SAE Society of Automotive Engineers

TN True Negative

TP True Positive

VFE Voxel Feature Encoder

VRU Vulnerable Road User

Chapter 1

Introduction

1.1 Overview

The purpose of this work is to investigate the 3D object detection capabilities of Autoware, an open source autonomous driving software stack. This section summarises some of the background knowledge for this task, and outlines the research question and project objectives.

1.2 What is Autonomous Driving?

Autonomous driving systems are developed in order to drive road vehicles, including cars, without a human directly controlling the vehicle. Such systems are expected to benefit society in various ways, such as reduction in air pollution and energy consumption for travel, prevention of road traffic accidents, and increased transport accessibility for people of limited mobility. [1]

SAE International is a standards organization related to the automotive industry. Their J3016 standard [2] aims to classify various levels of autonomy in autonomous driving systems. The standard outlines 6 levels of automation, from 0 to 5. Level 0 is a vehicle with no automation. Levels 1 and 2 encompass ”assistive driving technologies”, which automate a particular aspect of driving, such as adaptive cruise control, lane-keeping assistance, or automated parking. Level 1 systems provide a single assistive technology, whereas Level 2 systems are able to automate several of these tasks simultaneously. However, by this stage the operation of the vehicle is still the responsibility of the human driver. Level 2 systems are already present in many recent models from leading automotive manufacturers.

Level 3 automation is an important boundary where the responsibility for monitoring the operation of the vehicle is shifted towards the automated system. The vehicle is able to drive itself fully autonomously in very specific scenarios (for example, below a certain speed, only on straight and clear roads), but the human driver is required to remain attentive and re-assume control of the vehicle when the system requests it. Companies such as BMW, Mercedes-Benz, and Honda have recently included Level 3 systems in commercially available models, with geographic restrictions due to legal requirements [3–5]. However, Level 3 technologies present a safety issue regarding the human driver takeover system. At this Level the driver must take control of the vehicle when requested, which has the potential to cause accidents if the driver is inattentive and unable to assume control in good time. This may be contributing to the relatively slow adoption of this Level of automation when compared to Level 2 [6].

Finally, systems at Level 4 and 5 automate the entire driving process without human interaction. Level 4 systems will have some restrictions on driving scenarios they are able to operate in. Human driver takeover may still be required in busy urban areas for example, but the system is able to safely stop the vehicle and prevent accidents in the event of a failed human takeover. At the highest level, Level 5 systems represent a fully autonomous vehicle. There do exist a small number of Level 4 technologies in real-world operation, perhaps the most prominent of which is Waymo's autonomous taxi service. This service currently operates in three locations in the United States [7, 8]: Phoenix, Arizona; San Francisco, California; and Los Angeles, California.

1.3 Components of an Autonomous Driving System

Autonomous driving systems are typically comprised of four main components: sensing, perception, planning, and control. The sensing system makes use of sensors on the vehicle to continuously acquire data about the driving environment. Common sensors for perception include cameras, Light Detection and Ranging (LiDAR), and Radio Detection and Ranging (radar), among others. The perception system processes the sensor data and extracts higher-order information from it, such as detection and tracking of objects, and scene understanding (interpreting the driving context, spatial relationships, and dynamics of the entire environment). The planning system uses the perception data to make decisions about what actions the vehicle should take, which are then performed by the vehicle via the control system. This work will focus on the perception system.

1.4 3D Object Detection

The perception system is critical to the operation of autonomous driving systems, as the derived data representations directly inform the downstream decision making tasks. One of the most important tasks the perception system is responsible for is 3D object detection. This is defined as identifying the size and pose (position and orientation) of objects around the vehicle in 3D space. An object detection process produces "bounding boxes" [9, 10] which are 3D rectangular boxes (i.e. cuboids) that indicate where an object has been detected, and the limits of the objects' dimensions (i.e. the entire object is contained within the box). Often a detection process will also provide a classification for the detected object, such as car, pedestrian, bicycle, etc.

The leading 3D object detection methods from literature almost all make use of LiDAR point cloud data, and some perform sensor fusion to combine this with camera image data. This work will focus on LiDAR-based methods only.

1.5 3D Object Detection Evaluation

The standard metric for evaluation in 3D object detection is average precision (AP) [9, 11]. This is derived from metrics commonly used in evaluating classification problems. Figure 1.1 illustrates a confusion matrix, which defines how predictions interact with labels to produce True Positives, False Positives, True Negatives, and False Negatives, which are often abbreviated as TP, FP, TN, and FN respectively. The total count of these for a classification task allow calculating Precision and Recall, with the following formulae (also illustrated in Figure 1.1):

$$\text{Precision} = \frac{TP}{TP+FP}, \quad \text{Recall} = \frac{TP}{TP+FN}$$

Average Precision is defined as the area under a Precision-Recall curve (with precision plotted against recall) [9, 11]. Figure 1.2 illustrates a typical precision-recall curve. In order to find a curve for the predictions, a threshold is established for when predictions should be included or not, often using the confidence score assigned to a prediction. If the confidence is above the threshold the prediction is included in the calculation, and otherwise it isn't. The precision and recall are calculated at each threshold, and the threshold reduces until a defined end point is reached, or all the predictions are included. The interpolated precision can be used in place of the actual precision values to reduce the "wiggles" in the curve. This means if there is a precision

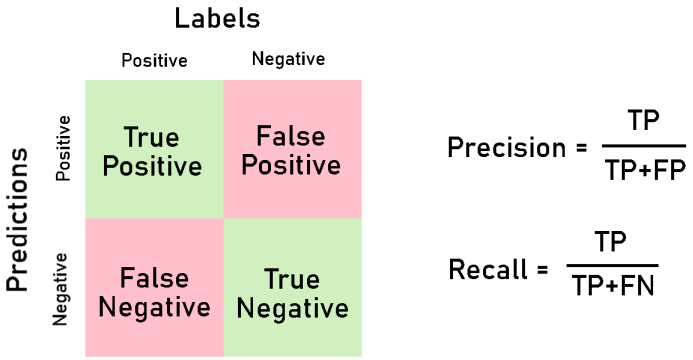

Figure 1.1: Confusion matrix illustrating prediction and label interactions.

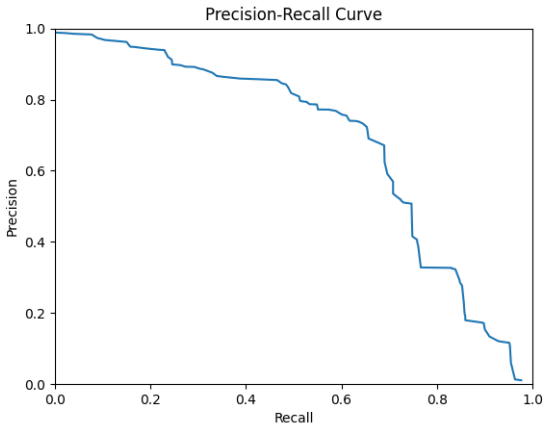

Figure 1.2: An example precision-recall curve using randomly generated data.

value which is greater at a higher recall, this value is used instead of the plotted precision at the current recall.

When applied to 3D object detection, average precision is calculated per class, and the average of the class AP values gives the mean average precision (mAP). Ground truth labels are considered to be the true labels, and there is no concept of a false label (it doesn't make sense to label where an object isn't). Then, the predictions must be matched up to the ground truths. The metric used to do this is Intersection over Union (IoU) [9, 11], which measures the proportion of the volumes of two bounding boxes which are shared between them (how much they overlap). For bounding box volumes V_A and V_B, the IoU is given by:

$$IoU = \frac{V_A \cap V_B}{V_A \cup V_B} = \frac{V_A \cap V_B}{V_A + V_B - V_A \cap V_B}$$

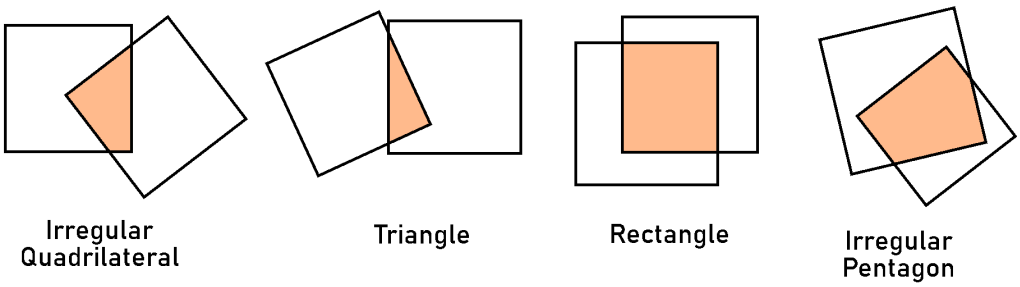

Figure 1.3: Examples of polygons (highlighted in orange) produced as the footprint of bounding box intersections.

In order to calculate IoU, we only need the volumes of both boxes (trivial since they are cuboids), and the intersection volume. Due to the possibility of bounding boxes having a non-zero yaw, the footprint (projection of the box onto the ground plane) of the intersection can be a variety of irregular polygons, as demonstrated in Figure 1.3. This means the intersection volume can be an irregular prism and is not as trivial to calculate.

The IoU between predictions and bounding boxes are used to match them up. Predictions are matched to the ground truth with which it has the highest IoU, unless another prediction matches that ground truth better. An IoU threshold eliminates possible matches if the overlap is not sufficient, e.g. IoU threshold of 0.3 means matches with IoU under 0.3 are not considered to match.

Once predictions and ground truths are matched up, the confusion matrix values can be derived. True positives are predictions which were matched with a ground truth, false positives are those which were not, and false negatives are ground truths which were not matched to a prediction. As there are no false labels, true negatives cannot be quantified. The average precision is then calculated either as before by varying the confidence threshold, or sometimes by instead varying the IoU threshold within a range.

1.6 Autoware

Autoware[1] is an open source software stack for autonomous driving built on the Robot Operating System (ROS)[2]. The project is open source and managed by the non-profit Autoware Foundation, with significant development efforts by TIER IV[3], a Japanese

[1] https://autoware.org
[2] https://www.ros.org
[3] https://tier4.jp/en

autonomous vehicles technology company (who played an important role in the creation of the Autoware Foundation).

The Autoware Universe repository on GitHub[4] provides components for various aspects of autonomous driving, including the perception component which is the focus of this work.

1.7 Research Question and Objectives

Research Question What is the performance of the LiDAR-based 3D object detection modules available in Autoware, as compared to state-of-the-art methods from recent literature?

Objectives

- Identify the LiDAR-based 3D object detection methods implemented in Autoware
- Determine an appropriate selection of these for evaluation, based on justification for their methodology from literature
- Develop software to interface with the detection modules via ROS, and collect the results
- Test and evaluate performance on both existing open source data sets, and own real world experimental data set, using standard metrics (average precision)

1.8 Document Outline

Chapter 2 introduces works in 3D object detection which are either current state-of-the-art methods, or form the foundation of those. Chapter 3 lays out the methodology for the model selection, experimental design, software interface with the detectors, and results collection and analysis. Chapter 4 summarises the results and provides an analysis. Chapter 5 critiques the methodology, compares the results to literature, and provides insight into further work.

[4] https://github.com/autowarefoundation/autoware.universe

Chapter 2

Related Work

2.1 Autonomous Driving Datasets

Three autonomous driving datasets have seen by far the most widespread usage [9] in 3D object detection works: KITTI [12, 13], Waymo Open [14], and nuScenes [15].

KITTI is the oldest of these (released in 2012) and was the standard for 3D object detection benchmarking for many years [9, 16]. The dataset has 22 scenes recorded in Karlsruhe, Germany, and the LiDAR sensor used was a Velodyne HDL-64E with 64 channels, operating at $10Hz$ with an $100m$ range. 3D bounding box annotations are provided for 8 classes, including car, van, truck, pedestrian, cyclist, etc. A criticism of this dataset from later works is that the data was collected in "ideal" environmental conditions; during the daytime, and in sunny weather. The dataset's proposed evaluation methodology for 3D objects includes the standard average precision metric, but also defines a cosine similarity variant based on the heading deviation from ground truth, matching bounding boxes using 0.5 IoU. This metric is called Average Orientation Similarity (AOS). Many works using this dataset opt for simply the usual metric of average precision [17–20].

2019[1] saw the release of two new large datasets, Waymo Open and nuScenes. Waymo Open records 1150 scenes of around 20 seconds each from the cities of San Francisco, Phoenix, and Mountain View in the USA. This covers a large geographic area of $76km^2$, far exceeding other datasets. Five LiDAR sensors are used, with the main one on top of the vehicle having its range restricted to $75m$, and the others restricted to $20m$. The classes annotated were vehicle, pedestrian, cyclist, and road sign. This dataset

[1] Dataset release date on their respective websites. Academic papers for each dataset were presented at 2020 CVPR conference.

includes variations in time of day and weather for the scenes, but still with most scenes occuring in the daytime. Proposed evaluation uses average precision with interpolation, and also introduces a new metric of average precision weighted by heading (APH). IoU of 0.7 is used for vehicles, and 0.5 for pedestrians. The work also points out a domain gap between the urban and suburban scenes in the dataset; detectors trained on only one domain perform more poorly in the other.

nuScenes provides a similar size of dataset (1000 scenes of around 20 seconds). The work claims to be the first dataset to provide radar data alongside camera and LiDAR, and also provides full 360° view for all sensors. The data was recorded in Boston and Singapore. The LiDAR sensor used was 32 channels at $20Hz$ with a $70m$ range. Annotations for 3D object detection use 10 classes. The evaluation methodology avoids using IoU for bounding box matching, instead using the 2D center distance. Average precision is included, but additionally a bespoke evaluation metric is devised called nuScenes Detection Score (NDS). Average precision is incorporated but other metrics (referred to as True Positive metrics) are also incorporated, such as Average Scale Error (ASE), Average Orientation Error (AOE), among others. All these metrics are combined to produce the NDS.

2.2 LiDAR-based 3D Object Detection

2.2.1 Overview

LiDAR sensors produce point cloud data, which is an unstructured collection of points in 3D space, representing the surfaces from which the LiDAR's lasers reflected. Such a 3D representation is well suited for locating objects within the scene [9–11] in order to derive appropriate bounding boxes.

Processing point cloud data presents a difficult computational challenge considering the real-time requirements of autonomous driving [11]. This is due to the fact that while point clouds are sparse and irregular, they also contain a large number of points [9]. This motivates the typical detector structure in which a feature encoder network (often referred to as a backbone network) produces an intermediate representation of the point cloud data, which is easier to process [11, 19]. This representation is then fed into a detector network (often called a detector head) [11, 19].

2.3 Point Cloud Feature Encoders

2.3.1 Voxel encoding

Drawing on techniques from 2D image detection, a leading methodology for detection applies a convolutional neural network (CNN) to the point cloud. In order to do this, the 3D space must be broken up into a regular 3D grid. Each cell in the grid is called a voxel (a 3D analogue of a pixel), and so this process is known as (regular) voxelization.

VoxelNet [17] was one of the first encoders to propose this methodology for 3D object detection in an end-to-end trainable deep learning architecture. Due to a variable number of points per voxel, random sampling is used to limit this.

The main feature of the encoder is the voxel feature encoder (VFE) layers, which are stacked in a chain to produce the point cloud feature representation. Element-wise max pooling is used to extract voxel-wise features. These are stored in a sparse tensor as many of the voxels are empty. This entails a reduction in computation cost and memory usage, which the authors state is ”a critical step in our efficient implementation”. Convolutional middle layers are then applied to produce a feature map. The final step of their method passes the feature map into a region proposal network to produce bounding box predictions.

The VoxelNet method demonstrated a significant improvement in average precision over contemporary methods on the KITTI 3D validation set, with 55.51mAP on the moderate level. Particular comparison is drawn by the authors with (at the time) state-of-the-art camera-LiDAR fusion method MV3D [21], finding an increase in average precision on the car class ”by 10.68%, 2.78% and 6.29% in easy, moderate, and hard levels respectively”.

However, due to the presence of 3D convolutions in the method, the inference time suffers [11, 16, 19]. The paper itself states an inference speed of $33ms$ ($\sim 30Hz$) on a TitanX GPU and $1.7GHz$ CPU, however other sources report around $225ms$ ($\sim 4.4Hz$) [18, 19] on GTX 1080Ti and i5-6500 4-core CPU, and even as high as $500ms$ ($2Hz$) [9], again on GTX 1080Ti. This arguably disqualifies VoxelNet as an appropriate methodology for an autonomous driving system in practice, as it does not address the real-time requirement.

However, SECOND [18] introduced sparse convolution operators into the VoxelNet method, greatly improving the inference speed, to around $50ms$ ($20Hz$). Along with additional improvements in a novel angle-loss regression approach and data augmentation approach for training, this method also surpassed VoxelNet in precision, with

60.56mAP on the KITTI 3D test set moderate level, compared to 58.25 for VoxelNet. SECOND became a widely used standard approach for voxel feature encoding [11].

2.3.2 Pillar-based Encoding

PointPillars [19] pioneered a novel feature encoding approach based on a bird's eye view (BEV) grid discretization, meaning the ground plane has a regular grid applied. The points within the point cloud are then contained within vertical columns (pillars) lying on the grid.

The method first decorates the points with calculated features (distance from arithmetic mean of points within pillar, and offset from pillar centre). The point features are then collected into a dense tensor of fixed size (D, P, N), where D is the number of features per point, P is the number of pillars, and N is the number of points per pillar. Sampling or zero padding is used as appropriate to construct the tensor. This stage exploits an important feature of point cloud data, in that a large proportion of the pillars in the point cloud are empty. The paper claims "at $0.162m^2$ bins the point cloud from an HDL-64E Velodyne lidar has 6k-9k non-empty pillars in the range typically used in KITTI for $\sim 97\%$ sparsity."

Each point within the tensor is fed into a simple feed-forward neural network per point (PointNet [22]). A max operation over the channels for the points is used to derive per-pillar features, which are then scattered back to the pillar locations on the BEV grid to produce a BEV feature map (called a pseudo-image in the paper).

The paper demonstrated the effectiveness of the encoder by attaching what was referred to as a "lean downstream network" to produce bounding box predictions. Using this, an impressive result (at time of publication) of 59.20mAP was achieved on the KITTI test 3D detection benchmark moderate level, while able to run at an inference frequency of $62Hz$, a three times increase over the next fastest method listed (SECOND). This demonstrated the potential of pillar-based encoders to provide the real-time capabilities required for autonomous driving without sacrificing precision.

2.3.3 Multi-view Pillar Encoders

Several later works build upon the PointPillars encoder method by introducing additional features by viewing the point cloud from multiple views, not just bird's eye. Some representative examples include MVF [20] and Pillar-OD [23].

MVF uses features from the perspective view (i.e. looking out at the point cloud

from the sensor location in the space) which uses spherical coordinates. This work also introduced a dynamic voxelization approach, which allows making use of all points in the cloud without sampling. The results on KITTI validation 3D showed an improvement to 79.12 AP on moderate level car class, over 74.7 AP in a reimplementation of PointPillars. However, the approach was slower than PointPillars, stating around $60ms$ inference time ($\sim 16.7Hz$) without hardware specified, with the PointPillars reimplementation stated at around $40ms$ ($\sim 25Hz$).

Pillar-OD builds further on MVF by replacing the spherical view with a cylindrical view, with the justification "the spherical projection in MVF ... causes unnecessary distortion of scenes". Along with a novel anchor-free pillar-based prediction module, this method achieves a further improvement over MVF in 3D mean average precision on the Waymo Open Dataset, with overall vehicle AP increasing to 69.8 from 62.93. No inference time metrics were reported.

2.4 Point Cloud Detection Methods

2.4.1 CenterPoint

CenterPoint [24] is a method combining both the PointPillars [19] and VoxelNet [17, 18] encoders with a centre-based 2D object detection head called CenterNet [25]. CenterNet reduces the image-based object detection problem to keypoint estimation, with the keypoints being objects' centres. Once the centres are identified, bounding box properties are predicted from nearby image features. The centre keypoint estimation is performed using a convolutional neural network which produces a heatmap of potential centre locations.

CenterPoint takes the BEV feature map produced by either PointPillars or VoxelNet (with CenterPoint versions termed CenterPoint-Pillar and CenterPoint-Voxel respectively), and considers it as a pseudo-image. This means instead of RGB channels in a regular image, we have the computed feature channels in each BEV grid cell. The pseudo-image is fed into a CenterNet detection head to produce preliminary bounding box predictions. The detection head is class-aware, as it produces a heatmap for centre detection per class, and chooses predictions as the best over all of these. A second stage in the detector refines the bounding boxes by incorporating extra data from the BEV features at the bounding boxes' faces. The results for CenterPoint-Voxel are stated to be 80.2 AP and 78.3 AP for vehicles and pedestrians respectively on Waymo Open test

set Level 1. The inference time is claimed as 77ms without hardware stated.

2.4.2 Other Detection Methods

PV-RCNN [26] integrates point-based and voxel-based learning methods into a single framework. The proposed voxel-to-keypoint encoder and RoI-grid pooling operator allow the method to ”encode much richer context information for accurately estimating object confidences and locations”. The method’s computational efficiency is improved in PV-RCNN++ [27], using novel strategies of sectorized proposal-centric keypoint sampling and VectorPool aggregation.

BtcDet [28] learns to identify occluded objects in the scene and uses learned shape priors with a probabilistic occupancy grid to improve detections.

CIA-SSD [29] is a single stage detector which uses a spatial-semantic feature aggregation to fuse high level semantic features with low level spatial features. An IoU-aware confidence rectification module is incorporated in the detection head to ”alleviate the misalignment between the localization accuracy and classification confidence”. Post processing uses distance-variant IoU-weighted Non-Maximum Suppression (DI-NMS).

2.5 Fusion Methods

2.5.1 Transformer-based Fusion

TransFusion [30] extends the CenterPoint method to perform camera-LiDAR fusion using a transformer architecture. A robust fusion strategy is presented which is able to ”leverage the cross-attention mechanism to build the soft association between LiDAR and images”. The method proceeds in two stages. The first stage follows CenterPoint to produce the centre predictions from the heatmaps. Rather than directly predicting the bounding boxes, the centre predictions are first combined with a category embedding. These are then fed as the queries into the transformer, with the keys and values coming from the BEV feature map in the CenterPoint encoder. The first stage uses only LiDAR data. This produces preliminary bounding box predictions which are then refined by incorporating camera data in another transformer in the second stage. This produces the final predictions.

As the first stage uses only LiDAR data, this can be extracted as a standalone LiDAR-only methodology, which is analysed in the paper as TransFusion-L. The results for this show an improvement to 65.6 mAP from 60.3 mAP for CenterPoint on nuScenes test

set. The inference time is stated as $115ms$ ($\sim 8.7Hz$), with a CenterPoint comparison claimed to be $117ms$ ($\sim 8.5Hz$), on Intel Core i7 CPU and a Titan V100 GPU. This demonstrated an inference time parity with an increase in precision when compared to CenterPoint. Whether $\sim 8.7Hz$ can be considered real-time enough for real applications in autonomous driving is debatable.

2.5.2 Other Fusion Methods

PointPainting [31] projects the LiDAR points onto the image plane and concatenates the point features with the results from an image semantic segmentation network. The concatenated points (which are referred to as ”painted”) can then be fed into a LiDAR-only method with some modifications to accomodate the extra features.

CLOCs [32] is a late fusion method which combines the bounding box results of both image and LiDAR detectors. The detections are combined into a sparse tensor which is used to predict improved object confidence scores for the 3D detections. Fast-CLOCs [33] significantly improves the computational efficiency by replacing the standard 2D detector with a 3D-Q-2D detector which projects the 3D detector’s predictions into the image plane, and uses these as region proposals for the 2D detection.

Chapter 3

Methodology

3.1 Overview

As discussed in 1.7, the first step was selecting appropriate models for evaluation from Autoware's repository. These were evaluated using average precision as discussed in 1.5.

For evaluation, both an existing large open source dataset and a smaller experimental dataset were used. Open source datasets are curated and carefully annotated, making them useful for evaluation on a large scale. Inclusion of an experimental dataset means the performance can be tested using more diverse real world data, in this case from an urban European city (Edinburgh) as opposed to the American urban, suburban, and highway locations used in many open source datasets.

The open source dataset selected was Waymo Open. The validation set was used, as the openly available version of the test set is unannotated. In addition, an experimental dataset was collected, for evaluating the detector in a particular scenario. Software development was required to interface with the detector, both in terms of feeding in the data, and collecting and analysing the results.

3.2 Autoware Module Selection

3.2.1 Overview

A review of the Autoware Universe repository's perception component found three modules performing LiDAR-only object detection. These are named `autoware_lidar_centerpoint`, `autoware_lidar_transfusion`, and `autoware_lidar_apollo_ins-`

tance_segmentation.

3.2.2 CenterPoint

autoware_lidar_centerpoint is an implementation of the CenterPoint [24] model, specifically CenterPoint using a PointPillars [19] backbone.

3.2.3 TransFusion

autoware_lidar_transfusion is an implementation of TransFusion-L, the LiDAR-only first stage of the TransFusion [30] method. This is built on CenterPoint with a VoxelNet [17] backbone.

3.2.4 Bespoke Apollo Method

autoware_lidar_apollo_instance_segmentation is a bespoke method ported from Apollo, another autonomous driving software stack. The following assessment is based on a README file[1] in the Apollo GitHub repository, linked from the Autoware documentation page[2]. The method does not appear to be based on an academic work. While this method uses some established ideas in recent literature (such as a BEV feature encoder, similar in concept to PointPillars), it also uses other stages such as point cloud pre-filtering, and an algorithmic approach (compressed Union-Find) to merge detections of the same object. While it is a LiDAR-only approach in terms of the sensor data it uses, it also requires provision of a HDMap of the road environment for the first stage filter. Finally, the output of the detector is not actually bounding boxes, rather point cloud clusters (a grouping of points comprising an object), from which a bounding box must be derived by other means.

3.2.5 Selection Decision

It was determined that both CenterPoint and TransFusion-L methods were appropriate models to evaluate, as representatives of recent high performing methods from literature, from 2021 and 2022 respectively. The Bespoke Apollo Method was therefore

[1] https://github.com/ApolloAuto/apollo/blob/r6.0.0/docs/specs/3d_obstacle_perception.md

[2] https://github.com/autowarefoundation/autoware.universe/blob/main/perception/autoware_lidar_apollo_instance_segmentation/README.md

determined to fall outside the scope of the work, for reasons including time constraints of the work, requirement of HDMap, and lack of academic backing.

However, the TransFusion component was very recently added to the Autoware repository, in June 2024[3] (during this project). There were technical problems in intregrating this new package into the existing University of Edinburgh Autoware fork, and as such it was not able to be included in this project. Only the CenterPoint model will be evaluated.

3.3 Experimental Aims and Design

3.3.1 Motivation

As part of the evaluation of the Autoware detector, a small experimental dataset was recorded using the sensors of the University of Edinburgh's autonomous vehicle. This was designed to evaluate the detector in a typical urban European scenario, and was specifically focused on detection of vulnerable road users (VRUs). The location chosen presented some typical difficulties of urban scenarios, while allowing a sufficiently controlled environment to focus the evaluation on VRU detection specifically. The scenario chosen was a pedestrian crossing the street in front of the vehicle, which is an extremely common scenario in a pedestrian-friendly urban European city such as Edinburgh.

3.3.2 Aims

- Collect data of pedestrian road crossing in order to evaluate Autoware detector capabilities in this scenario
- Observe and analyse the impact of confounding factors, such as distance from the sensor, varied VRU number and interactions, and typical urban occlusions

3.3.3 Sensor Specifications

The LiDAR sensor used for the data collection is an Ouster OS2-128, mounted on the roof of the autonomous vehicle. The accurate range is $200m$ at 10% reflectivity, and there are 128 channels. The sensor is operating at a sampling rate of $10Hz$. Sensor calibration was carried out before the beginning of this work.

[3] https://github.com/autowarefoundation/autoware.universe/pull/6890

Figure 3.1: A photograph of the experiment scene, taken from the camera on the autonomous vehicle's roof.

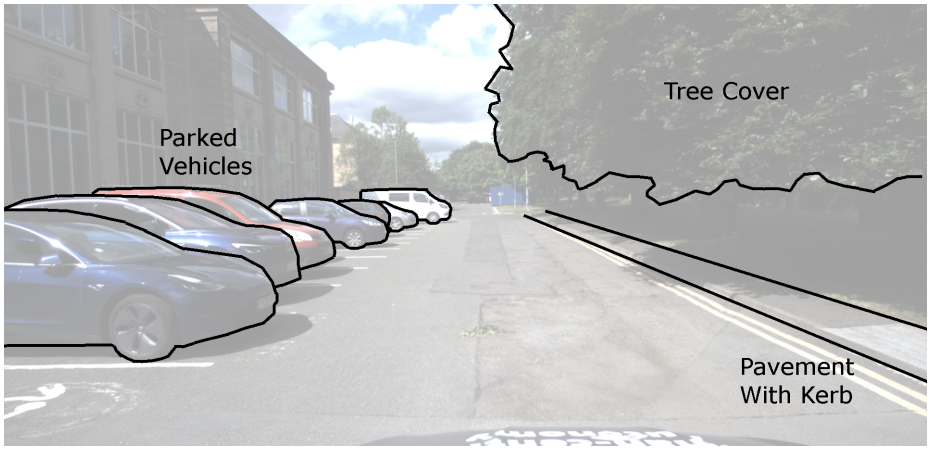

Figure 3.2: A photograph of the experiment scene with important elements labelled.

3.3.4 Scene Breakdown

The location for the experiment was a small car park at the King's Buildings campus of the University of Edinburgh. The scene is pictured in Figure 3.1. This section and following sections will contain graphics produced both from point cloud visualizations of the recorded data from the LiDAR sensor, as well as camera images from a front-facing camera sensor also mounted on the vehicle's roof.

The scenario chosen has the vehicle in a static position during data collection. The vehicle is positioned facing forward on a straight road of sufficient distance ($> 40m$) to allow variation in distance of the VRU crossing location. Typical urban features such as parked vehicles, trees on the roadside, and nearby buildings are present, as well as the raised curb with roadside pavement as is widespread within British cities. Figure 3.2 illustrates a visual breakdown of the important elements of the scene.

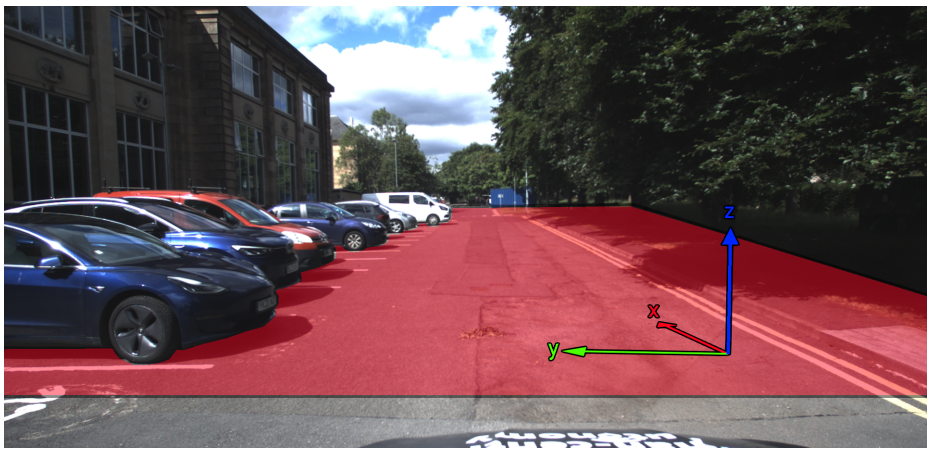

Figure 3.3: A photograph of the experiment scene with the defined experiment area highlighted in red. The axes directions are indicated for aid of visualisation (not the actual coordinate origin).

The defined experiment area is illustrated in Figure 3.3. This represents a restriction to between $2m$ and $50m$ in the x direction (depth), and between $7m$ and $-8m$ in the y direction (lateral). There was no restriction in the z direction (vertical). The positive x direction was forward into the scene, while the positive y direction was towards the left of the scene, and the positive z direction was upwards. Any detections outside the experiment area were not considered in the results, and accordingly no ground truths were present outside the area.

3.3.5 Design

In order to collect data incorporating the planned confounding factors, various crossing distances and VRU interactions were chosen as representatives to record data for. The distances chosen were $5m$, $10m$, $20m$ and $40m$, as a progression from quite close to the vehicle to the end of the small stretch of straight road. The distances were measured using a measuring tape of $10m$, with markers used for intermediate measurements. The measurements were from the front of the bonnet of the vehicle, which added roughly $2.5m$ to the measured distances from the frame of the sensor.

The VRU interactions were limited to the presence of 2 pedestrians. The first interaction was a simple one pedestrian crosses the road at a time. Further interactions had both pedestrians cross at the same time. One interaction had them walking opposite ways and crossing paths, while another had them walking side-by-side. Finally, a single pedestrian crosses while wheeling a bicycle. The interaction types will hence be referred

to as `1_ped`, `2_ped`, `2_ped_same_way`, and `1_ped_bike` respectively. Each of these four interaction types was recorded at each of the four distances, resulting in a total of 16 scenes for recording. Each scene consisted of roughly 10-20 seconds of data, which at a frequency of $10Hz$, results in roughly 100-200 frames of data per scene.

Occlusions were present in all of these scenes, with the pedestrian walking on the clear part of road in between vehicle occlusions on one side, and tree coverage on the other. The interactions with two pedestrians also provided some occlusion, with the opposite ways `2_ped` scenes having only a few frames of occlusion from one pedestrian to the other, while the same way `2_ped_same_way` scenes containing a significant level of occlusion on the pedestrian furthest from the sensor of the two as they walk alongside each other. In the `1_ped_bike` interaction, there is also a small level of occlusion provided by the bicycle itself when it is on the closest side of the pedestrian to the sensor, however this is minimal and the profile of the pedestrian is still largely visible.

One of the pedestrians was wearing a rucksack in order to provide some variation in pedestrian profile shape.

3.3.6 Data Annotation Strategy

In order to evaluate the detectors on the recorded dataset, it was necessary to provide ground truth labels for the pedestrians in the scene. The detections for the parked vehicles present in the scene were not considered in the analysis as they are fixed in the scene, and the aim of the scene is detection of the pedestrians.

The strategy used for annotating the pedestrians' paths through the scene was partial annotation of keyframes roughly every 3 seconds, with linear interpolation between these for the remaining frames. A fixed bounding box of dimensions (0.5, 0.7, 1.9) and axis aligned (zero yaw) was used, and only the center location was varied. The keyframe annotations were produced manually by eye using a bespoke point cloud visualizer script, present in the repository. Early experiments determined this was sufficient for providing reasonably accurate ground truth annotations.

This methodology is based on the assumptions that the pedestrian proceeds at a constant speed, and in a straight line. At an annotation frequency of every 3 seconds, this was determined (and manually verified) to be a sufficient approximation for the purposes of annotation. Extra keyframes were added in scenes where a significant deviation was observed using this methodology. Figure 3.4 illustrates a composite image of the pedestrian's location at two keyframes and two in-between frames, whose

Figure 3.4: A composite of the pedestrian's position in four frames. Blue boxes indicate manually annotated ground truths (keyframes), green boxes indicate interpolated annotations between these two keyframes. Green boxes are able to match the pedestrian profile appropriately.

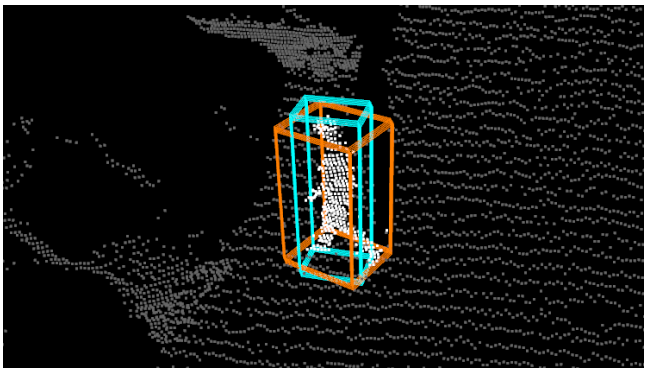

Figure 3.5: An example prediction in orange with ground truth in blue. The prediction's heading and larger dimensions better captures the pedestrian's feet in this frame than the ground truth with fixed size and zero heading. The IoU of these two boxes is 0.455.

annotations were interpolated from the keyframes shown.

However, due to the fixed size of the bounding box and lack of rotation, and after manual verification, it was determined a reduction from the standard of 0.5 IoU to 0.4 IoU was warranted for ground truth matching. This is intended to account for the small level of inaccuracy inherent in the annotation strategy used. Figure 3.5 illustrates an example of a matching between prediction and ground truth which justifies this reduction. Using an IoU of 0.5, this would not be considered a true positive match, however with an IoU of 0.4 it is.

3.3.7 Data Format

The sensor data was recorded using sensor drivers implemented in ROS, and so the data was recorded in the standard ROS data format, ROSbags. The ROSbags were stored in MCAP format.

3.4 ROS Detector Interface

3.4.1 Overview

As Autoware is built upon ROS, feeding data into the detectors requires creating a custom ROS node to read the data, package it into ROS messages, and send it on the appropriate topic. This was developed in an open source repository on GitHub[4], as a collaboration between myself, Hector Cruz from the University of Edinburgh AV lab team, and Julius Schulte, working on a separate but related MSc project. The development was done in Python. The data was fed into the detector in a sequential manner.

3.4.2 Ground Removal

A strategy which attempted to improve the precision of the detections was to filter out the points in the point cloud making up the ground. This was done using `autoware_ground_segmentation`, another package in Autoware's perception component. The chosen filter was scan_ground_filter. The point clouds were fed through this node before forwarding to the detector node. Evaluation was performed both using this method and without.

3.5 Result Collection

3.5.1 Overview

Similar to feeding data into the detectors, a ROS node is required to subscribe to the topic the detector publishes its results on. For simplicity this was included in the same node as the data publisher. The results for each frame were collected and saved in a JSON format. A library for interfacing with JSON files is inbuilt in Python. This

[4] https://github.com/ipab-rad/detection_utils

methodology allows for results to be collected once and analysed offline separately. which was critical in allowing multiple evaluations in different groupings, as discussed in 3.6.1.

3.5.2 Ground Truth Matching Strategy

Once the detector has produced the bounding box predictions, it is necessary to match these up with the ground truths and determine which predictions were true or false positives. In order to do this, the IoU between every ground truth and prediction must be found. This was done using the IoU algorithm from the python library PyTorch3D [34]. Using the IoU values, a matching must be found based on assigning ground truths to the prediction which has the highest overlap (IoU).

An algorithm referred to as the Hungarian Algorithm[5] is a widely used approach for this in literature [11, 14, 30]. An improved [35] variation on that algorithm, the Jonker-Volgenant algorithm, solving the same problem (linear assignment problem[6]), is implemented in the Python package scipy[7], and was used in this work.

3.6 Offline Evaluation

3.6.1 Calculation Groupings

In order to assess the impact of the various confounding factors, a few groupings of results for calculating the average precision were used. The groupings were by distance, by VRU interaction, single grouping (overall result), and single grouping without bicycles.

For example, the results from the `20m_1_ped`, `20m_2_ped`, `20m_2_ped_same_way`, and `20m_1_ped_bike` scenes would all be combined together when calculating the average precision by distance (in this case, at $20m$). Similarly, the results for `5m_1_ped`, `10m_1_ped`, `20m_1_ped`, and `40m_1_ped` would be combined together when grouping by VRU interaction type (`1_ped`). The single grouping combines the results for all 16 scenes to produce a full dataset result.

A final grouping using all scenes together except the bicycle ones (`5m_1_ped_bike`, etc.) was also used, as the bicycle scenes were deemed to be much more prone

[5] https://en.wikipedia.org/wiki/Hungarian_algorithm
[6] https://en.wikipedia.org/wiki/Assignment_problem
[7] https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linear_sum_assignment.html

to misclassification errors. This was because the detector was predicting what was annotated as a pedestrian wheeling a bicycle as instead a cyclist, and including the bicycle itself in the bounding box. This meant these scenes were significantly harder to correctly predict, and so a grouping assessing the precision on the more standard scenes was also included.

Chapter 4

Results and Analysis

4.1 King's Buildings Dataset

4.1.1 Results Tables and Graphs

Average precision results for distance, VRU interaction type, and overall groupings for CenterPoint are summarised in Tables 4.1, 4.2 and 4.3 respectively. Figure 4.1 illustrates the precision-recall curve for the overall result for sequential data feed.

A common source of error was misclassification of the pedestrian as a cyclist (including in scenes other than the scenes with a bicycle actually present). Since ground truth matching is on a per-label basis, classifying as a cyclist when there are no ground truth cyclists is always a false positive (and correspondingly the lack of correct prediction of the ground truth pedestrian results in a false negative). However, due to the lack of cyclist ground truths, the precision and recall for this class cannot be

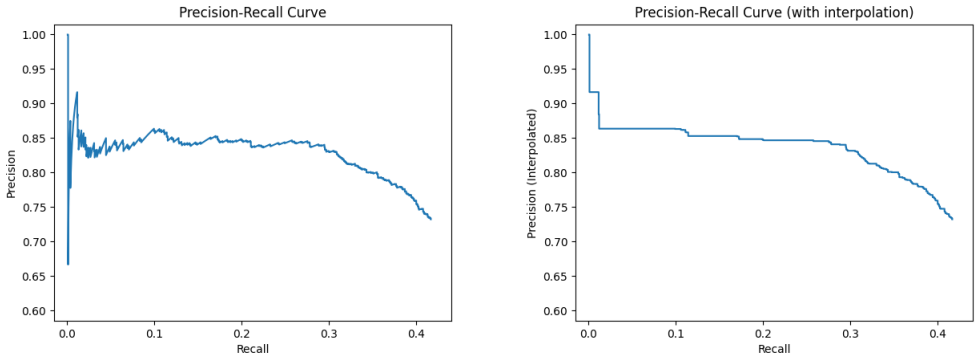

Figure 4.1: Precision-recall curve original (left) and interpolated (right) for King's Buildings overall result for sequential data feed.

Data Feed Strategy	Average Precision (%)			
	Distances			
	$5m$	$10m$	$20m$	$40m$
Standard	8.53	42.6	54.1	40.5
Ground Filtering	7.92	26.9	58.6	43.4

Table 4.1: CenterPoint pedestrian average precision on King's Buildings Dataset for distance result groupings, using different data feed strategies

Data Feed Strategy	Average Precision (%)			
	VRU Interaction Types			
	1_ped	2_ped	2_ped_same_way	1_ped_bike
Standard	45.7	42.3	30.7	7.88
Ground Filtering	45.3	37.9	25.5	7.35

Table 4.2: CenterPoint pedestrian average precision on King's Buildings Dataset for VRU interaction type result groupings, using different data feed strategies

Data Feed Strategy	Average Precision (%)	
	Overall Grouping	
	All	No Bicycle
Standard	34.9	39.2
Ground Filtering	31.9	35.8

Table 4.3: CenterPoint pedestrian average precision on King's Buildings Dataset for overall result groupings, using different data feed strategies

Scene	Cyclist Misclassification Count		Pedestrian Ground Truth Count
	Data Feed Strategy		
	Sequential	Ground Filtering	
5m_1_ped	1	0	120
10m_1_ped	1	0	136
20m_1_ped	3	1	121
5m_2_ped	2	3	166
5m_2_ped_same_way	13	4	185
10m_2_ped_same_way	1	1	169
40m_2_ped_same_way	1	1	70
5m_1_ped_bike	14	2	80
10m_1_ped_bike	26	5	80
20m_1_ped_bike	38	28	60
40m_1_ped_bike	4	2	45

Table 4.4: Cyclist misclassification count, and pedestrian ground truth count for comparison, across all scenes containing cyclist misclassification in King's Buildings Dataset. Unlisted scenes had no cyclist misclassifications.

defined. As a result, simple listings of cyclist misclassification count along with the total count of ground truths for pedestrians in the scene (i.e. an upper bound on the amount of predictions which identify the correct object but misclassify), are presented for comparison in Table 4.4.

4.1.2 Analysis

Out of the four distances evaluated, by far the best performance was found on the 20m distance. This could indicate that in the training data for the model, pedestrians in front of the vehicle are most often found around this range. The 5m predictions were much worse than the others, which could indicate the inverse, that pedestrians are almost never found there in the training data (pedestrians this close to the front of the vehicle would perhaps not be a realistic occurrence).

As expected, the results for the `1_ped_bike` scenes were much worse than the others, mostly due to the misclassification error. The scenes `10m_1_ped_bike` and `20m_1_ped_bike` were significantly affected by this, with 17.5% and 32.5% of ground truths misclassified respectively. The `2_ped_same_way` scenes also saw a drop off, which indicates the occlusion between VRUs presented a challenge to the detector.

The effect of the ground filtering strategy was mixed, with some scene types (20m, 40m) seeing an increase in AP, but overall the effect was lower precision. This could be explained by the training data having the ground present and so the detector is used to working around the ground being there.

4.1.3 Failure Modes

There are various instances where the detector commonly fails to correctly detect the pedestrian. A few are summarised in this section.

4.1.3.1 Cyclist Misclassifications

The pedestrian is sometimes classified instead as a cyclist, which usually is accompanied by a slightly wider bounding box than is appropriate. This occurs frequently in the scenes which do actually contain a pedestrian with a bike (see Table 4.4), but also sometimes in scenes without. Figure 4.2 illustrates the bounding boxes of the cyclist predictions in orange, with the actual pedestrian highlighted in white.

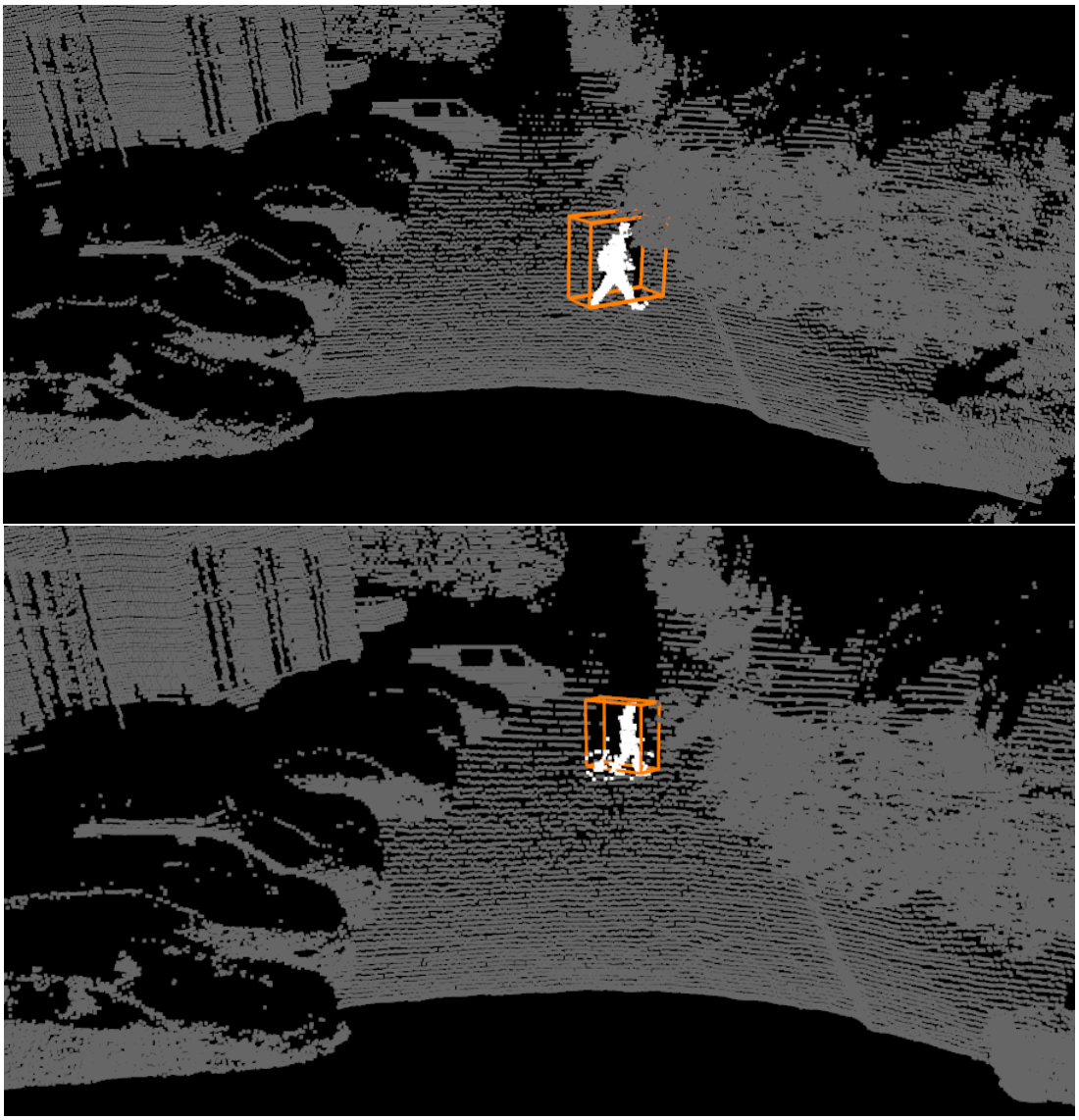

Figure 4.2: Cyclist misclassifications. Top figure is from `10m_1_ped`, and bottom is from `20m_1_ped_bike`.

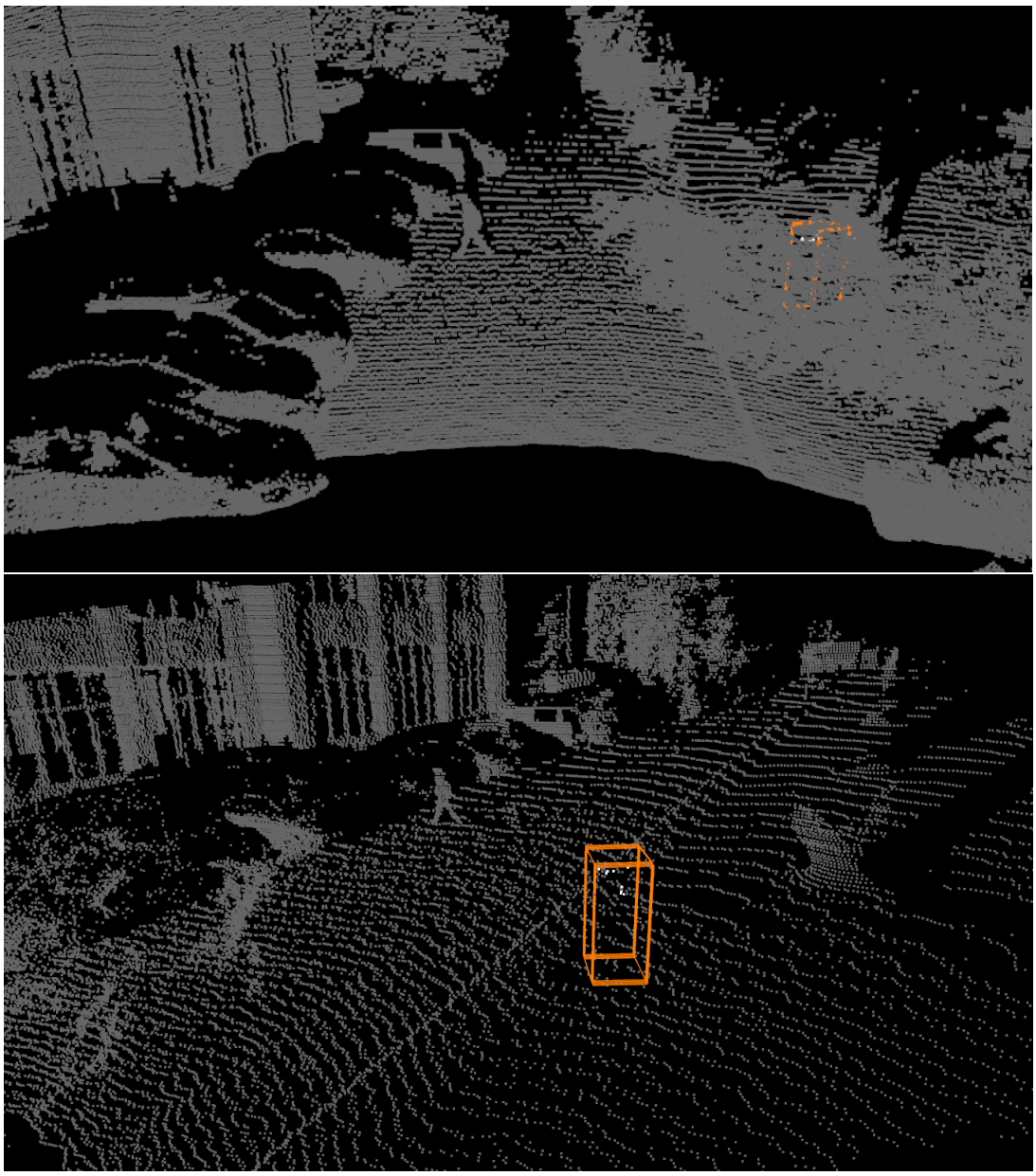

Figure 4.3: Tree false positive from `20m_1_ped`. Top marks location in scene and bottom shows the detected points from a better angle. The points are part of a section of leaves hanging down.

4.1.3.2 Tree False Positives

The detector had some trouble with the trees at the right hand side of the scene. One issue that occurred was detecting parts of trees (trunks or leaves) as pedestrians, which is a false positive. Figures 4.3 and 4.4 illustrate the erroneous predictions in orange with the section of tree highlighted in white.

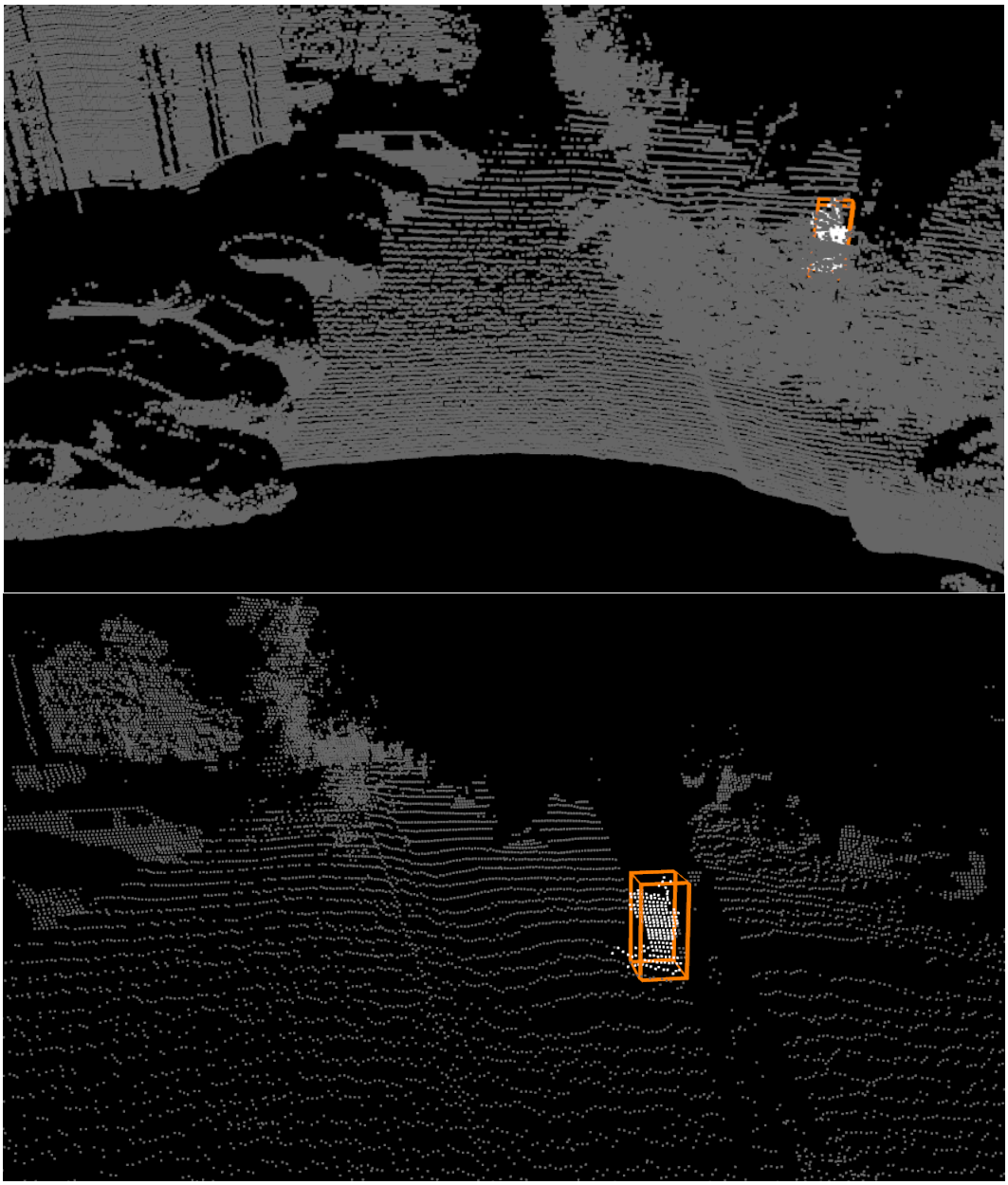

Figure 4.4: Tree false positive from `10m_2_ped_same_way`. Top marks location in scene and bottom shows the detected points from a better angle. The points are from a section of tree trunk.

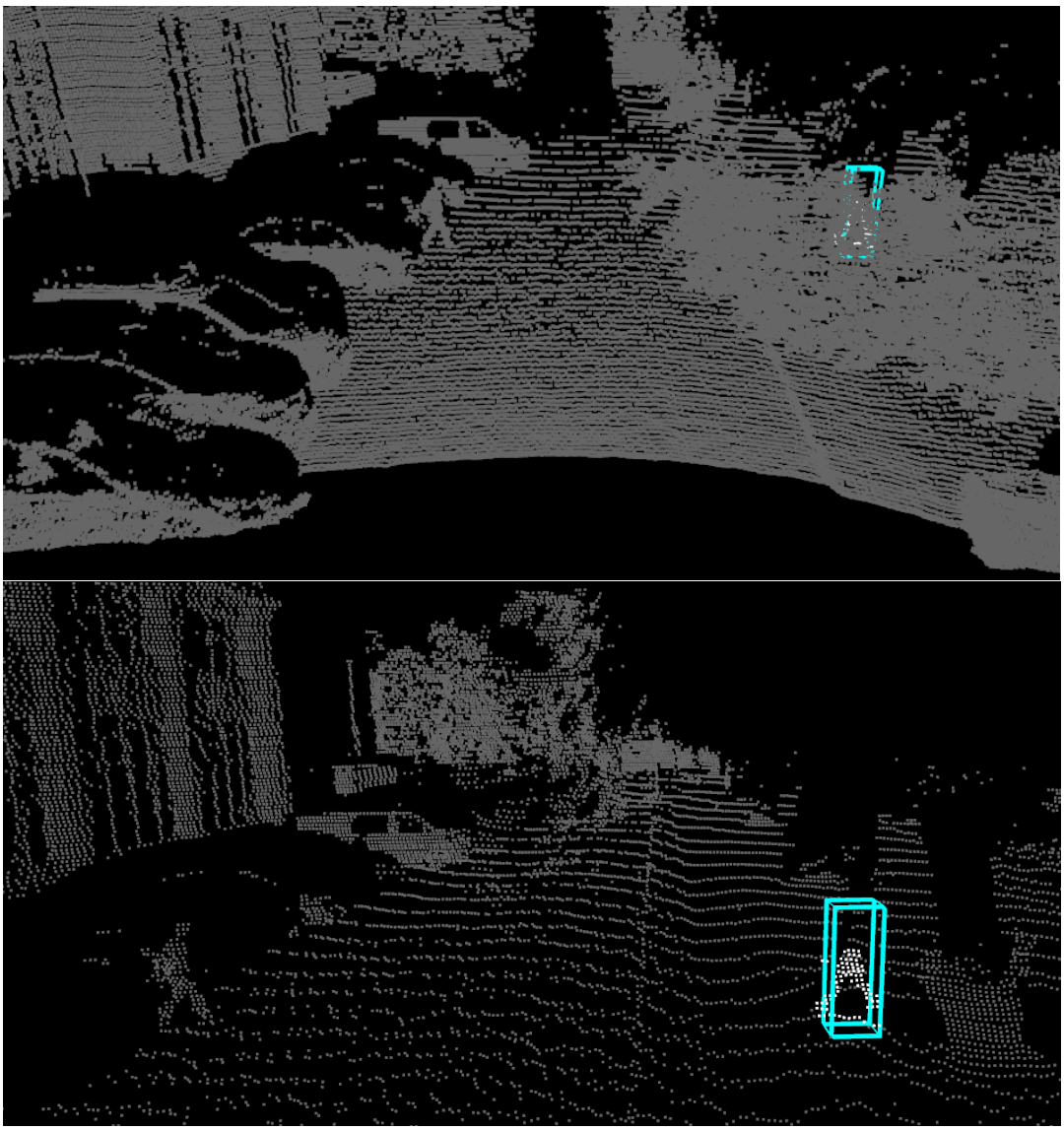

Figure 4.5: Occlusion by leaves from `20m_2_ped`. Top marks location in scene and bottom shows the pedestrian from a better angle. Only the bottom half of the pedestrian's body is visible in the point cloud, due to the leaf cover providing occlusion, as visible in the top image.

4.1.3.3 Environmental Occlusion

Occlusions in the scene such as leaves from the tree blocking the view on the right, and the parked vehicles on the left, meant the points making up the pedestrians were not always sufficient to be detected. Figures 4.5 and 4.6 illustrate the ground truth label in blue, with the unoccluded points from the actual pedestrian highlighted in white. Both of these examples did not have a prediction at all for the highlighted regions.

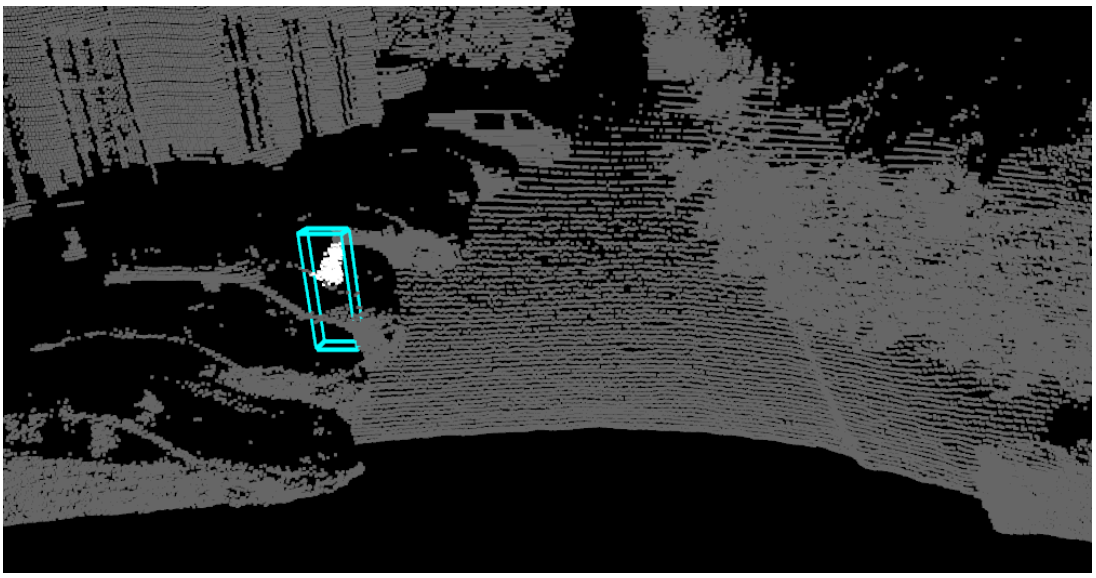

Figure 4.6: Occlusion by parked vehicle from `10m_1_ped`. Most of the pedestrian's body is occluded by the car.

4.1.3.4 VRU Occlusion

In the scenes with two pedestrians, the pedestrian closest to the sensor provides occlusion for the other pedestrian. The detector is usually able to overcome this confounder, but on the furthest away distance (40m) there are too few points in the occluded pedestrian to produce a detection in some instances. Figure 4.7 illustrates this, with the closest pedestrian marked in red, further pedestrian in blue, and predicted bounding box in orange.

4.2 Waymo Open Dataset

4.2.1 Results Tables and Graphs

Average precision results for each class and mean overall for CenterPoint are summarised in Table 4.5. Figure 4.8 illustrates the precision-recall curve for the Vehicle class for sequential data feed.

4.2.2 Analysis

Overall the results were very poor on the Waymo Open dataset. The performance was a little better on pedestrians than vehicles, and results for cyclists were at the very lowest. Ground filtering had a negative impact on the results.

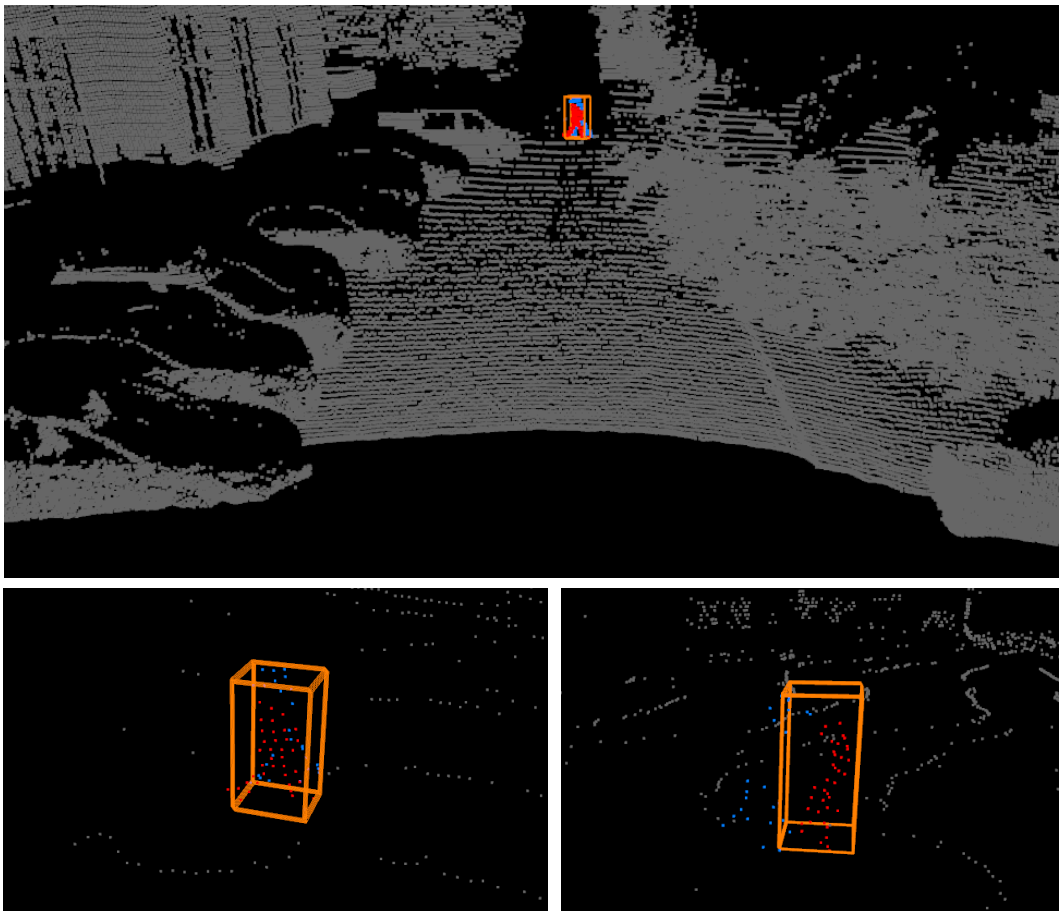

Figure 4.7: Occlusion by other pedestrian from `40m_2_ped_same_way`. Top image shows detection in scene, bottom left and right show the two pedestrians from front and side angles respectively. A few of the blue points were captured in the detection for the red pedestrian, but most of the blue points did not have a prediction at all.

	Average Precision (%)			
Data Feed Strategy	Mean AP (mAP)	Classes		
		Vehicle	Pedestrian	Cyclist
Standard	12.9	9.60	18.2	10.9
Ground Filtering	9.91	7.29	14.1	8.34

Table 4.5: CenterPoint pedestrian average precision on Waymo Open Dataset for various classes, using different data feed strategies

4.2.3 Failure Modes

4.2.3.1 Heading Misprediction

A common prediction failure occurs when the object's general location is correctly identified but the heading is off by too much to result in a positive match. An example of this is illustrated in Figure 4.9, where the blue box is the ground truth and the orange box is the prediction.

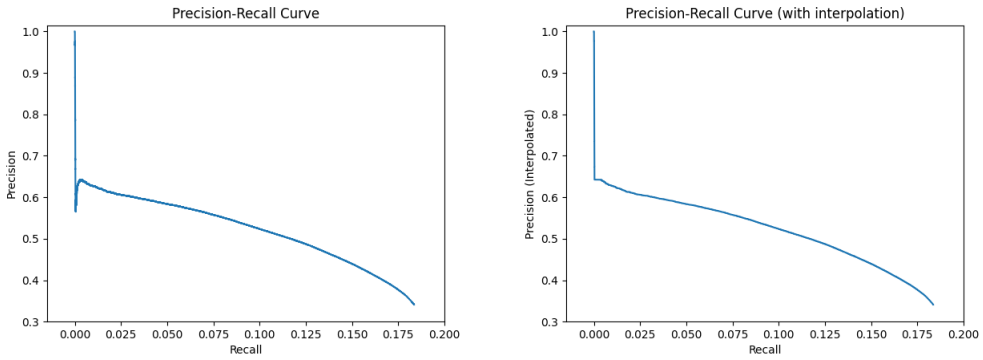

Figure 4.8: Precision-recall curve original (left) and interpolated (right) for Waymo Open Vehicle class for sequential data feed.

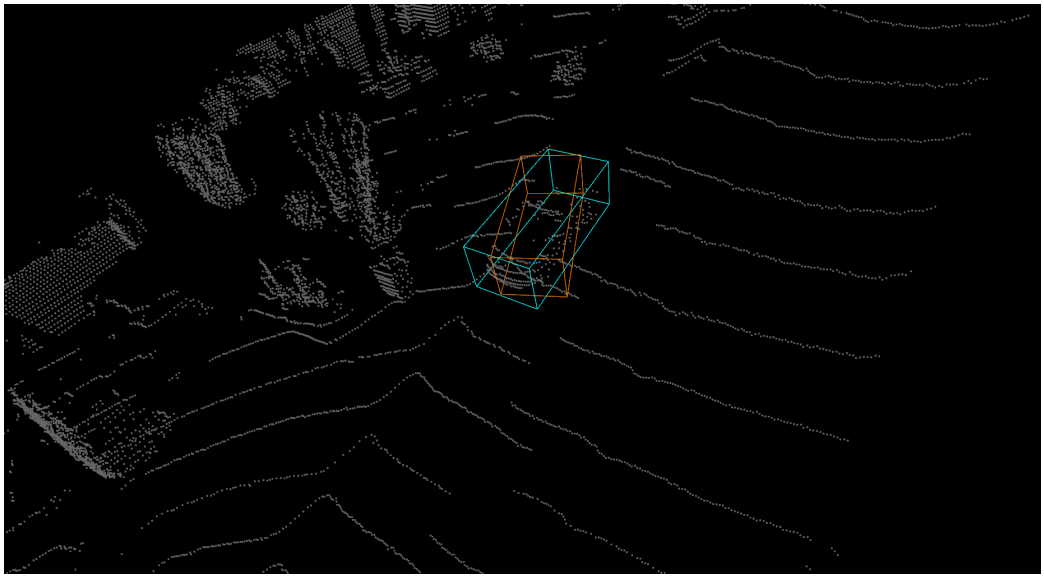

Figure 4.9: Prediction with incorrect heading. Prediction in orange, ground truth in blue.

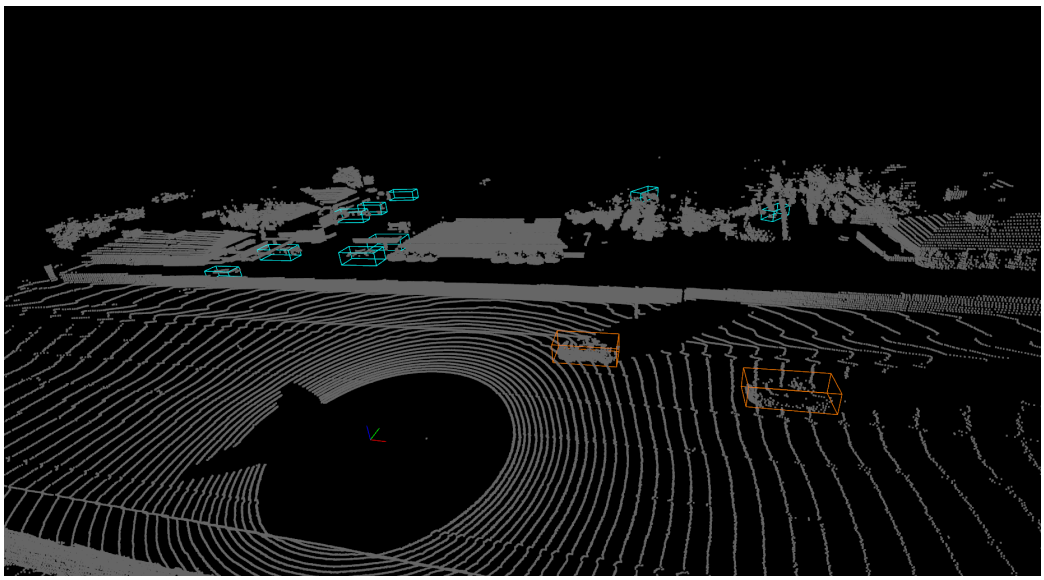

Figure 4.10: Scene with many missing predictions (false negatives). Ground truths matched to a predictions in orange, ground truths without a prediction in blue.

4.2.3.2 Many False Negatives

Some of the scenes had a significant number of false negatives where the model did not produce any predictions to cover the ground truths. Scenes significantly affected by this problem resulted in recall lower than 0.2 when all predictions are included, meaning less than 20% of the ground truths were successfully predicted. Figure 4.10 illustrates a scene where this occurs in the Vehicle class, with the successfully predicted vehicles in orange, and the unsuccessfully predicted ones in blue.

Chapter 5

Evaluation and Conclusion

5.1 Experimental Design Evaluation

5.1.1 Experiment Area

The restriction to only a specified area in front of the vehicle meant that the experiment was very focused on the particular scenario of pedestrians walking in front of the vehicle, and made it possible to keep the scene as quite a controlled environment. This also made the data annotation more straightforward and allowed for the annotation strategy used, as there were no unexpected or unannotated objects interfering with the scene.

However, a real autonomous driving scenario would require detections to be considered in a full 360° around the vehicle, and would certainly not only be limited to this particular scenario. As an evaluation of the detector in the specified scenario only, the experiment is thought to achieve the aims.

5.1.2 Data Annotation Strategy

The data annotation strategy allowed for producing a fully annotated set for the pedestrians from a much smaller number of manual annotations. Ground truth bounding boxes for every frame meant the typical detector evaluation strategy using IoU was possible. This kept the result analysis consistent with literature. The reduction to 0.4 IoU due to using a fixed size zero heading box was given justification. Manual verification was used to assess the quality of interpolated boxes, but without another source of ground truth to compare with this was the only appropriate method. Other ground truth matching strategies such as 2D center distance as in nuScenes [15] were considered, but standard IoU matching was chosen, as the most widely used approach in the literature.

5.1.3 Occlusions

The data annotation strategy did not quantify the level of occlusion on pedestrians in the scenes. Providing a quantification of this on each ground truth would have allowed a more thorough analysis of the impact of occlusions on the detections. However, this would have added significant complexity to the annotation strategy which was already a time-consuming manual task.

5.1.4 VRU Variety

Partly due to ethical restrictions on the number of participants for the experiment, only two pedestrians were able to appear in the scenes. This would not necessarily be representative of a busy urban road, where there would likely be far more pedestrians around. However, for the stated scenario of road crossing, this was determined to be sufficient. The inclusion of the bicycle added some variety but also posed a difficult question about how to classify a pedestrian wheeling a bike. It was determined to classify as pedestrian rather than a cyclist because they are walking rather than cycling. This meant the results for scenes with a bicycle were significantly worse, but this was accounted for in the results breakdown.

5.2 Results Evaluation

When compared to the stated results from the original paper on Waymo Open for CenterPoint, the results for Waymo Open in this work are far below the expected. CenterPoint was stated to achieve 80.2 AP and 78.3 AP for vehicles and pedestrians respectively on the test set. This evaluation instead used the validation set (as the test set is unannotated), but that does not account for the huge gap in performance.

A hypothesis for the cause of the performance gap is a domain difference. The CenterPoint model is stated in its documentation to be trained on both nuScenes and a TIER IV internal database. The version of CenterPoint in its original paper which was evaluated on Waymo Open (test set), was also trained on Waymo Open (training set). It may be the case the domain difference between the training data for the Autoware version and the Waymo Open validation data used for evaluation in this work was too large to produce a sufficiently precise detector for Waymo Open. Works on this topic such as [36] find detectors trained on a particular dataset in some cases do not generalise well to other datasets.

5.3 Conclusion and Further Work

While the detector provided a reasonable robustness for the pedestrian crossing the road scene in the King's Buildings dataset, the results for Waymo Open were significantly below expected, potentially due to the domain gap.

Further work should investigate incorporating a version of CenterPoint trained on Waymo Open into Autoware to compare the results with those in this work. Alternatively, the detector could be evaluated on nuScenes as this comprised a portion of the training data for the existing model. Further research into domain gaps in autonomous driving is also necessary.

References

[1] Y. Zhang, A. Carballo, H. Yang and K. Takeda, "Perception and Sensing for Autonomous Vehicles Under Adverse Weather Conditions: A Survey," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 196, pp. 146–177, Feb. 2023.

[2] SAE International, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," Standard J3016_202104, 2021.

[3] C. Koenig (BMW Group), "Level 3 highly automated driving available in the new BMW 7 Series from next spring," 10th Nov. 2023. [Online]. Available: https://www.press.bmwgroup.com/global/article/detail/T0438214EN/level-3-highly-automated-driving-available-in-the-new-bmw-7-series-from-next-spring (visited on 25/07/2024).

[4] Mercedes-Benz Group, "Certification for SAE Level 3 system for U.S. market," [Online]. Available: https://group.mercedes-benz.com/innovation/product-innovation/autonomous-driving/drive-pilot-nevada.html (visited on 25/07/2024).

[5] Honda, "Honda to Begin Sales of Legend with New Honda SENSING Elite," 4th Mar. 2021. [Online]. Available: https://global.honda/en/newsroom/news/2021/4210304eng-legend.html (visited on 25/07/2024).

[6] W. Morales-Alvarez, O. Sipele, R. Léberon, H. H. Tadjine and C. Olaverri-Monreal, "Automated Driving: A Literature Review of the Take over Request in Conditional Automation," *Electronics*, vol. 9, no. 12, Dec. 2020.

[7] A. J. Hawkins, "Waymo's driverless car: ghost-riding in the back seat of a robot taxi," *The Verge*, 9th Dec. 2019. [Online]. Available: https://www.theverge.com/2019/12/9/21000085/waymo-fully-driverless-car-self-driving-ride-hail-service-phoenix-arizona (visited on 25/07/2024).

[8] A. J. Hawkins, "Waymo is expanding its robotaxi service areas in San Francisco and Los Angeles," *The Verge*, 6th Aug. 2024. [Online]. Available: https://www.theverge.com/2024/8/6/24214454/waymo-robotaxi-expand-service-area-sf-la-map (visited on 06/08/2024).

[9] R. Qian, X. Lai and X. Li, "3D Object Detection for Autonomous Driving: A Survey," *Pattern Recognition*, vol. 130, 108796, 2022.

[10] D. Feng, C. Haase-Schütz, L. Rosenbaum *et al.*, "Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1341–1360, 2021.

[11] J. Mao, S. Shi, X. Wang and H. Li, "3D Object Detection for Autonomous Driving: A Comprehensive Survey," *International Journal of Computer Vision*, vol. 131, no. 8, pp. 1909–1963, 2023.

[12] A. Geiger, P. Lenz and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 3354–3361.

[13] A. Geiger, P. Lenz, C. Stiller and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013.

[14] P. Sun, H. Kretzschmar, X. Dotiwalla *et al.*, "Scalability in Perception for Autonomous Driving: Waymo Open Dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 2446–2454.

[15] H. Caesar, V. Bankiti, A. H. Lang *et al.*, "nuScenes: A Multimodal Dataset for Autonomous Driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 11 621–11 631.

[16] W. Liang, P. Xu, L. Guo, H. Bai, Y. Zhou and F. Chen, "A survey of 3d object detection," *Multimedia Tools and Applications*, vol. 80, no. 19, pp. 29 617–29 641, 2021.

[17] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018, pp. 4490–4499.

[18] Y. Yan, Y. Mao and B. Li, "SECOND: Sparsely Embedded Convolutional Detection," *Sensors*, vol. 18, no. 10, 2018.

[19] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang and O. Beijbom, "PointPillars: Fast Encoders for Object Detection From Point Clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 12 697–12 705.

[20] Y. Zhou, P. Sun, Y. Zhang *et al.*, "End-to-End Multi-View Fusion for 3D Object Detection in LiDAR Point Clouds," in *Proceedings of the Conference on Robot Learning*, vol. 100, Oct. 2020, pp. 923–932.

[21] X. Chen, H. Ma, J. Wan, B. Li and T. Xia, "Multi-View 3D Object Detection Network for Autonomous Driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 1907–1915.

[22] C. R. Qi, H. Su, K. Mo and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 652–660.

[23] Y. Wang, A. Fathi, A. Kundu *et al.*, "Pillar-Based Object Detection for Autonomous Driving," in *European Conference on Computer Vision*, 2020, pp. 18–34.

[24] T. Yin, X. Zhou and P. Krahenbuhl, "Center-Based 3D Object Detection and Tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2021, pp. 11 784–11 793.

[25] X. Zhou, D. Wang and P. Krähenbühl, "Objects as Points," arxiv:1904.07850, Apr. 2019.

[26] S. Shi, C. Guo, L. Jiang *et al.*, "PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 10 529–10 538.

[27] S. Shi, L. Jiang, J. Deng *et al.*, "PV-RCNN++: Point-Voxel Feature Set Abstraction With Local Vector Representation for 3D Object Detection," *International Journal of Computer Vision*, vol. 131, no. 2, pp. 531–551, 2023.

[28] Q. Xu, Y. Zhong and U. Neumann, "Behind the Curtain: Learning Occluded Shapes for 3D Object Detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, Jun. 2022, pp. 2893–2901.

[29] W. Zheng, W. Tang, S. Chen, L. Jiang and C.-W. Fu, "CIA-SSD: Confident IoU-Aware Single-Stage Object Detector From Point Cloud," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, May 2021, pp. 3555–3562.

[30] X. Bai, Z. Hu, X. Zhu *et al.*, "TransFusion: Robust LiDAR-Camera Fusion for 3D Object Detection With Transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022, pp. 1090–1099.

[31] S. Vora, A. H. Lang, B. Helou and O. Beijbom, "PointPainting: Sequential Fusion for 3D Object Detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 4604–4612.

[32] S. Pang, D. Morris and H. Radha, "CLOCs: Camera-LiDAR Object Candidates Fusion for 3D Object Detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10 386–10 393.

[33] S. Pang, D. Morris and H. Radha, "Fast-CLOCs: Fast Camera-LiDAR Object Candidates Fusion for 3D Object Detection," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2022, pp. 187–196.

[34] N. Ravi, J. Reizenstein, D. Novotny *et al.*, "Accelerating 3D Deep Learning with PyTorch3D," arxiv:2007.08501, Jul. 2020.

[35] D. F. Crouse, "On Implementing 2D Rectangular Assignment Algorithms," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 4, pp. 1679–1696, 2016.

[36] Y. Wang, X. Chen, Y. You *et al.*, "Train in Germany, Test in the USA: Making 3D Object Detectors Generalize," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 11 713–11 723.

Appendix A

Ethics Information

A.1 Participants' information sheet

Who are the researchers? Myself (Callum Turnbull), Julius Schulte.

What is the purpose of this study? Collect data using LiDAR and camera sensors on the autonomous vehicle to test 2D and 3D object detection algorithms.

Why have I been asked to take part? Act as pedestrians for the purposes of testing the aforementioned algorithms.

Do I have to take part? No – participation in this study is entirely up to you. You can withdraw from the study at any time, without giving a reason.

What will happen if I decide to take part? The data will be collected by the LiDAR and camera sensors on the autonomous vehicle. The data is point cloud data and images. The duration of the experiment session is around an hour. The experiment location is a quiet car park at King's Buildings campus.

Compensation No compensation.

Are there are risks associated with taking part? There are no significant risks associated with participation.

Are there any benefits associated with taking part? The collected data will be used as data for the projects of the participants.

What data are you collecting about me? The data we collect for our research is point clouds which are non-identifying, and camera image data which could potentially be identifying if your face is in view.

What will happen as a result of this study? The results of this study may be summarised in published articles, reports and presentations. The data will be deleted after the research is complete.

Who can I contact? If you have any further questions about the study, please contact the lead researcher, Callum Turnbull, s1935127@ed.ac.uk.

If you wish to make a complaint about the study, please contact inf-ethics@inf.ed.ac.uk. When you contact us, please provide the study title and detail the nature of your complaint.

Updated information If the research project changes in any way, an updated Participant Information Sheet will be made available on http://web.inf.ed.ac.uk/infweb/research/study-updates

Alternative formats To request this document in an alternative format, such as large print or on coloured paper, please contact Callum Turnbull, s1935127@ed.ac.uk.

A.2 Participants' consent form

As detailed in the ethics application, consent was gathered verbally from participants as this was limited to researchers for the project (myself and Julius).