Department of Informatics
Technical University of Munich

Master's Thesis in Robotics, Cognition, Intelligence

Real-Time LiDAR-Based 3D Object Detection on the Highway

Supervisor Prof. Dr.-Ing. habil. Alois C. Knoll

Advisor Walter Zimmer; M.Sc., Ercelik, Emec; M.Eng.

Author Xingcheng Zhou

Date June 1, 2021 in Garching

Disclaimer

I confirm that this Master's Thesis is my own work and I have documented all sources and material used.

Garching, June 1, 2021

(Xingcheng Zhou)

Abstract

As part of the Providentia project, this thesis aims to design a model that can detect the running vehicles on the highway in real-time. We analyze and split the task into two main problems. We first design a real-time LiDAR-only 3d object detector Pointpillars+ that could be applied to the real-world scenario on the highway. We choose Pointpillars as our baseline and improve its 3D detection performance at the expense of inference time latency. To prove the effectiveness of our proposed modules in Pointpillars+, we train and evaluate the model on KITTI, which is one of the most popular datasets in autonomous driving. The second practical problem is the lack of labeled frames in our customized dataset. We then present a semi-supervised algorithm, Statistical-Aware Pseudo Labeling (SAPL), to alleviate the label shortage problem. To implement the SAPL algorithm and prove its efficacy, we manually label several frames using the 3D bounding box annotation tool ProAnno and form a small customized Providentia dataset. We train and evaluate the model and show the validity of our SAPL algorithm based on the customized dataset. We do several sets of experiments for each module in the Pointpillars+ and SAPL algorithm and give some further suggestions for improvements in the future.

Contents

Abbreviations

CNN Convolutional Neural Network
LiDAR Light Detection and Ranging
NMS Non-Maximum Suppression
MLP Multiple-Layer Perceptron
TA Triple Attention
FC Fully Connected
FPN Feature Pyramid Network
IOU Intersection Over Union
PR Curve Precision Recall Curve
VFE Voxel Feature Encoding
FPS Farthest Point Sampling
kNN k Nearest Neighbour
GCN Graph Convolution Network
SSL Semi-Supervised Learning
HOG Histogram of Oriented Gradient
SIFT Scale Invariant Feature Transform
SVM Support Vector Machine
DNN Deep Neural Network
GHM Gradient Harmonized Mechanism
OHEM Online Hard Example Mining
LiDAR Light Detection and Ranging
mAP Mean Average Precision
SOTA State of the Arts
SAPL Statistical-Aware Pseudo Labeling

Chapter 1

Introduction

Autonomous driving provides a safer, more efficient, and more comfortable travel experience. It is a future-oriented technique that brings technology upgrading to the automotive industry and has become one of the hottest directions nowadays. The environment perception and positioning system is the foundation and prerequisite for vehicle path planning in autonomous driving control systems architecture.

In this chapter, the motivation of the Providentia project is introduced in section 1.1. We describe our role in the whole project and formulate the questions that aim to answer in this thesis in section 1.2. We also introduce the main challenges that we are met and list our main contributions in section 1.3. Finally, we give a brief overview of the structure of this thesis in section 1.4.

1.1 Motivation

The highway is a crucial application scenario for autonomous driving. Traveling on the highway generally takes a long time. People are prone to get distracted or feel sleepy during the tedious but highly concentrated driving process. Applying autonomous driving, in this case, can effectively reduce the accident rate and lower labor costs. Compared with the residential area situation, the highway usually has a better road environment and a more superficial, more stable driving condition, where fully automated driving is relatively easier to realize.

Current sensors in automated vehicles such as LiDAR, cameras or Radar, have limited detection perspectives because of their ego-based installation locations. For example, it is hard for cars to perceive passing vehicles due to the short sensoring range and shadowing effects. Providentia is a research project focusing on developing an intelligent infrastructure system on the highway, which provides external traffic information to automated vehicles. With the development of 5G radio techniques, a reliable and fast connection between vehicles and intelligent infrastructure becomes possible. The additional sensor data collected from the Providentia system can provide real-time and broad traffic information to all the road users, which gives a preview of the upcoming condition and helps the vehicles make more farsighted decisions such

as lane recommendations and accidental warnings.

Figure 1.1 depicts the workflow of the Providentia intelligent infrastructure system. The system is installed on the A9 Highway in Munich, and all the sensors are mounted on a gantry bridge. The whole system comprises eight scan cameras, LiDARs, a data-fusion unit, and the 5G mobile communication system. Cameras and LiDARs first provide accurate speed and position data, and the data-fusion unit then combines all information at the measurement point. Finally, the 5G communication system transmits the fused traffic data to all road users in real-time. It creates a complete digital twin of the highway, which is essential and widely applicable.

The second phase of the Providentia project expands the application scenario from the highway to the urban environment. LiDAR sensor will be used to detect all the objects in motion, such as Cars, Pedestrians, Motorcycles at the intersection. We will not include this part in the thesis.

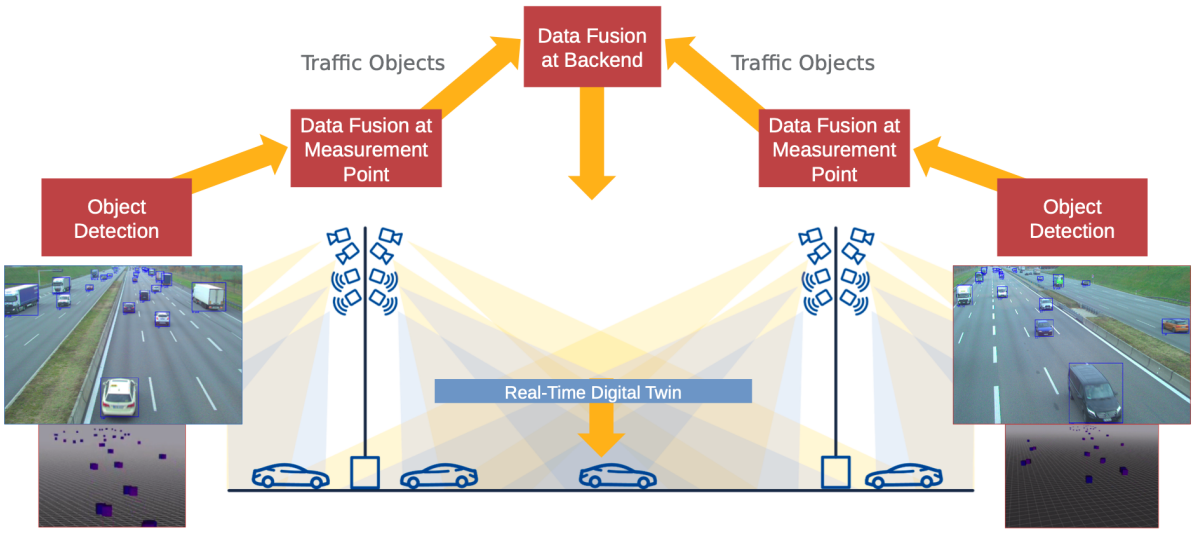


Figure 1.1: Overview of Providentia System

1.2 Problem Statement and Challenges

As a part of the Providentia project, this thesis focuses on solving the first step in the project, namely developing a real-time LiDAR-only 3D object detector for the moving vehicles on the highway. Specifically, we need to design a suitable detector that is fast and accurate and then collect sufficient data on the highway for model training and evaluation. Considering the high labor cost of manual labeling for 3D bounding boxes, we also need to solve the problem that only a few labels can be provided for training.

The whole thesis is then split into the following parts. Firstly, we study current

LiDAR-based 3D object detection methods and propose a model that achieves good 3D detection accuracy on KITTI. Since the number of KITTI's test dataset is far more extensive than ours, we assume that KITTI has good generalization ability, and the tricks which are proven to be helpful in KITTI can also boost the detection performance in our Providentia dataset. Then we record raw point cloud data on the highway and label a few parts of it manually utilizing 3D Bounding Box Annotation Tool (*3D-BAT*) [58]. Next, we apply transfer learning on the model to adapt the detector from KITTI domain to our Providentia domain and present methods to solve the few label problem. Finally, we insert the trained model into the toolchain for predicting vehicles on the spot.

In summary, two main problems are discussed and answered in this thesis:

- How to design a fast and robust LiDAR-only 3D object detector that performs well on the highway?
- How to solve the few label problem in 3D object detection when we need to train a model on the customized dataset?

To meet the speed and precision requirements of the Providentia system, we hope the detectors can achieve robust and precise detection results while keep running speed as fast as possible during the inference period. The designed model needs to reach a reasonable trade-off between the inference time and model performance. Besides, the annotation cost of 3D bounding boxes on the point clouds is quite expensive, and a few papers about applying semi-supervised learning on 3D object detection are proposed so far because of the large complexity of this task. Hence, we have to make full use of all labeled data and find some technical solutions to mitigate the label shortage problem.

1.3 Contribution

The thesis provides a complete analysis and solution for the single LiDAR detection part of the Providentia project. The main contributions in the thesis can be summarized as follows:

- We study the literature of SOTA 3D object detection methods and propose a single-stage LiDAR-only detector based on *Pointpillars*. We introduce three main modules to improve Pointpillars and evaluate the performance of the designed model on KITTI's validation dataset to check its validity.
- We record rosbag file on the highway with one Ouster LiDAR. We convert the rosbag file to a point cloud file with pcd format and create a Providentia dataset containing few manually labeled data and many unlabeled data.
- We propose a semi-supervised learning method Statistical-Aware Pseudo Labeling to mitigate the shortage of labels in 3D object detection task with the cus-

tomized dataset. We analyze the domain idiosyncrasy and evaluate the method on our Providentia dataset to prove its effectiveness and practicality.

1.4 Thesis Outline

The thesis is structured as follows: Chapter 1 includes the background of the Providentia project, the problems that need to be addressed, and the main contribution of the thesis. Chapter 2 introduces some theoretical concepts involved in the thesis: the property of point cloud, object detection task, common attentive mechanism in computer vision, and the evaluation metrics of 3D object detection. Chapter 3 briefly introduces some well-known existing LiDAR-based 3D object detection methods and semi-supervised learning methods for 2D and 3D object detection. Chapter 4 introduces Pointpillars+ with all its components and some implemented modules, which are proven to be helpful in the thesis. It also introduces SAPL algorithms. Chapter 5 analyses the characteristics of our customized dataset and introduces the implementation details in the experiments. Chapter 6 displays the experimental results of Pointpillars+ and SAPL algorithm, and chapter 7 discusses and analyses the results in chapter 6. Chapter 8 and chapter 9 conclude the whole thesis and give some suggestions for future work.

Chapter 2

Theoretical Background

This chapter presents the relevant theoretical knowledge in the thesis. Section 2.1 introduces some typical LiDAR sensors and the main properties of the point cloud. Section 2.2 gives a brief introduction of the object detection task and several commonly existing problems. Section 2.3 describes some classic attentive mechanisms in computer vision. Section 2.4 introduces the evaluation metrics used in 3D object detection.

2.1 LiDAR Sensor and Point Cloud

In autonomous driving, the vehicles should be able to perceive the surrounding environments. In comparison with camera-based perception methods, LiDAR Sensor captures more accurate 3D information. It is less susceptible to the influence of light, which is more suitable to the driving situation at night. Hence, 3D object detection with point cloud as input usually achieves better 3D performance than camera-based methods. It is necessary to introduce some basic concepts and properties of the LiDAR sensor and the point cloud data to understand LiDAR-based 3D object detection algorithms more comprehensively.

2.1.1 Introduction of LiDAR Sensor

LiDAR is a space measurement equipment that is mainly composed of a transmitter and receiver system. It determines the distance from LiDAR Sensor to an object by measuring the time from emitting the light pulse to receiving its reflected light. Given the speed of light, we can convert the traveling time to the measured distance. The relative three-dimensional coordinate of a point on the object's surface can then be accurately calculated based on the scanning angle of the laser. Assuming multiple light pulses are transmitted simultaneously, the point cloud data containing the coordinates of many points in a scene can be generated.

	OS1-64	Valeo SCALA2	Velodyne HDL-64E	Velodyne Alpha	Luminar Hydra
RANGE	120m	150m	120m	150-245m	250m
LASER BEAN	64-beam	16-beam	64-beam	128-beam	-
VERTICAL RANGE	45°	10°	26.8°	40°	30°
PRECISION	±0.7–5cm	±10cm	±2cm	±3cm	±1cm
POINTS PER SECOND	1.3million	-	1.3million	2.4million	-

Table 2.1: Comparision of three common LiDAR Sensors

Table 2.1 specifies some important parameters of the LiDAR sensors. LiDAR can detect objects ranging from a few meters to over 100m. Its detection range is always wide, which means the size of the feature map in the LiDAR-based model will be much larger than camera-based models to cover objects in the whole detection space. LiDAR sensor can be divided into single-beam LiDAR and multi-beam LiDAR based on the number of beams. The single-beam LiDAR has only one laser transmitter, which forms one discrete horizontal scanning line as the LiDAR sensor rotates. It can only determine whether some obstacles are in front or not. The multi-beam LiDAR has multiple laser transmitters in the vertical direction. It produces multiple discrete horizontal scanning lines simultaneously as the LiDAR rotates, during which a whole plane can be scanned. We usually choose multi-beam LiDARs such as 32-beam or 64-beam LiDAR for 3D object detection. Naturally, more beams also mean higher prices.

2.1.2 Properties of Point Cloud Data

The point cloud is a set of 3D points in the Euclidean space. Raw point cloud data which is generated from LiDAR sensor, has some major properties:

- Sparsity. As mentioned in section 2.1.1, LiDAR sensor collects point cloud data in a wide range. However, based on the working mechanism of LiDAR sensor, the scanned plane only forms a submanifold in 3D Euclidean space. Most of the areas in the detection space are empty and contain nothing, which leads to the sparse property of the point cloud.
- Unordered. Point cloud data is a set of unordered points. It means that the point cloud always represents the same scene, no matter how we shuffle its sequence. Assuming a point cloud contains N 3D points, the output of the network should be invariant to the input of its $N!$ permutations.
- Rotation and translation invariant. For a points-represented object in Euclidean space, its learned feature representation is invariant to the rigid transformation. It means that the category of the objects and the segmentation results of each point will not change when we apply rotation and translation to the point cloud.
- Spatial local relevant. After removing meaningless noisy points in the point cloud, the remaining points are not isolated. The relative locations of the points

in the neighbor contain valuable features of the spatial information of the point. The relevant relationship between neighboring points needs to be considered to capture the local spatial features more comprehensively.

2.2 Object Detection Task

Object detection is one of the most basic computer vision tasks. As illustrated in figure 2.1, the classification task aims to identify the given input category. The localization task requires the position information of the object, which is usually encoded as a bounding box. Object detection is the mixture of classification and localization tasks targeting multiple objects on the input.

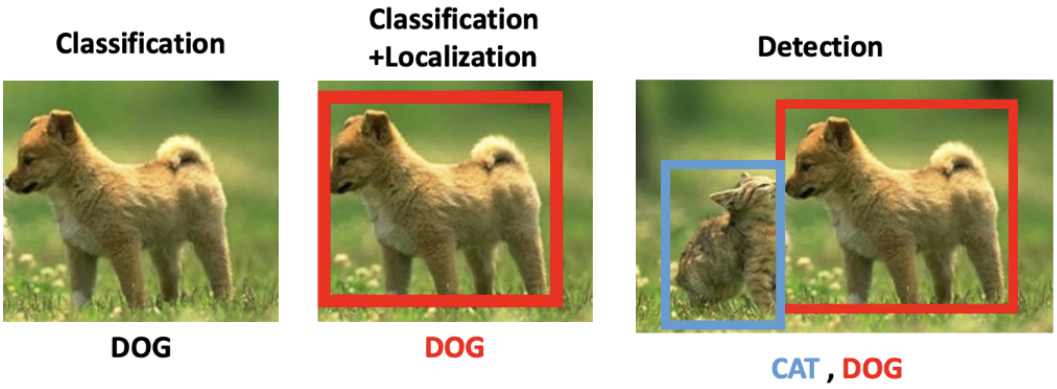


Figure 2.1: Comparision between classification, localization, and object detection

Classic object detection methods break down the task into two main steps: window-based feature extraction and classifier prediction. They employ the sliding window method and feature extractors such as histogram of oriented gradient (HOG) [9], or scale-invariant feature transform (SIFT) [23] for candidate box localization. Next, some commonly used classifiers such as support vector machines (SVM) are trained for category prediction. The introduction of *RCNN* [12] brings object detection into the era of deep neural networks (DNN).

Deep learning-based object detection methods can be divided into two groups based on the structure of the model: end-to-end single-stage-based methods or region proposal-based two-stage methods. Two-stage approaches first produce candidate bounding boxes and then rectify the initial proposals for better detection performance. On the contrary, single-stage approaches directly generate predicted results and provide an end-to-end training framework. Two-stage algorithms are usually more accurate with an additional proposal refinement module but slower, while single-stage algorithms are faster but with lower precision.

Object detection task contains 2D object detection and 3D object detection. 2D object detection usually takes images as the input and predicts the two-dimensional

positions and two-dimensional locations of 2D bounding boxes together with the corresponding categories. 3D object detection approaches have various inputs, such as LiDAR, radars, monocular or stereo cameras. The output bounding boxes of 3D object detection are usually represented with the oriented rectangular cuboids, which are encoded of three-dimensional position vectors (x, y, z) , three-dimensional location vectors $(height, width, length)$, and one-dimensional orientation vectors (θ).

There are some commonly existing problems for object detection algorithms in both 2D and 3D cases. The imbalance problem is one of the major problems which leads to the degradation of model performance. It comprises three main aspects:

- The imbalance between positive and negative samples
- The imbalance between hard and easy samples
- The imbalance between different classes

Specifically, in the object detection task, the positive sample refers to the foreground area whose overlap with any ground truth bounding box is over the positive threshold. In contrast, the negative sample refers to the background bounding box, whose IOU with any ground truth is lower than the negative threshold. The number of negative samples usually exceeds the number of positive samples by a large margin in object detection. In this way, negative samples will dominate the whole training and lead to low model utility if no measures are taken to deal with it. Usually, the ratio between positive and negative samples is empirically kept as 1:3 in the training procedure to mitigate the imbalance.

During training, hard and easy samples can be further divided into hard positive samples, hard negative samples, easy positive samples, and easy negative samples. Easy positive and easy negative samples usually account for a huge part of whole samples and dominate the loss function. Hard positive and hard negative samples only make up a small proportion of whole samples. The accumulative loss of hard samples is quite small despite the large loss per sample. Hence, it is difficult for a model to detect hard samples after training without additional tricks. Many effective methods are proposed to address this problem in the past few years, such as online hard example mining (OHEM) [38], focal loss [22], and gradient harmonized mechanism (GHM) [27].

The imbalance between different classes usually comes from the dataset. For example, the number of Cars is much larger than the number of Trucks in KITTI dataset. In this case, the model focuses on detecting Cars and ignore Trucks if we do not do any processing and use the raw dataset for training. Data argumentation is one of the most effective approaches to alleviate the imbalance between classes. Some common data argumentation tricks such as rotation, noise permutation can enrich the completeness of the dataset and increase the robustness of the model.

2.3 Attention Mechanism in Computer Vision

The attention mechanism is widely used in various tasks in artificial intelligence. It is essentially similar to the methods when human beings observe the environment. People used to pay more attention to some important local information and then combine all useful information to form an overall impression of the observed objects. The attention mechanism is the realization of this idea and was firstly implemented in natural language processing [42]. The Attention Mechanism assigns different weights to each part of the input to extract more critical information and transfer the focus from global to locally important features.

The attention methods in computer vision can be generally divided into three main categories based on its focusing domain, i.e., spatial domain-based, channel domain-based, and mixed domain-based. Suppose the shape of input image is $[N, C, W, H]$, where N means the batch size, C denotes the dimensionality of channels, $[W, H]$ denote the width and height of feature map, the spatial domain-based methods and channel domain-based methods transform the features along $[W, H]$ and C axis separately and produce a corresponding scoring matrix for re-weighting. Mixed domain-based methods combine the idea of these two approaches to enhance the performance of the attention mechanism.

2.3.1 Self-Attention Mechanism

The self-attention mechanism achieves great success in *Transformer* [28] and is recently widely employed in computer vision tasks. It is a variant of attentive mechanism which is good at capturing the internal correlation and aggregating the global structure of the input features. One of the most representative blocks is scaled dot-product attention. It provides an efficient and effective method for extracting context features by a triple tuple: key, query, and value.

The input is first converted to an embedded vector to produce query, key, and value by multiplication. The query is then multiplied by key to calculate the scoring matrix used to measure the internal similarity. Next, the scoring matrix is normalized, masked, and activated by the softmax function. The activated matrix is finally multiplied by the value to generate the output.

2.3.2 Squeeze-and-Excitation Network

Squeeze-and-Excitation(SE) Network [29] belongs to channel domain-based methods. It proposed the SE block, which is easy to be implemented and inserted into the existing network. SE block explicitly produces the correlation between different channels and adaptively re-calibrates the feature response. The importance of each feature channel can be learned during the training process. Based on the impor-

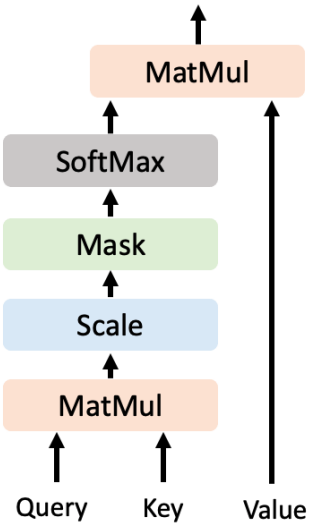


Figure 2.2: Scaled Dot-Product Attention. Source [28]

tance information, the SE block emphasizes the useful channels and suppresses the contribution of useless channels in the meantime.

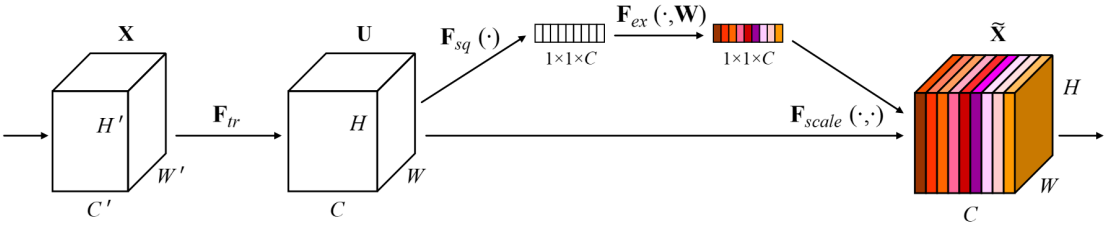


Figure 2.3: Squeeze and Excitation Block. Source [29]

SE block is composed of three steps: Squeeze, Excitation, and Scale. It firstly compresses the 2D spatial features of each channel to a single parameter, which theoretically contains the information of the global receptive field. The dimension of the output channel matches the input channel dimension, and it represents the global distribution of the channel response. It employs global average pooling on the feature map with the shape of $C*H*W$, and generates a squeezed feature map with the shape of $1*1*C$.

In the excitation step, two simple FC layers are followed by the squeezed feature map. It forms a bottleneck by reducing the channel dimensionality in the first FC layer from C to C/r, and increases the dimensionality of the channel back to C in the second FC layer, where r denotes the reduction ratio. Compared to a single FC layer, the bottleneck structure captures more non-linear information, which can better fit the complex interdependencies between channels, and greatly reduces the number of parameters and computation cost.

Then, a sigmoid function is employed to output the FC layers as a simple gating

mechanism to ensure the emphasis of multiple channels simultaneously. The last scale step in the SE block means the re-weighting process between different channels. It is a simple channel-wise multiplication operation between the original feature map and the vector expanded by the broadcasting mechanism from the excitation step.

2.4 Evaluation Metrics of 3D Object Detection

Average precision (AP) is the primary indicator for evaluating a single class's performance in object detection algorithms. It is the area under Precision-Recall Curve (PR Curve) and ranges from 0 to 1. PR Curve sets recall as x-axis and precision as y-axis. It plots a sequence of points by choosing different confidence thresholds and calculating the area under PR Curve. The definition of precision and recall are: $Precision = TP/(TP + FP)$, $Recall = TP/(TP + FN)$, where TP, TN, FP, FN denotes True Positive,True Negative, False Positive and False Negative separately.

KITTI dataset uses 40 interpolated points methods to calculate AP value. It firstly spaces the recall axis equally into 40 subgrids, and then takes the mean precision value of these subsampled recall positions. To satisfy the monotonicity of PR Curve, the precision value at recall position $\tilde{r}$ is replaced with the maximum precision value for any recall greater than or equal to $\tilde{r}$. The precise mathematical definition can be formulated as:

$$AP = \frac{1}{40} \sum_{r \in \{0.0,....,1.0\}} P_{interp}(r)$$

where

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

Mean average precision (mAP) is the mean value of multiple single class AP. It also ranges from 0 to 1 and is used to measure the algorithm's overall performance of all categories. Similar to the 2D object detection task, 3D object detection also use AP for model evaluation. There are three commonly used AP metrics in 3D object detection: AP_{2D}, AP_{3D}, and AP_{BEV}. AP_{2D} projects 3D detection results to the 2D image view and determines TP and FP by calculating 2D IOU with ground truth in the image. AP_{3D} directly calculates 3D IOU with ground truth in the 3D space. AP_{BEV} projects predicted bounding boxes to the bird eye view and then calculate the AP value.

Average precision only considers the performance on the location and classification abilites of the model, but in autonomous driving-relevant 3D object detection tasks, we still need to measure the direction of the detected vichels. Average Orientation Similarity (AOS) is usually used to evaluate the similarity of direction between the ground truth and detection results. The definition of AOS is:

$$AOS = \frac{1}{11} \sum_{r \in \{0.0,....,1.0\}} max_{\tilde{r} \geq r} s(\tilde{r})$$

where

$$s(r) = \frac{1}{|\, D(r) \,|} \sum_{i \in D(r)} \frac{1 + cos\Delta_{\theta}^{(i)}}{2} \delta_i$$

The calculation method of AOS is similar to AP, where $s(r)$ denotes the orientation similarity when the recall is equal to r, $|\, D(r) \,|$ means the set of all positive samples at the recall position r, and $s\Delta_{\theta}^{(i)}$ denotes the difference between the angle of i-th predicted bounding box and its corresponding ground truth. δ_i is set as 1 when the selected ground truth is not matched to any detected boxes and set as 0 otherwise. It is designed to make sure that each ground truth only matches the bounding box once.

Chapter 3

Related Work

This chapter gives an overview of the related work about our thesis. Section 3.1 introduces some well-known datasets in the field of object detection. Section 3.2 describes several state-of-the-art 3D object detectors and summarizes their performance in the table 3.2. Section 3.3 introduces some representative semi-supervised methods for 2D and 3D object detection.

3.1 3D Object Detection Datasets

This section introduces several currently most well-known 3D object detection datasets in the automated and autonomous driving domain. Object detection datasets cover various domains in our lives, from agriculture, wildlife to medical imaging and sports. Most of the datasets only contain RGB camera images and 2D annotations. In the autonomous driving domain, 3D information is strongly required during the perception process of the vehicles.

Many datasets in autonomous driving contain multiple type of sensors since single sensor information is commonly insufficient for a robust and comprehensive understanding of the whole scene. LiDAR sensors provide 3D location information proven efficient and accurate in novel LiDAR-based 3D object detection algorithms. It is the most widely used sensor for 3D object detection. Besides LiDAR information, most datasets also provide corresponding RGB images, which are dense and contain more semantic features to compensate for the sparsity in LiDAR point clouds. The cost of achieving RGB images is also much lower than the point clouds. Some datasets also provide radar information such as nuScenes. Compared with the LiDAR sensor, radar has a farther scanning range and is less affected by extreme weather conditions.

The KITTI dataset is one of the earliest 3D object detection datasets, whose data sources come from two gray-scale cameras, two RGB cameras, one Velodyne 3D laser sensor, and a GPS/IMU inertial navigation system. The dataset is recorded from a moving platform by driving around Karlsruhe, Germany, and its annotations for 3D object detection are comprised of 8 classes, i.e., *Car*, *Van*, *Trunk*, *Pedestrian*, *Person*, *Cyclist*, *Tram*, and *Misc*. KITTI belongs to a relatively small dataset containing around

Name	Year	Frames	3D boxes	Classes	Locations
KITTI [11]	2012	15k	200k	8	Karlsruhe
SUN RGB-D [40]	2015	5k	65k	37	-
ApolloScape [15]	2018	20k	70k	8	China
EuroCity [4]	2018	47k	-	7	European Cities
BDD100K [51]	2018	100k	0	10	San Francisco, New York
ArgoVerse [6]	2019	44k	993k	15	Miami, Pittsburgh
H3D [32]	2019	27k	1.1M	8	San Francisco
Lyft Level 5 [14]	2019	46k	1.3M	9	Palp Alto, London, Munich
A*3D [33]	2020	39k	230k	7	Singapore
BLVD [46]	2020	120k	249k	3	Changshu
Boxy [2]	2020	200k	1.99M	1	-
nuScenes [5]	2020	40k	1.4M	23	Boson, Singapore
Waymo [41]	2020	200k	12M	4	US

Table 3.1: Autonomous Driving datasets that are used for 3D object detection.

15 thousand frames and 80 thousand 3D annotated bounding boxes. It divides the objects into three difficulty levels based on their size, visibility, occlusion and trucation conditions:

- **Easy**: Min. bounding box height: 40 Px, Max. occlusion level: Fully visible, Max. truncation: 15 %
- **Moderate**: Min. bounding box height: 25 Px, Max. occlusion level: Partly occluded, Max. truncation: 30 %
- **Hard**: Min. bounding box height: 25 Px, Max. occlusion level: Difficult to see, Max. truncation: 50 %

The nuScenes dataset is published in 2019 by Nutonomy. It is about two times larger than the KITTI dataset with 40 thousand annotated frames containing 1.4 million bounding box annotations. The dataset is recorded by six RGB cameras, one spinning LiDAR sensor, five radars, one GPS, and one IMU sensor. In comparison to KITTI, it captured more scenarios such as different times (day, night) and weather conditions (sunny, rainy, cloudy). Each scene has a full 360 degree field-of-view. nuScenes contains 23 annotated classes totally, but only 10 of them are used for the object detection task, i.e., *Barrier*, *Bicycle*, *Bus*, *Car*, *Construction vehicle*, *Motorcycle*, *Pedestrian*, *Traffic cone*, *Trailer*, and *Trunk*.

The Waymo Open dataset is also published in 2019. It is the largest open-source 3D object detection dataset so far in autonomous driving. The dataset is recorded by five LiDAR sensors and five pinhole cameras with 12 million 3D LiDAR box annotations and 12 million camera box annotations. With five LiDAR sensors, Waymo provides a whole scene surrounding the recording platform and annotates four classes in total, i.e., *Vehicle*, *Pedestrian*, *Cyclist*, and *Sign*.

3.2 SOTA Point-Cloud Based 3D Object Detection

Current industrial solutions for 3D object detection in autonomous driving are mostly based on multiple sensors. They usually fuse LiDAR, RADAR, HD map, and camera information for more comprehensive, precise, and robust detection performance. Various 3D object detectors and fusion methods are proposed to fit the specific types and properties of the input information. Considering the final model in the thesis is the purely point cloud-based model, we only focus on introducing the SOTA Lidar-only 3D object detection methods in this section.

Same as 2D cases, 3D object detectors also can be categorized into single-stage and two-stage methods. The general structure of the single-stage models is composed of three main modules: Input representing and embedding module, 3D or 2D feature extraction module, and detection head module. Two-stage methods add an extra module after detection head for prediction refinement. According to the representation approaches of the input data, Lidar-based 3D object detection methods can be divided into four main categories: voxel-based, point-based, projection-based, and hybrid methods.

Table 3.2 summarizes several most representative LiDAR-only 3D object detection methods in recent years. It can be observed that dual-stage methods usually outperform single-stage methods by a large margin, while single-shot methods can reach a faster running speed during the inference period.

3.2.1 Voxel-Based Methods

Voxel-based methods partition the point cloud into many equally spaced voxels in the three-dimensional cartesian space and represent each voxel by grouping and sampling the points inside. Then the grouped points within each voxel are embedded into high-dimensional vectors for feature representation and dimension reduction.

Paper [57] proposed *VoxelNet*, which is the first method to unify voxelization, feature extraction, and proposal prediction together, and builds an end-to-end learning network in 3D object detection. It introduces the voxel feature encoding (VFE) layer to transform a group of points to a vector and apply 3D convolution to the embedded voxels to consolidate the z-axis. The extracted feature map is followed the region proposal network with 2D convolution for producing predicted bounding boxes. However, the computation cost of 3D convolution is too high due to the sparse property of the point cloud, which results in its low inference speed, at only 4 *Hz*. *SECOND* [47] presented an improved 3D sparse convolution method to accelerates the 3D feature extraction process. Since the predicted box is the same when orientation is equal to 0 and π, this paper also proposed a new sin-error loss for angle regression to solve the adversarial problem that the angle loss is different.

PointPillars [17] shows excellent inference time at 62 *Hz* after the acceleration of *TensorRT* on KITTI dataset. As shown in table 3.2, *PointPillars* is the fastest voxel-based 3D object detectors with only 2D convolution. The main contribution of PointPillars is the Pillar Feature Net. Pillar is a method of encoding point clouds and could be viewed as a special case of voxelization. It organizes the point clouds only in vertical columns. Without split along the z-axis, the number of pillars gets greatly decreased, and in the meanwhile, computationally expensive 3D convolution layers could be removed. The input of point could then be transformed to a dense tensor with size (B, D, P, N), where B denotes batch size, N denotes the number of sampled points per pillar, P denotes the number of non-empty pillars, D denotes the dimension of input. The pillars are then sent to a simplified PointNet with linear layer, 1D batch normalization, and ReLU to increase the feature dimension from D to C and embedded by a max operation. After encoding, the pillars with shape (B, C, P) can be easily scattered back to their original location in the bird eye view pseudo-image. Once scattered, the size of pseudo-image becomes (B, C, H, W), where (H, W) are the width and height of the image. In this way, 2D CNN can be utilized to process the 3D LiDAR feature map. As shown in the figure 3.1, the encoded pillar features are followed by a simple 2D CNN backbone with downsampled and upsampled blocks. It takes a modified SSD detection head from 2D object detection for prediction.

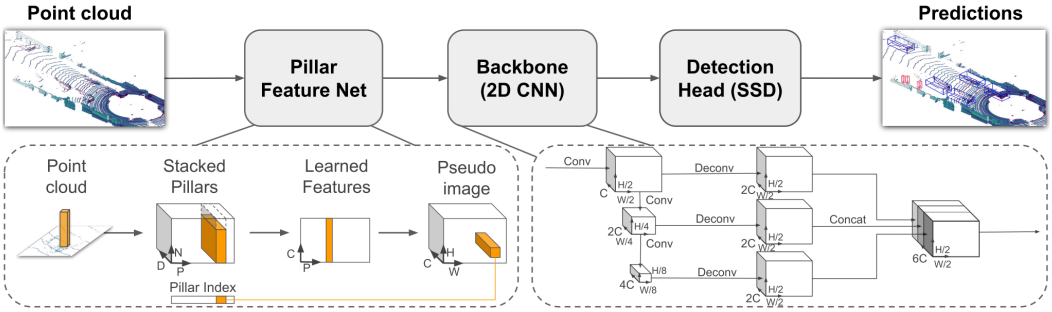


Figure 3.1: The Structure of PointPillars. Source [17]

HotspotNet [7] uses same voxelization methods and 3D sparse convolution backbone as *SECOND*. It proposed a new object representation method: Hotspot, and the corresponding anchor-free target assignment strategy. *HotspotNet* solves the point distribution imbalance problem inside the object by selectively choosing limited hotspots and balancing the number of positive hotspots for each object. This strategy also prevents the model from biasing too much towards large objects that contain more points. *SA-SSD* [13] takes advantage of the structural information of the point cloud to boost the localization precision of the predicted bounding boxes. Specifically, it employs an attachable auxiliary network with point-wise supervision tasks to help the 3D backbone acquire the object structure information from point clouds. This auxiliary network will not be used during the inference period and introduces no extra time cost. Besides, *SA-SSD* also introduces a PS-Warp module to mitigate the discordance between the classification scores and the corresponding predicted bounding

boxes.

CIA-SSD [55] proposed an IoU-aware confidence rectification module to alleviate the misalignment between the classification and localization in single-shot object detectors. It employs an extra IOU prediction head to rectify the confidence and further applies distance variant IOU weighted NMS to reduce the false-positive predictions for distant objects. *VoxelRCNN* is a voxel-based two-stage object detector. It does not introduce point-wise features for accurate location information but instead uses the spatial relationship between voxels to boost detection accuracy. Without considering point-level features, *VoxelRCNN* [10] saves large computation and storage cost and can reach a real-time inference rate at a speed of 25 FPS. Specifically, it first partitions the point cloud into small voxels and extracts the 3D features with a sparse 3D backbone. Then the 3D features are compressed to the bird eye view and fed into a 2D backbone and region proposal network for generating initial prediction proposals. The proposals are subsequently sent to Voxel ROI Pooling for feature refinement, which uses the spatial context from the last layer of the 3D backbone network. The features after Pooling are finally exploited to refine predictions.

3.2.2 Point-Based Methods

Point-based methods take raw point cloud data as the input and extract point-wise local features by grouping and aggregating the neighboring points. *PointNet* [34] is the cornerstone of point-based methods. It makes use of the unordered property of the point cloud and produces permutation invariant feature representations by applying a symmetric function on the aggregated features. Specifically, the raw point cloud with a shape of $n \times 3$ is first fed to several shared MLPs to extract point-level features and embed the point set to high dimension M. Then, it uses a max-pooling layer as the symmetric function to extract the global features and transforms the shape of the feature from $n \times M$ to $1 \times M$. Figure 3.2 depicts the structure of a simplified *PointNet*. *PointNet* also theoretically proves that its output feature is invariant to the order of the input points.

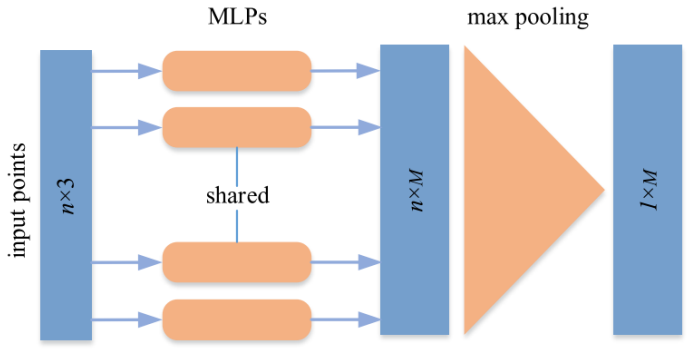


Figure 3.2: Structure of simplified PointNet. Source [34]

As mentioned in section 2.1.2, the point cloud is spatially local relevant. However, the feature of each point in *PointNet* is independently extracted, and the spatial relationship between different points is ignored. *PointNet++* [19] is proposed to address this limitation. The core operation in *PointNet++* is Set Abstraction Module, which is composed of three main steps: sampling, grouping, and *PointNet*-based feature extraction. Specifically, it first samples several points within the given space with Farthest Point Sampling (FPS). Then for each sampled point, a limited number of the neighboring points are grouped by using ball query or k Nearest Neighbor (kNN). Finally, *PointNet* is used to capture the local features of the grouped points. By hierarchically stacking several Set Abstraction Modules, *PointNet++* progressively extracts the local geometric structures of the point cloud and enlarges the receptive field of point-wise features layer by layer.

STD [50] is a classic point-based two-stage detector. It takes *PointNet++* as the backbone and generates initial proposals based on the manually designed spherical anchors on the points. The introduction of point-based anchors brings a higher recall rate and lower computational cost. The point-wise proposals are then sent to a PointsPool layer to transform the sparse points within the proposals to dense representation. Finally, *STD* introduces a 3D IOU branch in parallel with the box prediction branch for proposal refinement, which helps mitigate the misalignment between the localization results and classification scores. *PointRCNN* [37] first extracts the point-wise features using *PointNet++* and gets predicted foreground points by segmentation. For each foreground point, a bin-based 3D bounding box proposal is generated in the meantime. Then the points within each proposed box are transformed from LiDAR coordinate to their canonical coordinate systems for better local spatial feature representation. At last, the transformed spatial features and global semantic features from *PointNet++* are fused to generate accurate 3D box predictions. *PointRGCN* [52] introduces Graph Convolution Network (GCN) to *PontRCNN*. It leverages residual GCN to extract features, gather the useful information between the points within each proposal, and then utilize contextual GCN to aggregate the contextual features between different proposals.

3DSSD [49] is the pioneering work that brings point-based 3D object detection methods to real-time stage. It abandons the time-consuming upsampling and refinement modules in classic point-based methods and introduces a new fusion sampling strategy to preserve more representative points during the downsampling process. The fusion sampling is a mixture of Distance-FPS and Feature-FPS, where D-FPS tends to cover the entire point set as wide as possible, and F-FPS tends to preserve the foreground points. The backbone of *3DSSD* is composed of several set abstraction layers from *PointNet++* with D-FPS and fusion sampling strategy. The sampled points are then fed to the candidate generation layer to exploit point-wise features and produce the candidate points. Finally, a candidate-based anchor-free detection head is introduced to predict 3D bounding boxes.

3.2.3 Projection-Based Methods

Projection-based 3D object detection methods usually project the 3D point cloud to the bird eye view and use the projected 2D features as input. *BirdNet* [3] introduces a cell encoding method for BEV representation. It applies 2D convolutions on the BEV feature map for feature extraction and takes the region proposal network from Faster-RCNN [35] to generate 2D proposals. A ground estimation network works parallel with the 2D proposal generation network, which converts the 2D proposals to 3D predictions. *PIXOR* [48] views the height information of the point cloud as one channel in 2D images from the BEV. It then employs a fully convolutional network with a top-down branch similar to FPN [21] for feature fusion and 3D box prediction. Profiting from the efficient BEV encoding method and light-weight backbone, *PIXOR* runs at a speed of 28 FPS. *BirdNet+* [1] is also an end-to-end two-stage detector solely based on the BEV images. It removes the post-processing process and outperforms the *BirdNet* by a large margin.

3.2.4 Hybrid Methods

Hybrid 3D object detectors usually leverage both point-based and voxel-based feature representation methods in the model. Normally, point-wise features contain more accurate structural and location information but take much computational cost. In contrast, voxel-based features lose some original point cloud information, but it can be processed more effectively, and CNN has better perceptive ability than MLP.

Fast PointRCNN [8] leverages voxel representation for 3D feature extraction and initial proposal generation, and exploits raw point cloud for proposal refinement. The point-wise information in the RefinerNet brings extra location information. It boosts the regression performance, while 3D and 2D convolution layers in the voxel region proposal network quickly produce semantic and spatial features for different layers. *PV-RCNN* [36] takes advantage of both point-based and voxel-based methods and has a strong ability to mine hard samples. It first voxelizes the input point cloud and takes 3D sparse convolution as the backbone to extract multi-scale 3D features and propose initial 3D boxes. Then Voxel Set Abstraction module samples key points in the raw point cloud and concatenates the multi-scale semantic features for each key point to a vector. The argumented point-wise features are then re-weighted by segmentation and sent to ROI-grid Pooling module together with the bounding box proposals to keep rich context information for location refinement and confidence prediction. *SIENet* [20] adopts a hybrid-paradigm RPN for feature extraction and proposal generation, which is composed of voxel-wise sparse convolution branch, point-wise voxel set abstraction branch, and an auxiliary branch. The proposal is fed to a spatial shape prediction network for 3D shape completion. The completed proposal is subsequently extracted and fused with the pooled features from ROI-grid Pooling. Finally, it produces refined 3D boxes and confidence based on the fused features. It is hard for hybrid methods to run fast, but they always have more robust

detection performance because of hybrid feature representation in multiple modalities.

3.3 Semi-Supervised Learning for Object Detection

Machine learning methods can be roughly divided into supervised or unsupervised methods according to whether labels are included in the training data. However, in many practical cases, labeled samples and unlabeled samples are often available simultaneously, and unlabeled samples account for the vast majority, while labeled samples are relatively fewer. Although labeled data can effectively boost the model's performance, obtaining labels is usually quite difficult. It requires much time, sufficient equipment, and theoretical knowledge as support. The concept of semi-supervised learning is then introduced to cope with these circumstances.

Consistency regularization and self-training are the two most representative methods in semi-supervised learning. Self-training-based methods firstly train the model with available labeled data and make predictions. The predictions with high confidence are then added to the training data for next-round training. Consistency-based methods enforce the prediction of the model to be consistent, although the input is slightly permuted. It improves the generalization ability of the model and makes the model more robust to the noise.

3.3.1 Semi-Supervised 2D Object Detection

Although semi-supervised learning achieves great success in the last few years, most methods focus merely on image classification tasks. There are a limited number of works incorporating semi-supervised learning methods into object detection, and most of them are based on the idea of self-training and consistency regularization.

The consistency-based semi-supervised learning method for object detection (CSD) [16] is a consistency-based method. The prediction of unlabeled data is enforced to keep consistent with the prediction of its flipped images. The method employs L2 Loss as consistency loss for localization and Jensen-Shannon Divergence as consistency regularization for classification.

SSL framework for visual object detection along with a data augmentation strategy (STAC) [39] is a simple yet effective semi-supervised learning framework for visual object detection along with a data augmentation strategy. The method is based on pseudo labeling and augmentation-driven regularization. It first trains the model with labeled images and then generates predicted bounding boxes and categories. The predictions are post-processed by NMS and also filtered to remove redundant proposals with low confidence. Next strong data augmentations such as global geometric rotation, cutout, and boxes-level rotation are applied to unlabeled data and

the generated pseudo labels. The final training loss contains the supervised and unsupervised parts.

Paper [24] proposes Unbiased Teacher, which introduces the teacher-student mutual learning strategy to semi-supervised learning for object detection. The detector is firstly trained with labeled data. It applies weak augmentation to unlabeled samples and feed them to the teacher model for generating pseudo labels. The unlabeled samples and pseudo labels are next strongly augmented to train the student model. The teacher model's weight also gets slowly updated via exponential moving average (EMA) from the student model. Specifically, the unsupervised loss of the student model only contains the classification part, and the teacher model usually performs better than the student model by around 2 mAP after the whole training process.

3.3.2 Semi-Supervised 3D Object Detection

There are only two prior works discussing semi-supervised learning for 3D object detection. Self-Ensembling Semi-Supervised 3D Object Detection (SESS) [53] adopts a teacher-student mutual training scheme, where the student and teacher model utilize different data augmentation strategies. The student model learns from supervised loss and three unsupervised consistency losses: center-aware, class-aware, and size-aware loss. 3DIoUMatch [43] points out that the consistency loss in SESS is suboptimal because the teacher and student model predictions are only uniformly regularized. Like SESS, 3DIoUMatch also adopts two-stage training schemes with teacher-student structure and takes VoteNet as the structure of teacher and student networks. The input samples of the student model are strongly augmented, and the teacher model takes weakly augmented data. It applies a confidence-based filtering method to remove proposals with low qualities and directly trains the student model with selective pseudo labels after IoU-guided lower-half suppression. Experimental results show that 3DIoUMatch outperforms SESS in both SUN-RGBD and ScanNet datasets by a large margin.

Method		Year	Paradigm	Hardware	Speed (fps)	Cars		
						E	M	H
Voxel-based	VoxelNet [57]	2017	single -stage	TITAN X	4	77.47	65.11	57.73
	SECOND [47]	2018	single -stage	GTX 1080 Ti	26	84.65	75.96	68.71
	PointPillars [17]	2019	single -stage	GTX 1080 Ti	**62**	82.58	74.31	68.99
	HotspotNet [7]	2019	single -stage	TITAN V100	25	87.60	78.31	73.34
	SA-SSD [13]	2020	single -stage	GTX 2080 Ti	25	88.75	79.79	74.16
	CIA-SSD [55]	2021	single -stage	TITAN XP	32	89.59	80.28	72.87
	VoxelRCNN [10]	2021	two -stage	GTX 2080 Ti	25	**90.90**	**81.62**	**77.06**
Point-based	STD [50]	2019	two -stage	TITAN V	12	87.95	**79.71**	**75.09**
	PointRCNN [37]	2019	two -stage	TITAN XP	10	86.96	75.64	70.70
	PointRGCN [52]	2019	two -stage	TITAN XP	4	85.97	75.73	70.60
	3DSSD [49]	2020	single -stage	TITAN V	**25**	**88.36**	79.57	74.55
Projection-based	BirdNet [3]	2018	two -stage	-	9	40.99	27.26	25.32
	PIXOR [48]	2019	single -stage	TITAN XP	28	**81.70**	**77.05**	**72.95**
	BirdNet+ [1]	2020	two -stage	TITAN XP	10	70.14	51.85	50.03
	BEVDetNet * [30]	2021	single -stage	GTX 2080	**330**	82.46	77.90	77.45
Hybrid	Fast PointRCNN [8]	2019	two -stage	Tesla P40	**16**	84.28	75.73	67.39
	PV-RCNN [36]	2020	two -stage	GTX 1080 Ti	8	**90.25**	81.43	76.82
	SIENet [20]	2021	two -stage	TITAN XP	6	88.22	**81.71**	**77.22**
	HVPR [31]	2021	single -stage	TITAN V	36	86.38	77.92	73.04

* The result is measured on the KITTI's validation set with IOU threshold 0.7.

Table 3.2: Comparision of LiDAR-only 3D object detection models speed and 3D AP results on KITTI test benchmark. **E**, **M**, **H** means the difficulty level as easy, moderate and hard seperately in car class. The IOU threshold for cars is 0.7.

Chapter 4

Solution

In this chapter, we propose the solutions for the aforementioned problems. Section 4.1 introduces a LiDAR-only single-stage 3D object detector based on *Pointpillars*. We call the solution *Pointpillars*+. It contains three main improving modules, which are formulated in section 4.1.2, 4.1.3, and 4.1.4. Section 4.1.5 proposes two modules proven to be useful but not included in the model at last. Section 4.2 introduces our solution for few label problems, which is called Statistical-Aware Pseudo Labeling (SAPL).

4.1 Proposed Detection Solution

In this section, we introduce the structure of our proposed *Pointpillars*+ network. The default frequency of the Ouster Sensor is about 10 to 20 FPS. Therefore, we define the real-time running speed as 20 Hz. However, the inference speed of Pointpillars is 42Hz without the acceleration of *TensorRT*, which means there is some redundant time gap that can be used to improve the accuracy of *Pointpillars*. Hence, we use *Pointpillars* as the baseline and improve its detection performance by introducing three additional modules, and we follow the rule that the model's performance can be boosted without sacrificing its inference speed too much.

4.1.1 Network Overivew

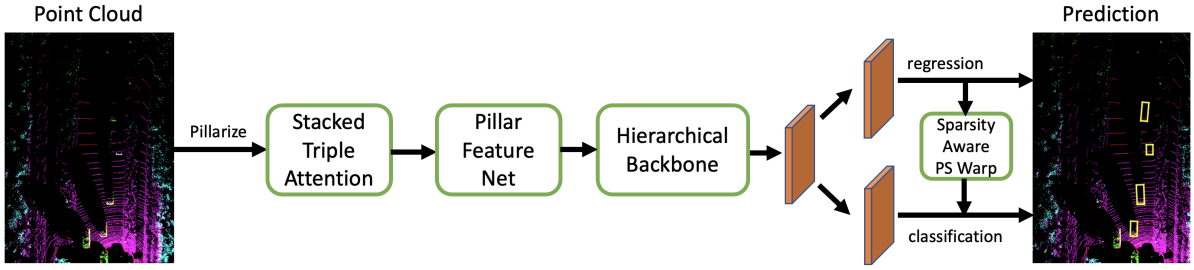


Figure 4.1: Overview of the proposed real-time LiDAR-based single-stage 3D object detector

The overflow of our *Pointpillars+* is depicted in figure 4.1. Firstly, the point cloud is pillarized and sent to the Stacked Triple Attention module for more robust and discriminative feature representation. Then the pillars are encoded and scattered back to a 2D feature map through the Pillar Feature Net. Subsequently, we replace the backbone of *Pointpillars* with an Attentive Hierarchical Backbone and use a Sparsity-Aware Part-Sensitive Warping module to alleviate the misalignment problem between predicted boxes and classification scores from SSD detection head. In comparison to *Pointpillars*, we introduce three extra additional modules: Stacked triple attention module, Hierarchical backbone, and Sparsity-aware part sensitive warping. The following subsections introduce these modules in detail.

4.1.2 Staked Triple Attention Block

The sparsity property of point clouds determines that most of the sampled points are background points and contribute little to the final detection results. Meanwhile, the performance of the model gets largely degraded when addtional noisy points are added to the ground truth bounding box. In order to alleviate the influence of noise and boost the robustness of the model, an attention mechanism is then introduced. Triple Attention Module [54] takes a set of voxels with shape of (B, D, P, N) as input , where B denotes batch size, N denotes the number of sampled points per pillar, P denotes the number of non-empty pillars, and D denotes the dimension of input feature.

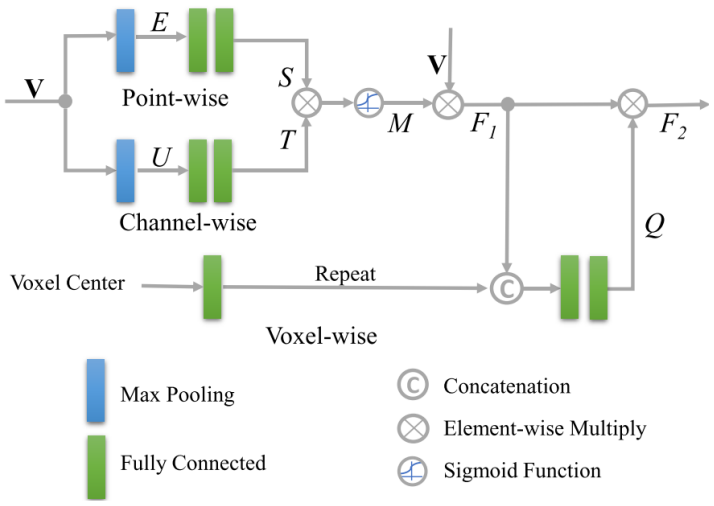


Figure 4.2: Triple Attention module. Source [54]

Triple Attention Module refers to the idea of *Squeeze and Excitation Net* [29] and performs *SE Attention* in the level of channels, points and voxels separately. To be more specific, the Triple Attention module captures discriminative channels, points, and voxels along (D, N, P) axis, respectively. As shown in the figure 4.2, after reallocating the weight for channels and points inside each voxel, TA module concate-

nates the center coordinates of each voxel together with the re-weighted feature map. The concatenated features are further cast to high dimensions and employed to re-weight the importance of voxels.

To fully exploit the advantage of the Triple Attention Module and boost its ability to capture important channels, points, and voxels, the author introduces stacked TA with two TA modules. The output of the first TA Module is concatenated to the raw input and then followed by an FC layer, which projects the feature to a higher dimension. The second TA module adds its output to its input which can be viewed as a residual structure like ResNet [25]. The output of the second TA module is also followed by an FC layer.

After imposing max-pooling operation on the feature of points inside voxels, stacked Triped Attention Bolck can be used to re-weight the features before Pillar Feature Net in *PointPillars*.

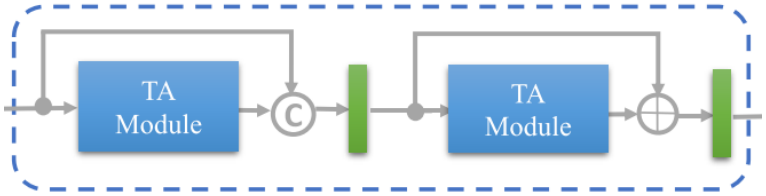


Figure 4.3: Stacked Triple Attention. Source [54]

4.1.3 Attentive Hierarchical Backbone

As shown in 4.2, the 2D backbone of *PointPillars* can be separated into two main parts. The first part extracts features with a top-down structure and produces feature maps at three different resolutions. The second part applies transposed convolution on the previous output separately and upsamples them to generate the feature maps with the same spatial resolution. The output of original backbone is a simple concatenation of three output feature maps.

The structure of the 2D backbone in *Pointpillars* is quite simple, and only a top-down structure is leveraged for feature extraction. One important reason is that the detection range of LiDAR-based 3D object detection is much wider than 2D images, and the size of the feature map is hence much larger. In this case, we can't introduce too many extra features at different spatial sizes to enrich the feature information because of the limit on computation resources and the requirement of real-time inference speed. Based on these considerations, two main modifications are employed.

Firstly, an extra bottom-up pathway in figure 4.4 is introduced to effectively and completely extract the features. The structure of new backbone is quite similar to

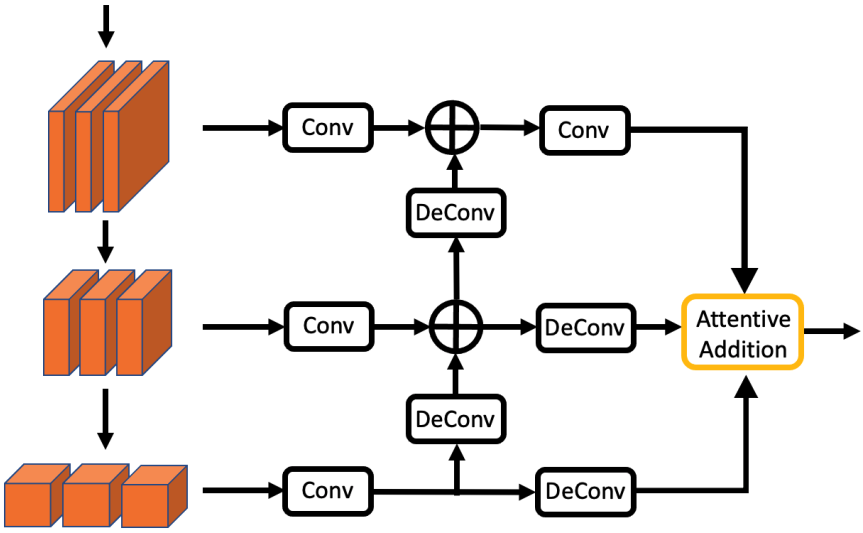


Figure 4.4: Attentive Hierarchical Backbone Structure

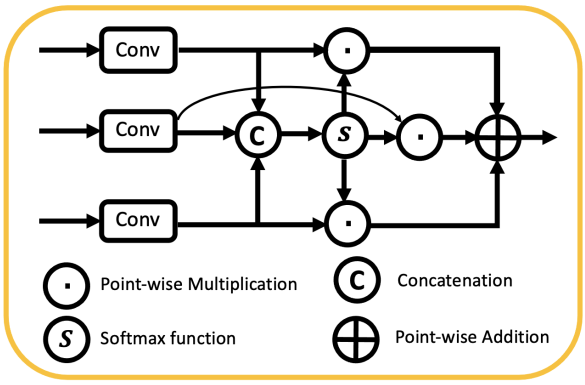


Figure 4.5: Attentive Addition Module Structure

the architecture of Feature Pyramid Network [21], but they are different in the lateral connection block. The lateral connection block is used to merge the feature maps from two pathways. In FPN, it uses upsampling operation with a stride of 2 in the bottom-up pathway and merges the corresponding feature map from the top-down pathway, which is followed by a $1 * 1$ convolution layer to adapt the channel dimension by an element-wise addition operation. In our case, we use deconvolution instead of upsampling operation in the lateral connection of FPN. The previous corresponding feature maps are resized to the same resolution by convolution and deconvolution blocks separately and sent to the Attentive Addition Module for fusion. The structure of Attentive Addition Module is depicted in figure 4.5.

4.1.4 Sparsity-Aware Part-Sensitive Warping

Part-Sensitive Warping (PS-Warp) is proposed in [13] to alleviate the misalignment between the predicted bounding boxes from the regression head and their corresponding confidence scores from the classification head. To more specific, the classi-

fication confidence is derived from the feature map in the position of its perception field, but the predicted bounding boxes usually have some offset to their default centers, which results in discordance.

The main idea of PS-Warp is to re-evaluate the confidence score of the predicted bounding boxes. It introduces a new head before the last convolution layer of the classification network, which is parallel to the SSD detection head. The PS-Warp head is composed of a 2D convolution layer with a kernel size of 3, a 2D batch normalization, and a ReLU layer. Then a $1*1$ 2D convolution layer is added at last to change the filter space dimensionality to K. The output of the PS-Warp head preserves the complete classification features before the SSD detection head, and its size is (K,H,W). H and W means the height and width of the output feature map, which is exactly the same as the feature map output in the SSD head. The dimension of channels K is the same as the number of parts that will be encoded in different positions of the bounding boxes.

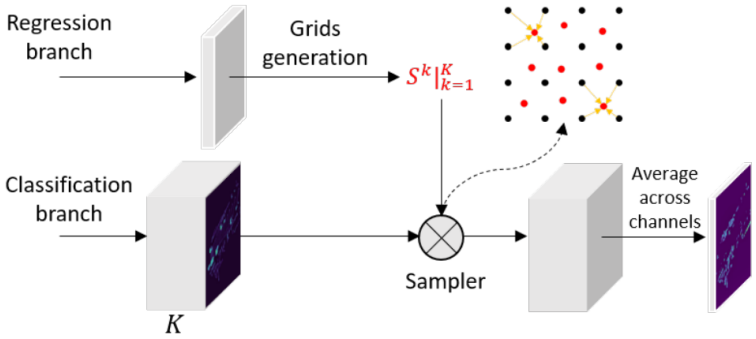


Figure 4.6: Part-Sensitive Warping. Source [13]

PS-Warp performs a spatial transform on the predicted bounding boxes. As shown in the figure 4.6, each predicted bounding box is partitioned into K parts by discretizing the whole box along x and y axes. It extracts the value from the center position of each sub-window on the PS-Warp feature map by bilinear interpolation and gets K corresponding scores. We use bilinear interpolation to estimate the value on the center of each subgrid. The re-scored confidence takes the mean value of K sampled points and can be formulated as follows:

$$C_p = \frac{1}{K}\sum_{k=1}^{K} \sum_{i\in\{\lfloor u^k \rfloor \lfloor u^k+1 \rfloor, j\in\{\lfloor v^k \rfloor \lfloor v^k+1 \rfloor\}} \chi_{ij}^k \times b(i,j,u^k,v^k)$$

where χ^k is the k^{th} channel of the PSWarp feature map and $b(i,j,u^k,v^k)$ is the kernel of bilinear sampler with the form $b(i,j,u^k,v^k) = max(1-|i-u|,0)\times max(1-|j-v|,0)$. The PS-Warp is inspired by PSRoIAlign [26] and has the advantage of speed by comparision. It takes much less computation time but achieves competitive performance. However, there is a premise when we perform spatial transforms on the predicted bounding boxes and consider all K sampled scores for re-weighting. It supposes the

object in the predicted bounding box is dense and most of the sampled points correspond to different positions on the surface of the object.

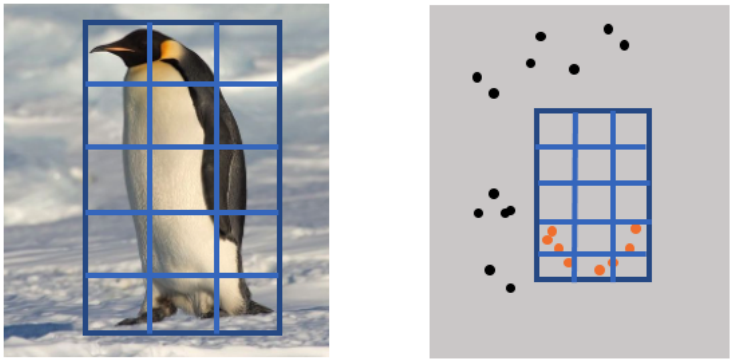


Figure 4.7: Bounding Box in Image and Point Cloud

As shown in Figure 4.7, almost all the sampled points in a 2D image are the foreground points and lie on the surface of the object. However, due to the sparsity property of point clouds, most of the sampled points in the bounding boxes are background points and don't contain much useful information. These background points may deteriorate the re-scoring performance of PS-Warp and bring unapparent improvements in comparison to negative bounding box samples. A simple idea is introduced to solve this problem. Instead of considering the values of all K sampled points, I apply Top K' strategy among K scores and take the average value of K' points as the final confidence. The new Sparsity-Aware Part-Sensitive Warping can be formulated as:

$$C_p = \frac{1}{K'} \sum_{k=1}^{K'} TopK'\{ \sum_{i \in \{\lfloor u^k \rfloor \lfloor u^k+1 \rfloor, j \in \{\lfloor v^k \rfloor \lfloor v^k+1 \rfloor\}} \chi_{ij}^k \times b(i,j,u^k,v^k)\}$$

4.1.5 Two Extra Proposed Modules

Multi-view PointPillars

The Pillar Feature Net in *PointPillars* discretizes the 3D detection space to several vertical columns and embeds the sampled points inside each vertical column to a high-dimensional vector. In this way, the feature of each encoded pillar only contains the information from bird eye view and loses the horizontal correlations between different pillars.

3D convolution in *SECOND* can extract features of the spatial correlations both vertically and horizontally and enlarge the perceptive field in 3D space. However, 3D convolution consumes more time and takes up larger GPU memory in comparison to 2D convolution. In order to leverage the spatial information of points sufficiently

and effectively, we further introduce pillars from the front view to enrich the feature representation of *Pointpillars*.

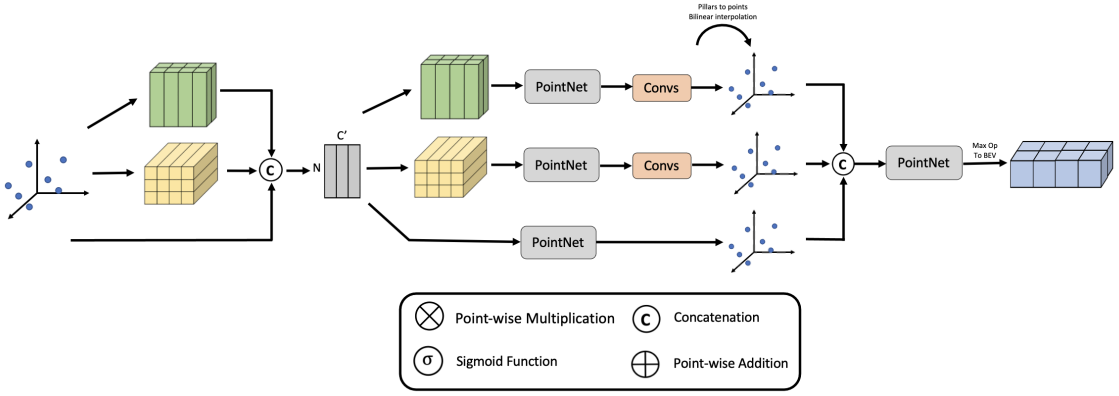


Figure 4.8: Multi-view Feature Encoding Net

Figure 4.8 shows the workflow of Multi-View Feature Encoding Net. It firstly utilizes the pillar encoder from both bird-eye view and the front view, and establishes 4 mapping function between points and voxels, i.e. $F_V^{bev}(p_i)$, $F_V^{front}(p_i)$, $F_P^{bev}(v_i)$ and $F_P^{front}(v_i)$. Then it concatenates the features from two views together with raw points to generate a point-wise feature map with a shape of (N, C). In order to fully exploit the spatial information in two views, we further project all the points back to the front view and the birds-eye view according to$F_V^{bev}(p_i)$, $F_V^{front}(p_i)$ respectively. These pillars are next followed by a simplified *Pointnet* block and a single 2D convolution block to extract their spatial correlations and generate two 2D feature maps separately. Finally, we scatter two feature maps back to point-wise format with bilinear interpolation by pillar-to-point mapping function $F_P^{bev}(v_i)$ and $F_P^{front}(v_i)$ and concatenate the point-wise features of bird-eyes view, front view, and raw data. The multi-view point-wise features could then be mapped to a 2D bird-eye feature map by exploiting maximum operation to all the points within a pillar.

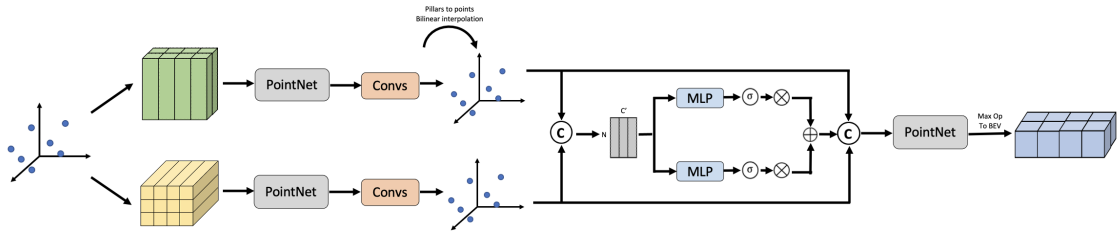


Figure 4.9: Attentive Multi-view Feature Encoding Net

As illustrated above, we use point-wise operations when we fuse the features from two views. It simplifies the process of feature fusion but, in the meanwhile, brings the extra problem that most of the points are background points and won't contribute a lot to final detection results. The attentive mechanism is then introduced to increase

the importance of useful points.

Figure 4.9 shows the overflow of attenteive mult-view feature encoding net. Different from figure 4.8, the raw point clouds are directly pillarized in two views and processed by *Pointnet* and two CNN backbones. Pillar-wise features are then scatter back to point-wise feature with bilinear interpolation according to mapping functions $F_P^{bev}(v_i)$ and $F_P^{front}(v_i)$. Then we use a point-wise attentive fusion block to adaptively allocate weight to different points from two views. The MLP in the block is composed of a linear layer, a batch normalization layer and a ReLU layer. Suppose two input point-wise features are $\mathbf{F}_{bev}$ and $\mathbf{F}_{front}$, and output is $\mathbf{F}_{out}$, the point-wise attentive fusion block can be formulated as follows:

$$\mathbf{F}_c = Concat([\mathbf{F}_{bev}, \mathbf{F}_{front}])$$
$$\mathbf{F}_{a.bev} = \mathbf{F}_{bev} \otimes \sigma(MLP_{bev}(\mathbf{F}_c))$$
$$\mathbf{F}_{a.front} = \mathbf{F}_{front} \otimes \sigma(MLP_{front}(\mathbf{F}_c))$$
$$\mathbf{F}_{out} = Concat([\mathbf{F}_{bev}, \mathbf{F}_{front}, \mathbf{F}_{a.bev} \oplus \mathbf{F}_{a.bev}])$$

where *Concat* denotes channels concatenation. $\otimes$ means element-wise multipilication. σ means sigmoid activation function.

Double Attentive Dynamic Voxelization

Normal voxelization divides the point clouds into K spatially evenly distributed voxels, where T points will be sampled within each voxel. If the number of non-empty voxels or the number of points inside one voxel exceeds buffer size, only K voxels and T points will be kept by sampling. On the contrary, if the number of points or voxels is less than its buffer capacity, the rest volumes will be padded with zero.

The fixed buffer size of the normal voxelization technique and its hard property brings some unavoidable drawbacks. Firstly, some useful information may get discarded after the sampling process when voxel or points exceed the pre-defined buffer size. Secondly, the sampling process is non-deterministic, which will result in unstable voxel embedding, and consequently affect the final detection performance. Thirdly, when voxels or points are less than the fixed size, extra zero-padding contains no useful information and not only brings unnecessary computation cost but also slows down the running speed of the model.

Dynamic voxelization [56] is proposed to replace the traditional hard voxelization. Instead of setting a fixed buffer capacity for voxels and points, it keeps all the voxels and points after the grouping process. It formulates the relationship between points and voxels as a many-to-one mapping, i.e., each voxel corresponds to all the points inside it.

Figure 4.10 shows the overflow of dynamic voxelization. It takes raw point cloud with shape (N, C) as the input, where N is the number of points, and C is the di-

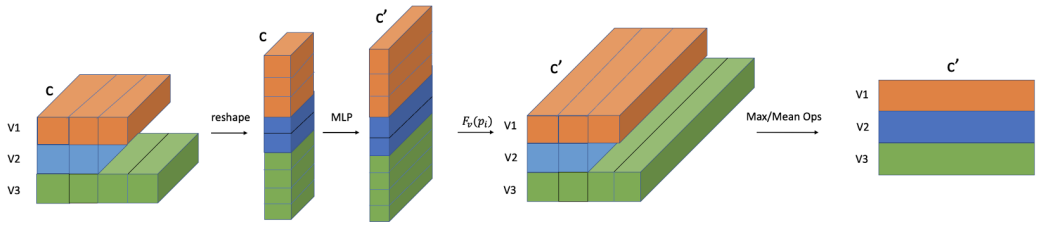


Figure 4.10: Dynamic Voxelization Overflow

mension of LiDAR input. Dynamic voxelization firstly establishes a mapping function $F_V(p_i)$ from points to voxels according to their spatial coordinates. After that, the point-wise input will be sent to a simple shared-weight MLP and embedded into a higher dimension from C to C'. The shared-weight MLP is composed of a linear layer, a batch normalization layer, and a ReLU activation layer. The embedded points with size (N, C') is scattered back to each voxel by referencing mapping $F_V(p_i)$. In practice, we store the voxel as the shape of (K, T_{max}, C'), where K is the number of non-empty voxels, T_{max} is the maximum number of points inside the voxels. Finally, we apply maximum or mean operation along the axis of points T_{max} and get the final encoding feature map with size (K, C').

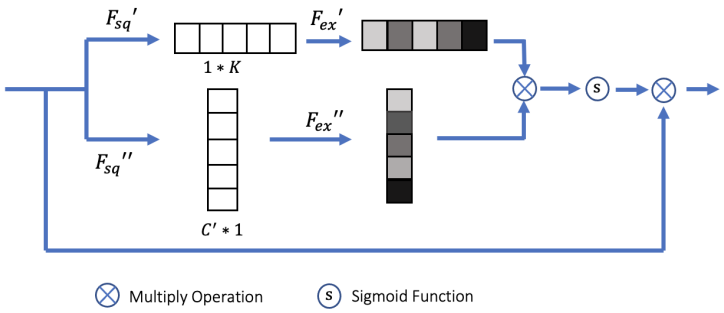


Figure 4.11: Double Attentive Operation

After obtaining the encoded point-wise features, *Squeeze-and-Excitation Block* [29] is imposed along both channel and voxel axis. Referring to the idea of Triple Attention Module [54], we multiply the channel-wise attention by voxel-wise attention. F_{sq}' and F_{sq}'' represents max-pooling operations in N and C' axis. F_{ex}' and F_{ex}'' are composed of two fully connected layers, in which the first one is followed by batch normalization and ReLU function, while the second one is a single linear layer. Suppose the input as I, output as O, the whole triple attentive module could be formulate as:

$$O = \sigma(M \otimes N) \otimes IM = W_2\delta(W_1 MaxPool(I))N = W_2'\delta(W_1' MaxPool(I^T))$$

4.2 Proposed Semi-supervised Solution

In this section, we introduce the details of our proposed semi-supervised algorithm SAPL in the thesis.

4.2.1 Pseudo Labeling

Pseudo labeling is a simple and efficient method of semi-supervised learning (SSL). The main idea of pseudo labeling is to boost the performance of a model that is already trained with labeled data by means of unlabeled data. As shown in figure 4.12, the process can be divided into three main steps: Firstly, the model is trained with the labeled data. Then the unlabeled data are sent to the trained model to generate predictions. Finally, the predicted pseudo labels are combined with the original labeled data, with which the model is re-trained.

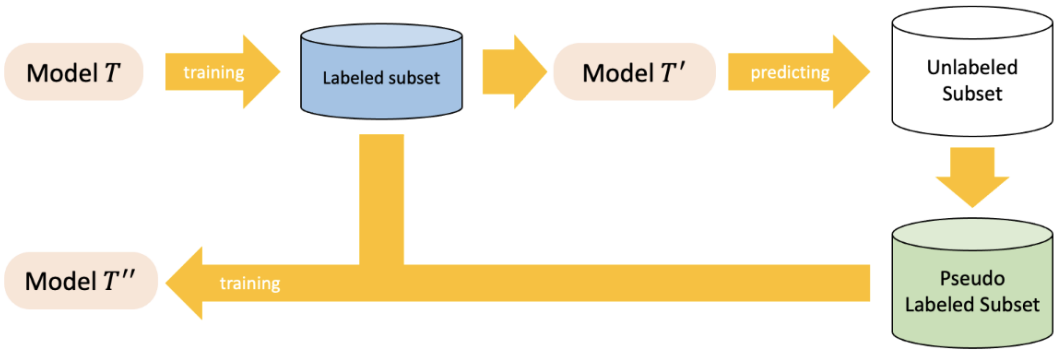


Figure 4.12: The overflow of the pesudo labeling technique

Paper [18] points out that the samples in a high-density region have a greater probability of being in the same class for the classification task. Similarly, when we apply pseudo labeling to the object detection task, the key task is to find a suitable threshold for the classification score. If we set the classification threshold too high, many ground truth will be marked as negative samples, which lead to a low recall rate. However, many false-positive samples will appear if the threshold is set too low, which results in low model precision. In addition, the success of the pseudo labeling trick in object detection tasks extremely relies on the high precision of the trained model with labeled data. If the basic trained model in the first step is not precious enough, the predicted pseudo labels could be inaccurate, which may even lead to the degradation of the model.

Practically, the pseudo labeling trick has been proven to be effective in the 2D object detection task. We further introduce it to solve the label shortage problem in our 3D object detection case.

4.2.2 Domain Adaption from KITTI Dataset

The domain of the KITTI and Providentia datasets is not independent and identically distributed. Hence, we need to employ transfer learning techniques to mitigate the discordance between source and target domain. Domain Adaption (DA) is one of the most representative methods in transfer learning, which boosts the model's performance on the target domain by leveraging rich and similar information from the source domain.

We adopt two effective domain adaption methods in our Statistical-Aware Pseudo Labeling algorithm: Few-Shot Fine-Tuning and Statistical Normalization. Few-Shot Fine-Tuning leverages a few numbers of labeled examples from the target domain to tune the object detector that is already trained on the source domain. Paper [44] points out that the detector tends to predict the boxes that have similar size in the source domain instead of predicting the real physical size in the target domain. Statistical Normalization (SN) algorithm is then proposed. It assumes that the statistical distribution of size in the target domain is available. SN resizes all bounding boxes and the corresponding point cloud to sizes that are similar to the size of objects in the source domain by adding $(\Delta h, \Delta w, \Delta l)$, where Δh, Δw, and Δl are the difference of mean height, width, and length value respectively between the source and target domain. As shown in figure 4.13, SN enlarges or shrinks the ground truth bounding box and the associated point clouds. It essentially reduces the distance of distribution between source and target domain.

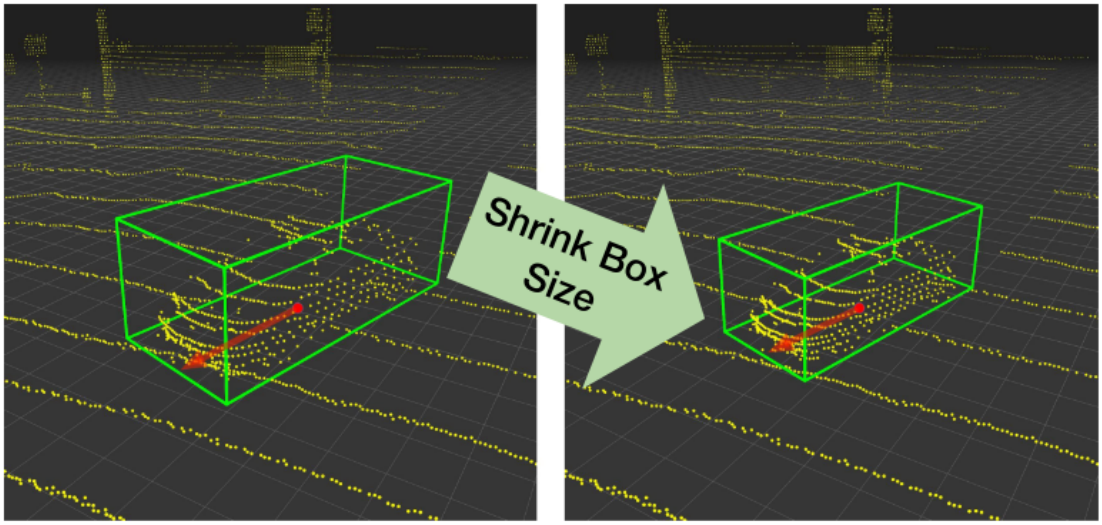


Figure 4.13: The example of statistical normalization. Source [44]

4.2.3 Statistical-Aware Pseudo Labeling for 3D Object Detection

In this section, we introduce our proposed semi-supervised method for 3D object detection: Statistical-Aware Pseudo Labeling (SAPL). Our SAPL algorithm is suitable for

the practical scenario where the size distribution of target domain, few labeled, and a large number of unlabeled target samples are available. It combines the Pseudo Labeling method from semi-supervised learning, Few-Shot Fine-Tuning, and Statistical Normalization together and formulates an effective algorithm.

Algorithm 1 Statisical-Aware Pseudo Labeling Algorithm

Input: Labeled source data $\{x_s^l, y_s^l\}$, labeled target data $\{x_t^l, y_t^l\}$, unlabeled target data x_t^u, raw teacher model α_r, raw student model β_r

Output: student model β_f

1: Investigaing the statistical distribution of y_t^l and applying statistical normalization to source data $\{x_s^l, y_s^l\}$
2: Training the teacher model α_r on the normalized source data $\{x_n^l, y_n^l\}$
3: Employing few-shot fine-tuning on the pre-trained teacher model α_p and getting α_f
4: Generating pesudo label y_t^p for unlabeled target data x_t^u using the teacher model α_f
5: Training the raw student model β_r on the normalized source data $\{x_n^l, y_n^l\}$ and obtaining pre-trained student model β_p
6: Transferring the student model β_p to target domain by fine-tuning on $\{(x_t^l, x_t^u), (y_t^l, y_t^p)\}$ and getting student model β_f
7: **return** β_f;

Algorithm 1 shows the specific precedures of SAPL. Given labeled source data $\{x_s^l, y_s^l\}$, labeled target data $\{x_t^l, y_t^l\}$, unlabeled target data x_t^u, raw teacher model α_r, raw student model β_r, the mean and variance value of vehicle's size in target domain $\{u_k, \sigma_k^2\}$, $k \in \{l, w, h\}$ where l, w, h denote length, width and height of 3D boxes. SAPL first apply Statitical Normalization to the labeled source dataset $\{x_s^l, y_s^l\}$ based on $\{u_k, \sigma_k^2\}$ to resize the size of objects in source domain. Then we train both teacher and student model α_r on the normalized source dataset $\{x_s^l, y_s^l\}$. Subseqently, we fine-tune the teacher model with few labeled target data $\{x_t^l, y_t^l\}$ and generate pesudo labels of unlabeled target data y_t^p with the teacher model after few-shot fine-tuning α_f. Finally, we re-train the student model on combined dataset $\{(x_t^l, x_t^u), (y_t^l, y_t^p)\}$.

Figure 4.14 illustrates the workflow that applies the SAPL algorithm to our thesis, where KITTI is used as source domain data, and Providentia dataset is utilized as target domain data.

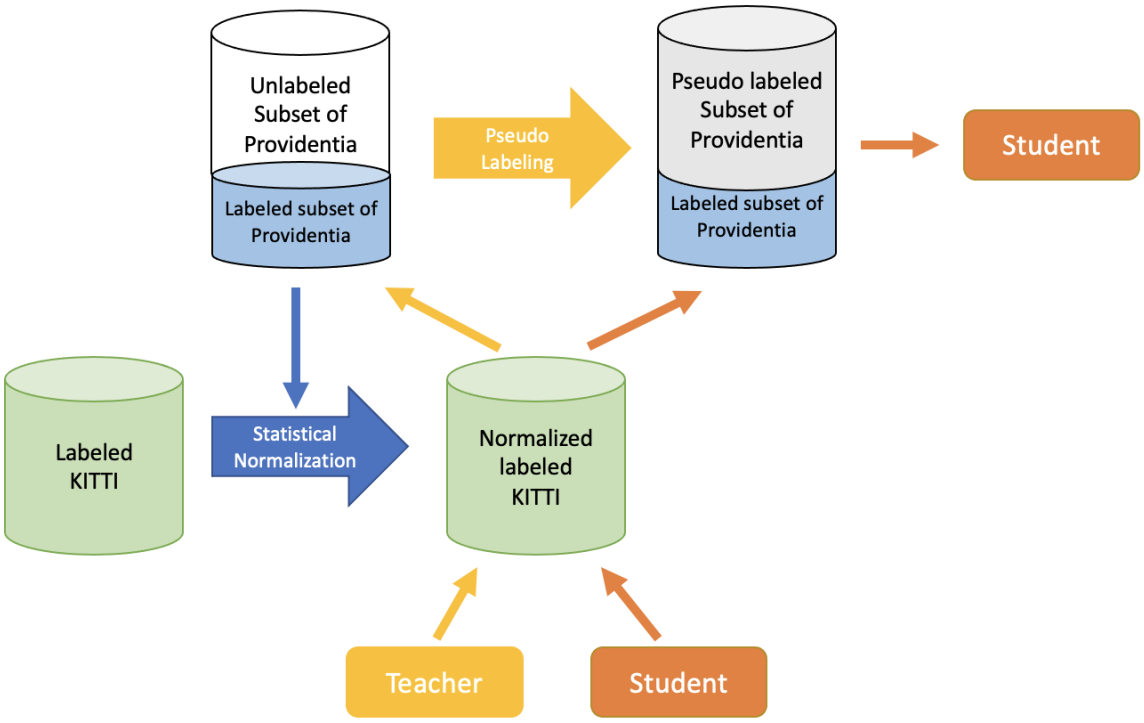


Figure 4.14: The workflow of Statisical-Aware Pseudo Labeling on Providentia Dataset

Chapter 5

Experimental Details

In this chapter, we describe the details of *Pointpillars+* in section 5.1 and SAPL algorithm in section 5.2 during our experimental and implementation process.

5.1 Pointpillars+ Implementation Details

5.1.1 Dataset

KITTI dataset contains 7,481 training samples and 7,518 testing samples. Following the conventional practice, we split the training sets into 3,712 samples for model training and 3,769 samples for validation. We consider three difficulty levels in KITTI and calculate the model's performance under these circumstances separately. We follow the new evaluation rule in the KITTI benchmark and calculate all the average precision values with 40 recall positions. KITTI comprises many categories such as Car, Van, Truck, Pedestrian, Person, and Cyclist, but as illustrated in figure 5.1, the number of Cars, Vans, and Pedestrians overwhelms the other classes. Considering pedestrian is a rare category on the highway, we only evaluate the mAP of Car and Van categories in our following experiments of *Pointpillars+* on KITTI.

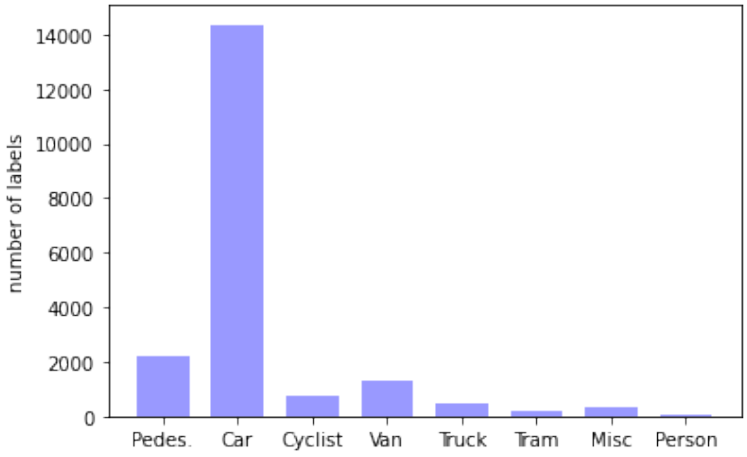


Figure 5.1: Number of different labels in KITTI

5.1.2 Data Augmentation

Data augmentation is an important method to solve the label shortage problem. It not only enriches training sets that improve the model's generalization ability but also brings additional noisy samples to boost the model's robustness. We introduce the data augmentation tricks that are applied in our *Pointpillars+* as follows:

We utilize some classic data augmentation tricks for training our *Pointpillars+* such as cut-and-paste strategy following *SECOND*. In specific, we first create an object pool by collecting the ground truth 3D bounding boxes for all classes and the associated points that fall inside the boxes. Subsequently, we randomly sample some instances for each category and paste them into the current point cloud data. The maximum selecting number of cars is 15. The pasting strategy follows the physical rule that the 3D bounding boxes of two objects have no intersection. Besides, we also apply local rotation and translation to all the objects, including pasted samples. Local rotation is applied to each object along the vertical y-axis with noise drawn from the uniform distribution $\mathcal{U}(-\pi/4, \pi/4)$, while the noise of local translation for x, y, and the z-axis is separately drawn from a Gaussian distribution $\mathcal{N}(0, 0.5)$. In addition, we apply global random flip along the x-axis with Bernoulli probability distribution $p = 0.5$, and global rotation along yaw axis with same uniform noise distribution $\mathcal{U}(-\pi/4, \pi/4)$ as in local rotation. We also scale the whole point cloud scene and the annotations with the scaling factor uniformly selected from $[0.95, 1.05]$.

5.1.3 Training

Following *Pointpillars*, we set the detection range of *Pointpillars+* as (0m,70.4m) for the x-axis, (-40m,40m) for y-axis and (-3m,1m) for z-axis respectively. Both the points outside this detection range and the points that are invisible in the corresponding image will get dropped. We also limit the raw orientation range of ground truth bounding boxes from $-\pi$ to π.

The resolution of input pillars along the x and y-axis is set as 0.2m. We randomly sample 10000 pillars at most with FPS and set the maximum number of points per pillar to 40. Hence, the size of the input feature map is $C * 352 * 400$, where C is the dimension of the augmented input. The raw input point is a four-dimensional vector with the 3D location information x, y, z and reflectance r. We then calculate the relative coordinate of each point to the arithmetic mean center of all the points inside its associating pillar: $(x - x^c, y - y^c, z - z^c)$, and the relative x, y coordinate of each point to the spatial pillar center of its corresponding pillar: $(x - x^s, y - y^s)$. Therefore, all the augmented points are nine-dimensional vectors and $C = 9$.

We train *Pointpillar+* for 80 epochs using the Adam optimizer. The initial learning rate, decay weight of Adam, and the batch size are set to $3 * 10^{-3}$, 10^{-2} and 2, respectively. We apply L2 norm clipping to the gradient during the training process

with threshold 10 and one cycle cosine annealing strategy to the learning rate.

We set the positive and negative IOU matching threshold for Cars as 0.6 and 0.45, respectively. When calculating IOU between the ground truth and anchors, we first convert the rotated boxes to their nearest horizontal bounding boxes and then take the 2D IOU value in the bird eye view. The anchors whose IOU threshold with any ground truth bounding box exceed the positive matching threshold are assigned as positive anchors. The anchors that have the highest IOU with any ground truth are also set as positive to make sure that no ground truth will be missed during the training process. We set the anchors as positive samples if their highest IOU with all ground truth is under the negative matching threshold. The loss of all other anchors is not considered during the back propagation period.

5.1.4 Network

Attentive Hierarchical Backbone

In the Attentive Hierarchical Backbone, we characterize the structure with a series of blocks (K, L, S), where K is the size of the kernel, L denotes the output channels, and S means the up or down stride of the block. We adopt three stacked convolutions for each feature extraction stage in the top-down network, where the first block consists (3,64,2), (3,64,1), (3,64,1), the second block consists (3,128,2), (3,128,1), (3,128,1), and the third block consists (3,256,2), (3, 256,1), (3,256, 1) respectively. We use a single deconvolution layer (3,128,2) to increase the resolution of the feature map and keep their output shapes the same for element-wise addition in the bottom-up pathway. Both three input and the output of attentive fusion block are of the same shape $[B, 128, 200, 176]$, where 128 is the number of channels, and $(200, 176)$ are the resolution of the feature map. The convolution layer in the attentive fusion module comprises a sequence of 2D Conv (1,128,1) and 2D batch normalization layer.

Sparsity-Aware Part-Sensitive Warping

The head of PS-Warp has two convolution layers with a number of channels 28, where the first convolution with 3×3 kernel is followed by a batch normalization layer and ReLU function, and the second layer has 1×1 kernel. The shape of the output feature map is $[B, 28, 200, 176]$, where B denotes the batch size, 28 is the number of 4×7 partitioned sub-windows, 200 and 176 are the height and width of the output feature map.

We set both positive and negative IOU matching threshold for the PS-Warp head to 0.7 during the training process. The target assignment strategy follows the rule in the dense head except the similarity metric switching from nearest IOU similarity to rotated IOU 3D similarity. The positive matching threshold of PS-Warp gets increased compared to the threshold of SSD head to boost the quality of proposals for confidence re-scoring.

When applying Sparsity-Aware PS-Warping, we train the head following the settings in the original paper with sampling grid $K = 28$. We only utilize the Top K' strategy during the inference period to filter out the most representative positions in the bounding boxes and further improve the re-scoring performance.

Stacked Triple Attention Module

The input of stacked triple attention module is a set of pillars with a shape of $[P, N, C_{in}]$, where P is the number of non-empty pillars, N is the maximum number of points inside each pillar, C is the augmented input. As mentioned above, we set $N = 40$, $C_{in} = 9$. The number of P varies for different point cloud input, and its maximum number is 10000.

We set the boosting channel dimension C_{boost} to 64 and the reduction ratio r to 8. It means the output's shape of the first triple attention module is boosted from $[P, 40, 9]$ to $[P, 40, 64]$. The fully connected layer in the TA module is composed of two linear layers and an activation function, and the size of the bottleneck between two linear layers is only the dimension of input divided by 8, which efficiently reduces the number of parameters.

Multi-view Pointpillars

The detection range of Multi-view *Pointpillars* is (0m, 69.12m), (-39.68m, 39.68m), (-3m, 1m) for x, y, z axis on the bird eye view, and (0m, 69.12m), $(-\pi, \pi)$, (-3m, 1m) for (ρ, φ, z) on the cylindrical view respectively. We follow the cylindrical view settings in paper [45], and encode the coordinate of front view as follows:

$$\rho = \sqrt{x^2 + y^2}, \quad \varphi = arctan\frac{y}{x}, \quad z = z$$

We use point-wise feature concatenation to fuse the spatial features from the front view and bird eye view. The input of the model is a set of points with a shape of $[B, N, C_{in}]$, where B is the batch size, N is the number of sampled points for each scene, C_{in} is the dimension of input features. We use $N = 20000$ and $C_{in} = 4$ in our model. Then we augment the point-wise input features for two views by calculating the relative coordinate to pillar's center $[B, N, 3]$, the number of points in each pillar $[B, N, 1]$, positions after view transformation $[B, N, 3]$, the covariance of points $[B, N, 9]$, and relative coordinate to pillar's centroid $[B, N, 3]$.

Multi-view *Pointpillars* concatenate all the augmented features to a tensor with a shape of $[B, N, 45]$ and apply a *Pointnet* (45,64) to embed the raw features. We then scatter the embedded features back to the front view and bird eye view and apply max operation to the point-wise feature inside each pillar to get the respective feature map. The shape of the feature map is $[432, 496, 64]$ on the bird eye view and $[128, 120, 64]$ on the front view. During implementation, the gather and scatter

operation in $PyTorch$ can not satisfy our requirements. Therefore, we utilize an effective third-party library $PytorchScatter$ for the scatter-relevant operations such as scatter-max, scatter-add, scatter-mean.

Anchors

Pointpillars+ is an anchor-based object detector. Both anchors and ground truth 3D bounding boxes in *Pointpillars+* are encoded as 7-dimensional vectors $(x, y, z, w, l, h, \theta)$, where x,y,z represent the location of the box's center, w,l,h denote the width, length, and height of the box respectively, θ denotes the rotation angle of the box along the upright axis.

In the regression branch, we train and predict the residuals of each value. The encoding of localization parameters are defined as follow:

$$\Delta x = \frac{x^g - x^a}{d^a}, \quad \Delta y = \frac{y^g - y^a}{d^a}, \quad \Delta z = \frac{z^g - z^a}{d^a}$$

$$\Delta w = log\frac{w^g}{w^a}, \quad \Delta h = log\frac{h^g}{h^a}, \quad \Delta l = log\frac{l^g}{l^a}$$

$$\Delta\theta = sin(\theta^g - \theta^a),$$

where the uperscript a and g represent anchors and ground truth seperately, and d^a is used for residuals normalization with $d^a = \sqrt{(w^a)^2 + (l^a)^2}$.

We set two anchors for each cell grid in the output feature map of *Pointpillars+*. The rotation angles of the two anchors are 0 and 90 degrees. We set the width, length, height of the Car's anchor size as 1.6m, 3.9m, and 1.56m, respectively. The center of the Car's anchor is located at $z = -1m$.

5.1.5 Loss Function

The loss of *Pointpillars+* comprises four main parts: object classification loss $\mathcal{L}_{cls}$, box direction classification loss $\mathcal{L}_{dir}$, box regression loss $\mathcal{L}_{reg}$, and PS-Warp refinement loss $\mathcal{L}_{warp}$. The totoal loss is formulated as follow:

$$\mathcal{L} = \beta_{cls}\mathcal{L}_{cls} + \beta_{dir}\mathcal{L}_{dir} + \beta_{reg}\mathcal{L}_{reg} + \beta_{warp}\mathcal{L}_{warp}$$

As mentioned in section 2.2, the focal loss can effectively mitigate the imbalance between hard and easy samples during the training process. Therefore, we choose sigmoid focal loss for both the object classification task and score refinement task:

$$\mathcal{L}_{cls}(p_t) = -\alpha_t(1-p_t)^\gamma log p_t,$$

$$\mathcal{L}_{warp}(p_t) = -\alpha_t(1-p_t)^\gamma log p_t,$$

where p_t is the predicted class probability for an anchor after the activation of the sigmoid function, α controls the weight between positive and negative samples, while γ controls the weight between easy and hard samples. Following the setting in the original paper, we use $\alpha = 0.25$ and $\gamma = 2$.

We choose smooth L1 loss for regression. It can suppress extremely large gradient at the beginning of training and guarantees low gradient value during convergence period.

$$SmoothL1(x) = \begin{cases} |x| - 0.5, |x| > 1 \\ 0.5x^2, |x| \leq 1 \end{cases}$$

The loss of our localization regression task is formulated as:

$$\mathcal{L}_{reg} = \sum_{b \in (\Delta_x, \Delta_y, \Delta_z, \Delta_w, \Delta_h, \Delta_l, \Delta_\theta)} SmoothL1(b),$$

where $\Delta_x, \Delta_y, \Delta_z, \Delta_w, \Delta_h, \Delta_l, \Delta_\theta$ are the forementioned encoded residual value between anchors and ground truth.

The direction classification task is introduced to solve the issue that regression loss of the boxes with opposite directions are the same after adding sinuous function to residual rotation angle $\Delta\theta$. We use softmax with cross-entropy loss for direction classification $\mathcal{L}_{dir}$.

Following the settings in SA-SSD, we set $\beta_{cls} = 1$, $\beta_{dir} = 0.2$, $\beta_{reg} = 2$, and $\beta_{warp} = 1$ as the coefficients of our training loss.

5.1.6 Evaluation

We evaluate our *Pointpillars+* detector on an Nvidia RTX 2080 super GPU. We set the IOU threshold to 0.3 for non-maximum suppression during the inference process. When introducing SAPS-Warp, we filter the boxes with low re-score classification confidence again under thr = 0.1.

We use two different object splitting methods to evaluate the performance of *Pointpillars+* on KITTI's validation dataset. One method follows the rule of KITTI's benchmark, which divides the objects into easy, moderate, and hard cases based on their truncated and occluded conditions. Another splitting method follows the settings in paper [44], which evaluate the model under various distance-based ranges starting from the object to the ego-car. Objects under distance-based evaluation are filtred with the same truncation and occlusion thresholds as KITTI hard case. We evaluate the objects at three different ranges of distance: 0-30m, 30-50m, and 50-70m, respectively.

5.2 SAPL Implementation Details

5.2.1 Providentia Dataset

Our customized Providentia dataset is recorded by one Ouster OS1-64, gen 2 sensor. The specification of the sensor is listed in the table 2.1. We use the annotation tool 3DBAT for ground truth labeling. Figure 5.2 and figure 5.3 shows two frames from our Providentia dataset on the bird eye view and front view. We record four 1 minute sequential frames on the highway with 10 Hz and extract each fourth frame from their rosbag files. In this way, around 600 frames are extracted and manually labeled with three main categories: Cars, Vans, and Trailers. Considering the precision of labeling, we only select 300 labeled frames from them and use the selected part for training and evaluation in the thesis. We split the Providentia dataset into three parts: 100 labeled frames for training, 200 labeled frames for evaluation, and 300 unlabeled frames for pseudo labeling.

As shown in the figure 5.3, we install the sensor on the gantry bridge that is around six meters above the ground plane. The installation position of the sensor determines that most of the foreground points in the Providentia dataset lie on the front top or back part of the vehicles. Hence, we can not label the 3D bounding boxes of most of the objects completely preciously based on these gathered points and need to estimate their length and height during the labeling process. Besides, the orientations of objects in our Providentia dataset have two options in most cases: 0 or π, because the vehicles always go straight along the road on the highway.

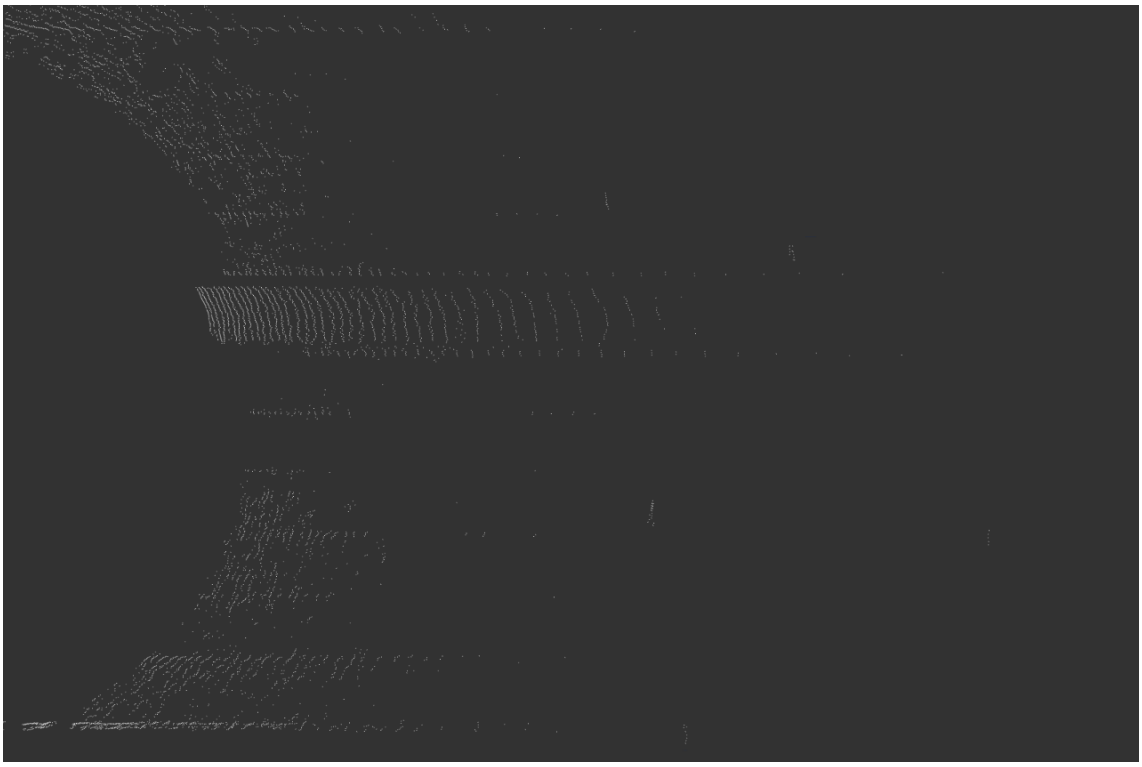
5.2.2 Network

We choose *Pointpillars+* introduced in section 4.1 as both our student and teacher network. We set the NMS confidence for testing to 0.2 and the default anchor size to mean value in table 5.2. We keep most other settings the same as in Pointpillars+.

We train the model on the car and van categories and evaluate their mean average precision on the bird eye view with IOU threshold 0.25 because the number of trailers in our Providentia dataset is insufficient for producing a convincing evaluation result. As mentioned in section 5.2.1, we set the IOU threshold lower than KITTI's setting due to the perspective particularity of our Providentia dataset. We focus on the average precision on BEV instead of 3D AP because of the ambiguity of vehicles' heights and their positions along the z-axis.

5.2.3 Statistical Normalization

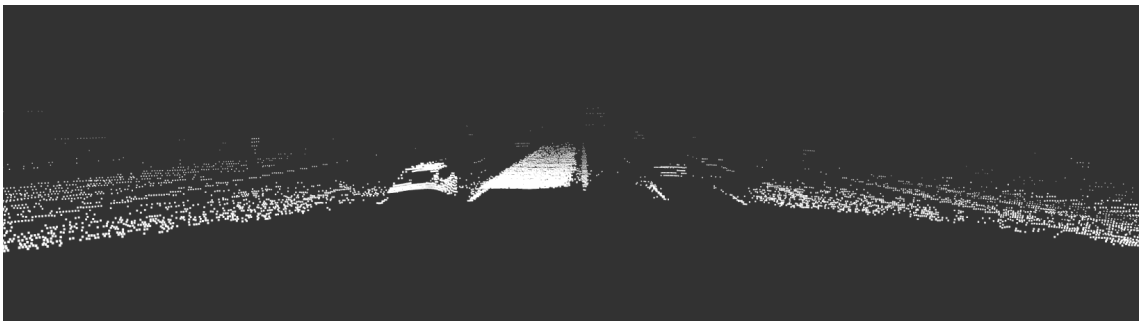
Statistical Normalization is an important component in our SAPL algorithm, where the statistics of vehicle size for both source domain and target domain are required.



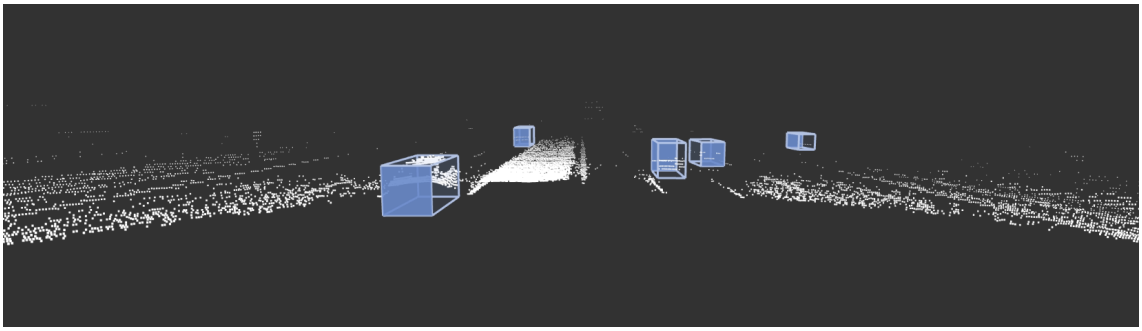
(a) Raw point cloud data of a frame from Providentia dataset on the bird eye view

(b) Labeled Point cloud data of a frame from Providentia dataset on the bird eye view

Figure 5.2: A frame from Providentia dataset on the bird eye view



(a) Raw point cloud data of a frame from Providentia dataset on the front view

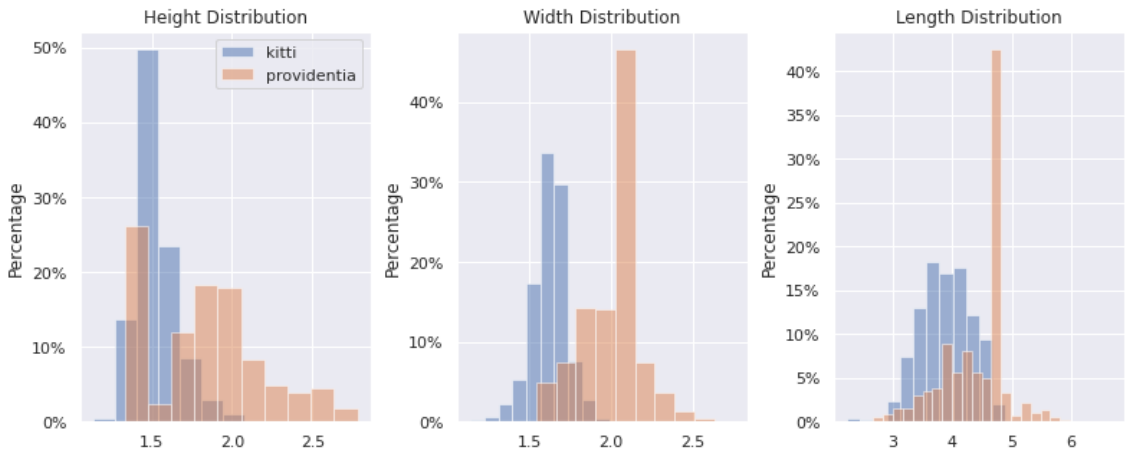


(b) Labeled Point cloud data of a frame from Providentia dataset on the front view

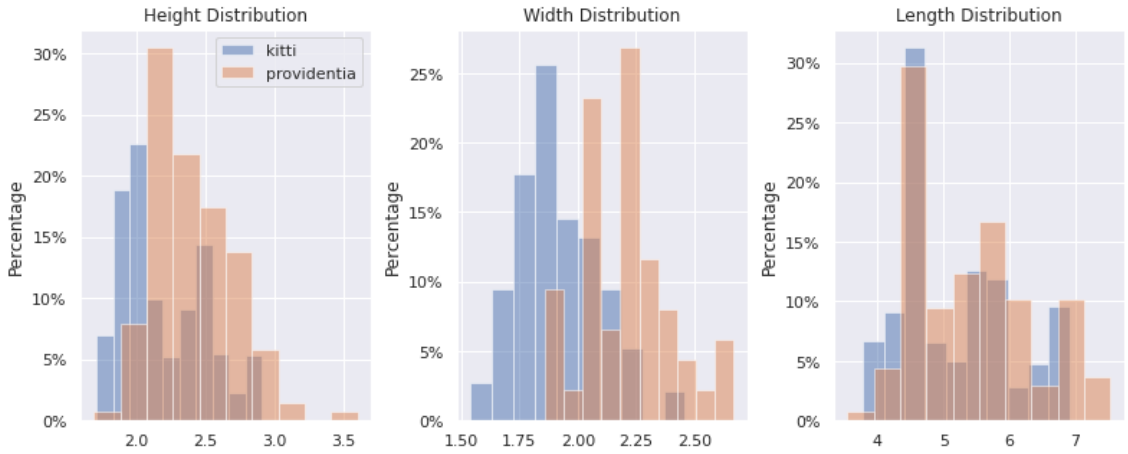
Figure 5.3: A frame from Providentia dataset on the front view

Here we choose KITTI as the source data for transfer learning and our customized Providentia dataset as target data. We use the training dataset from KITTI and all 300 labeled frames of our Providentia dataset for obtaining the statistical distribution of vehicles' size.

Figure 5.4 depicts the 3D bounding box size distribution of car and van categories for both KITTI and our customized Providentia dataset. It is obvious that the distributions of KITTI and Providentia are quite close, but the size range of the Providentia dataset is relatively larger than KITTI's size range. Table 5.1 and table 5.2 display the mean and variance of 3D bounding box size in two datasets. The mean and variance value of the Providentia dataset under almost all sizes are larger than KITTI's value. According section 4.2.2, we add $(\Delta h, \Delta w, \Delta l) = (0.33, 0.38, 0.47)$ for car and $(\Delta h, \Delta w, \Delta l) = (0.24, 0.31, 0.22)$ for van to generate a new rescaled KITTI dataset.



(a) 3D Bounding Box Size Distriburion of Car



(b) 3D Bounding Box Size Distriburion of Van

Figure 5.4: Comparision of 3D Bounding Box Size Distriburion between Providentia and KITTI dataset

Mean/Var (m)	Height	Width	Length
Car	1.53/0.14	1.62/0.11	3.89/0.44
Van	2.19/0.32	1.91/0.18	5.15/0.85

Table 5.1: Size of 3D Ground Truth in KITTI dataset

Mean/Var (m)	Height	Width	Length
Car	1.86/0.31	2.00/0.18	4.36/0.56
Van	2.43/0.32	2.22/0.20	5.37/0.85

Table 5.2: Size of 3D Ground Truth in Providentia dataset

Chapter 6

Results

This chapter presents the experimental results of the Pointpillars+ and SAPL algorithms. We show the experimental results of five modules separately in section 6.1: sparsity-aware part-sensitive warping, stacked triple attention, attentive hierarchical backbone, double attentive dynamic voxelization, and multi-view Pointpillars. The experimental results of statistical normalization, few-shot fine-tuning, and pseudo labeling in SAPL have presented in section 6.2. We also display the result of the ablation study for both Pointpillars+ and SAPL algorithm as the end of each section.

6.1 Detector Performance

As mentioned in section 5.1.1, we evaluate the performance of Pointpillars+ on KITTI's validation dataset. In this section, we show the experimental results of the five aforementioned modules: sparsity-aware part-sensitive warping, stacked triple attention module, attentive hierarchical backbone, double attentive dynamic voxelization, and multi-view Pointpillars.

6.1.1 Effect of Sparsity-Aware Part-Sensitive Warping

We investigate the effectiveness of the part-sensive warping module and the additional Top K' operation in our sparsity-aware part-sensitive warping module using Pointpillars as the baseline by selecting different K'. We choose the aforementioned parameter settings in chapter 5 to train the baseline of Pointpillars. The default spatial resolution of SAPS-Warp is 4×7. Therefore, SAPS-Warp is equivalent to the original PS warping module when k' $= 28$.

We report the performance of PS-Warp from two perspectives. Table 6.1 shows the 3D and BEV mAP results at different difficulty levels following the definition in KITTI's benchmark.

k'	3D mAP			BEV mAP		
	E	**M**	**H**	**E**	**M**	**H**
Baseline	89.02	76.00	72.25	93.95	88.50	**86.76**
10	89.21	76.01	71.63	92.25	87.47	84.19
20	89.78	78.48	73.89	93.61	88.64	85.74
24	**89.82**	**79.14**	74.38	94.11	89.20	86.38
28	89.74	78.16	**74.84**	**94.20**	**89.93**	85.11

Table 6.1: 3D object detection results of the sparsity-aware part-sensitve warping module. We report the 3D and BEV mean avearge precision results mAP_{3D}/mAP_{BEV} of Car and Van category on the KITTI' validation dataset using pointpillars as baseline under 0.7 IOU threshold with 40 recall positions. **E**, **M**, **H** denote easy, moderate, and hard seperately.

6.1.2 Effect of Stacked Triple Attention

Table 6.2 and table 6.3 shows the contribution of stacked triple attention module. We use the Pointpillars with PS-Warp as our baseline and consider the situation of Pointpillars with a single TA module and with a stacked TA module. The single TA module is the front part of the stacked TA module, which is followed by a concatenation operation. The Pointpillars with TA module has an inference time of 34ms, and the speed of Pointpillars with stacked TA module is 35ms.

Method	Speed (ms)	3D mAP			BEV mAP		
		E	**M**	**H**	**E**	**M**	**H**
Baseline*	**33**	89.74	78.16	74.84	94.20	89.93	85.11
TA	34	**90.81**	80.32	75.33	**96.77**	**89.97**	87.22
S-TA	35	90.50	**80.44**	**75.55**	96.55	89.92	**87.23**

Table 6.2: 3D object detection results of the stacked triple attention module. We report the 3D and BEV mean average precision results mAP_{3D}/mAP_{BEV} of Car and Van category on the KITTI validation dataset under 0.7 IOU threshold with 40 recall positions. **E**, **M**, **H** denote easy, moderate, and hard separately.* We use Pointpillars with PS-Warp as the baseline in this table.

Method	3D mAP			BEV mAP		
	0-30m	30-50m	50-70m	0-30m	30-50m	50-70m
Baseline*	86.77	44.14	**5.85**	93.65	67.52	**15.58**
TA	87.27	46.88	5.27	93.68	**67.77**	14.94
S-TA	**87.35**	**47.56**	5.14	**93.77**	67.27	14.19

Table 6.3: 3D object detection results of the stacked triple attention module. We report the 3D and BEV mean avearge precision results mAP_{3D}/mAP_{BEV} of Car and Van category on the KITTI' validation dataset under 0.7 IOU threshold with 40 recall positions. 0-30m, 30-50m, 50-70m denote the distance from objects to ego-cars.* We use Pointpillars with PS-Warp as baseline in this table.

6.1.3 Effect of Attentive Hierarchical Backbone

Table 6.4 and table 6.5 shows the performance of different backbones for Pointpillars. The baseline uses the default backbone in the original paper, as shown in figure 3.1. We demonstrate the effectiveness of our attentive hierarchical backbone and attentive addition fusion block by replacing the attentive fusion block with two simple operations: point-wise addition and channel-wise concatenation. We also change the activation function in the attentive addition block from Softmax to Sigmoid to show that the Softmax function is more suitable for our model.

Method	Speed (ms)	3D mAP			BEV mAP		
		E	**M**	**H**	**E**	**M**	**H**
Baseline	**32**	89.02	76.00	72.25	93.95	88.50	**86.76**
Addition	**32**	89.16	77.76	73.57	**96.08**	88.59	86.68
Concat	**32**	89.13	77.68	72.31	93.95	88.21	86.21
AttFuse (sigmoid)	34	88.95	77.67	72.38	94.00	88.15	86.36
AttFuse (softmax)	34	**89.56**	**78.26**	**74.23**	96.03	**88.63**	**86.76**

Table 6.4: 3D object detection results of the attentive hierarchical backbone. We report the 3D and BEV mean avearge precision results mAP_{3D}/mAP_{BEV} of Car and Van category on the KITTI' validation dataset using Pointpillars as baseline under 0.7 IOU threshold with 40 recall positions. **E**, **M**, **H** denote easy, moderate, and hard seperately.

Method	3D mAP			BEV mAP		
	0-30m	30-50m	50-70m	0-30m	30-50m	50-70m
Baseline	84.83	41.27	5.24	94.32	69.23	22.40
Addition	84.96	43.93	**6.41**	94.25	69.38	22.92
Concat	84.84	42.36	5.34	94.07	68.66	22.68
AttFuse (sigmoid)	84.70	43.83	4.97	94.12	68.83	21.77
AttFuse (softmax)	**85.35**	**44.25**	5.46	**94.46**	**69.44**	**23.78**

Table 6.5: 3D object detection results of the attentive hierarchical backbone. We report the 3D and BEV mean average precision results mAP_{3D}/mAP_{BEV} of Car and Van category on the KITTI validation dataset using Pointpillars as baseline under 0.7 IOU threshold with 40 recall positions.0-30m, 30-50m, 50-70m denote the distance from objects to ego-cars.

6.1.4 Effect of Double Attentive Dynamic Voxelization

We investigate the effect of dynamic voxelization and double attentive voxelization modules using Pointpillars as the baseline. We simply replace the original hard voxelization operation with these two new modules and keep other components and parameters unchanged. Table 6.6 and table **??** display the detection performance under different settings.

Method	Speed (ms)	3D mAP			BEV mAP		
		E	**M**	**H**	**E**	**M**	**H**
Baseline	**32**	89.02	76.00	72.25	93.95	88.50	86.76
DV	45	89.00	76.25	72.79	94.05	88.64	**86.88**
DA-DV	48	**89.08**	**76.70**	**72.88**	**94.25**	**89.23**	86.84

Table 6.6: 3D object detection results of the double attentive dynamic voxelization. We report the 3D and BEV mean average precision results mAP_{3D}/mAP_{BEV} of Car and Van category on the KITTI validation dataset using Pointpillars as baseline under 0.7 IOU threshold with 40 recall positions. DV denotes dynamic voxelization, and DA-DV denotes double attentive dynamic voxelization.

Table 6.6 compares the running speed of three models. As shown in the table, dynamic voxelization operation brings 7ms latency in average during inference time compared to hard voxelization operation, and double attentive operation adds 3ms latency.

6.1.5 Effect of Multi-View Pointpillars

We implement Multi-view Pointpillars and Attentive Multi-view Pointpillars following section 5.1.4 and show the experimental results in table 6.7. Different from the Pointpillars baseline in section 6.1.1, section 6.1.2, section 6.1.3 and 6.1.4, we choose the version of Pointpillars from repository OpenPCDet as the baseline, which is evaluated only on Car category. This version is more accurate but slower compared to the previous baseline due to some larger parameter settings. Here we compare our Multi-view Pointpillars and Attentive Multi-view Pointpillars with this baseline.

Method	Speed (ms)	3D AP			BEV AP			AOS		
		E	**M**	**H**	**E**	**M**	**H**	**E**	**M**	**H**
Point -pillars	**38**	87.37	78.29	75.40	91.47	87.90	86.98	95.61	**93.65**	91.15
Multi -view	45	89.61	79.10	**77.70**	93.76	88.31	**87.54**	**97.28**	93.34	**91.34**
AttMulti -view	46	**89.74**	**80.17**	77.30	**94.10**	**90.16**	87.53	97.19	93.56	90.93

Table 6.7: 3D object detection results of the multi-view Pointpillars. We report the 3D, BEV avearge precision results AP_{3D}/AP_{BEV}, speed and avearge orientation similarity AOS of Car category on the KITTI' validation dataset using Pointpillars as baseline under 0.7 IOU threshold with 40 recall positions. **E**, **M**, **H** denote easy, moderate, and hard seperately.

6.1.6 Qualitative Comparision

Table 6.8 shows ablation results for the effect of each component in our Pointpillars+. We summarize the speed and main 3D mAP results on the Car and Van category under different difficulty levels. After combing three proposed modules, our Pointpillars+

achieves a 3D mAP of 92.46%, 80.53%, 75.50% for easy, moderate, and hard objects with the running speed of 36 ms on our 2080 Super GPU.

Method			Speed (ms)	3D mAP		
S-TA	AH -Backbone	SAPS -Warping		Easy	Moderate	Hard
			32	89.02	76.00	72.25
	✓		34	89.56	78.26	74.23
		✓	33	89.82	79.14	74.38
✓		✓	35	90.50	80.44	**75.55**
	✓	✓	35	90.06	78.28	74.86
✓	✓	✓	36	**92.46**	**80.53**	75.50

Table 6.8: Ablation Study of Pointpillars+. Results are evaluated on KITTI's validation dataset for Car and Van category at 0.7 IOU threshold with 40 recall positions.

To show the detection performance of our Pointpillars+ in a more direct way, we select three frames in KITTI's validation dataset and visualize the prediction using Pointpillars+ and the corresponding ground truth bounding boxes from both bird eye view and image view. Figure 6.1 and figure 6.2, figure 6.3 and figure 6.4, figure 6.5 and figure 6.6 depict three different scenes from two perspectives respectively.

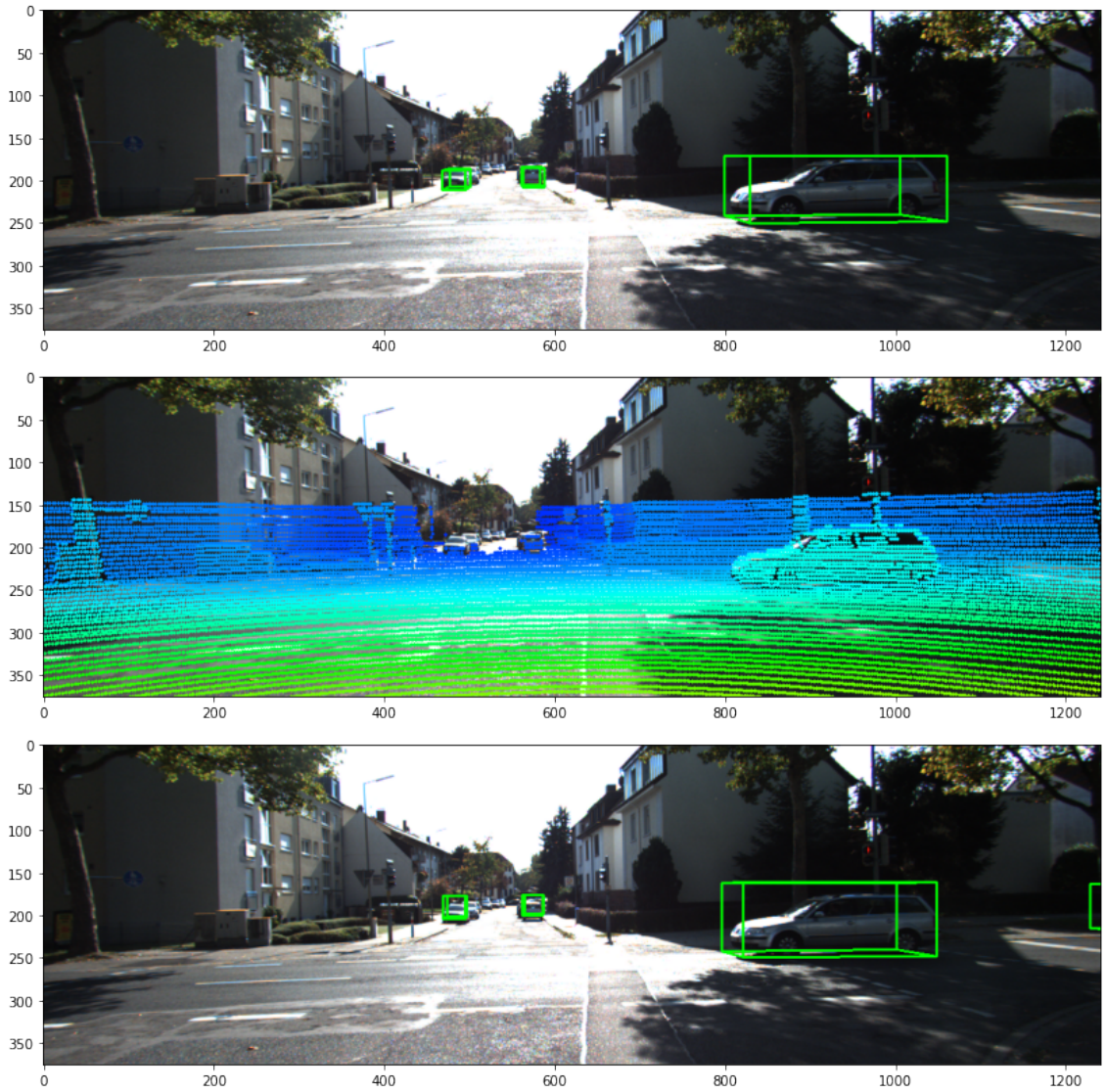


Figure 6.1: 2D detection result using Pointpillars+ on a scene in the KITTI dataset. The first figure (top) shows the ground truth bounding boxes that are projected into the image. In the second figure (middle), we projects scanned LiDAR points into the 2D image. The third figure (down) presents the prediction results using our Pointpillars+, which are projected from detected 3D bounding boxes to 2D boxes. We keep the detected boxes with confidence score over 0.3.

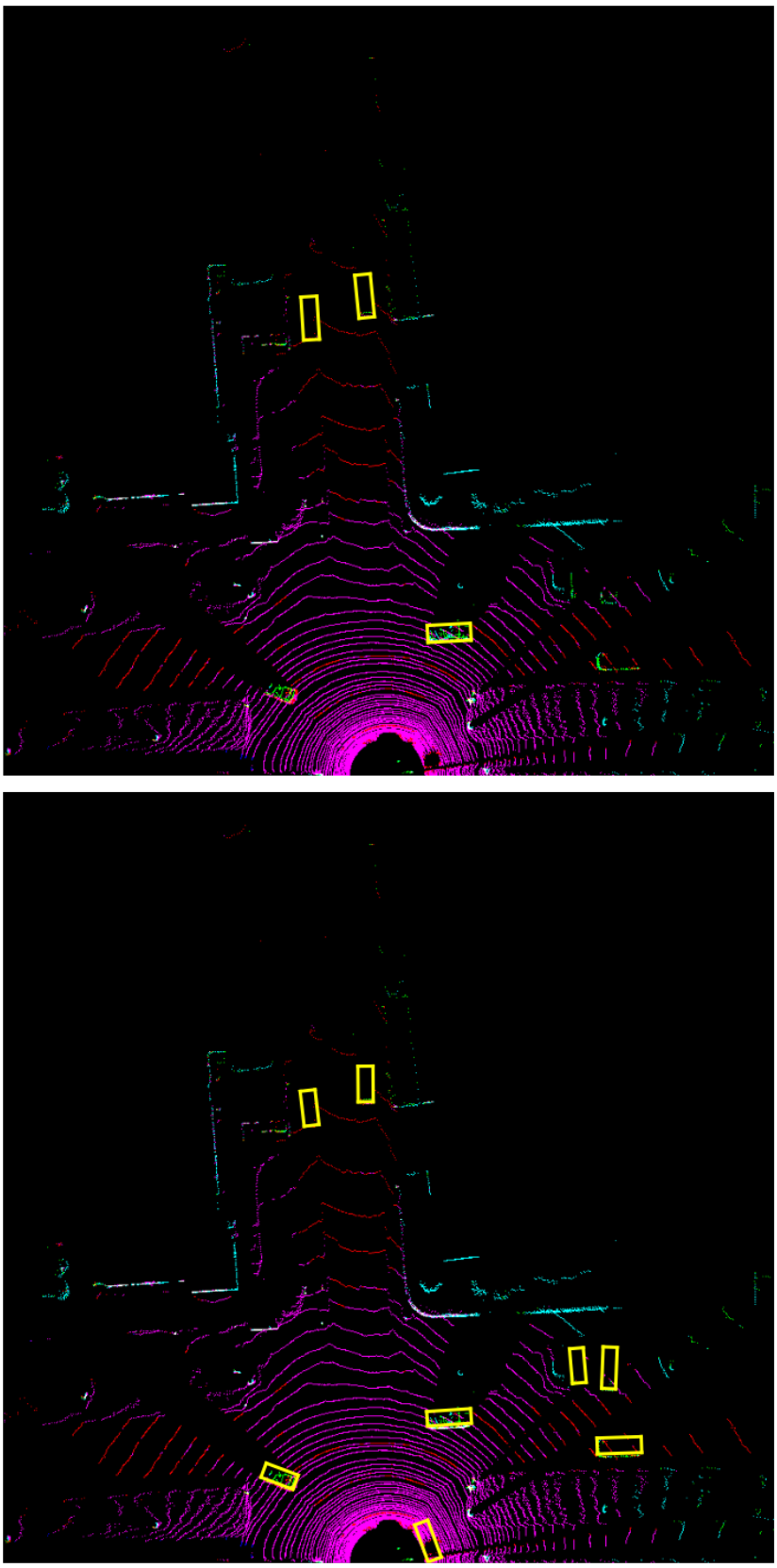


Figure 6.2: BEV detection result using Pointpillars+ on a scene in the KITTI dataset. This scene is the same as figure 6.1. The first figure (top) depcicts the ground truth bounding boxes that are projected into the BEV. The second figure (down) shows the prediction results using our Pointpillars+ on the BEV.

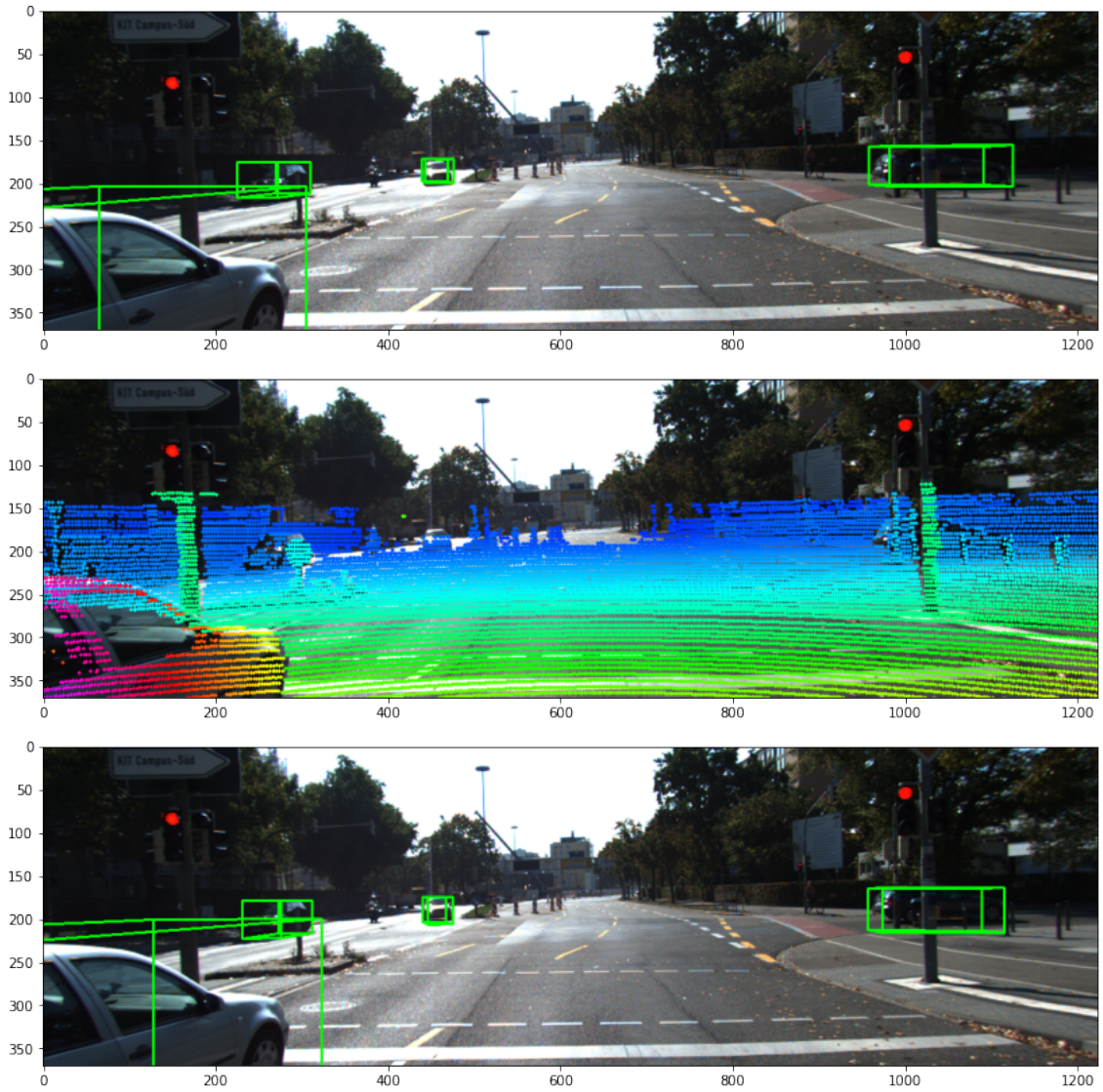


Figure 6.3: 2D detection result using Pointpillars+ on a scene in the KITTI dataset. The first figure (top) shows the ground truth bounding boxes that are projected into the image. In the second figure (middle), we projects scanned LiDAR points into the 2D image. The third figure (down) presents the prediction results using our Pointpillars+, which are projected from detected 3D bounding boxes to 2D boxes. We keep the detected boxes with confidence score over 0.3.

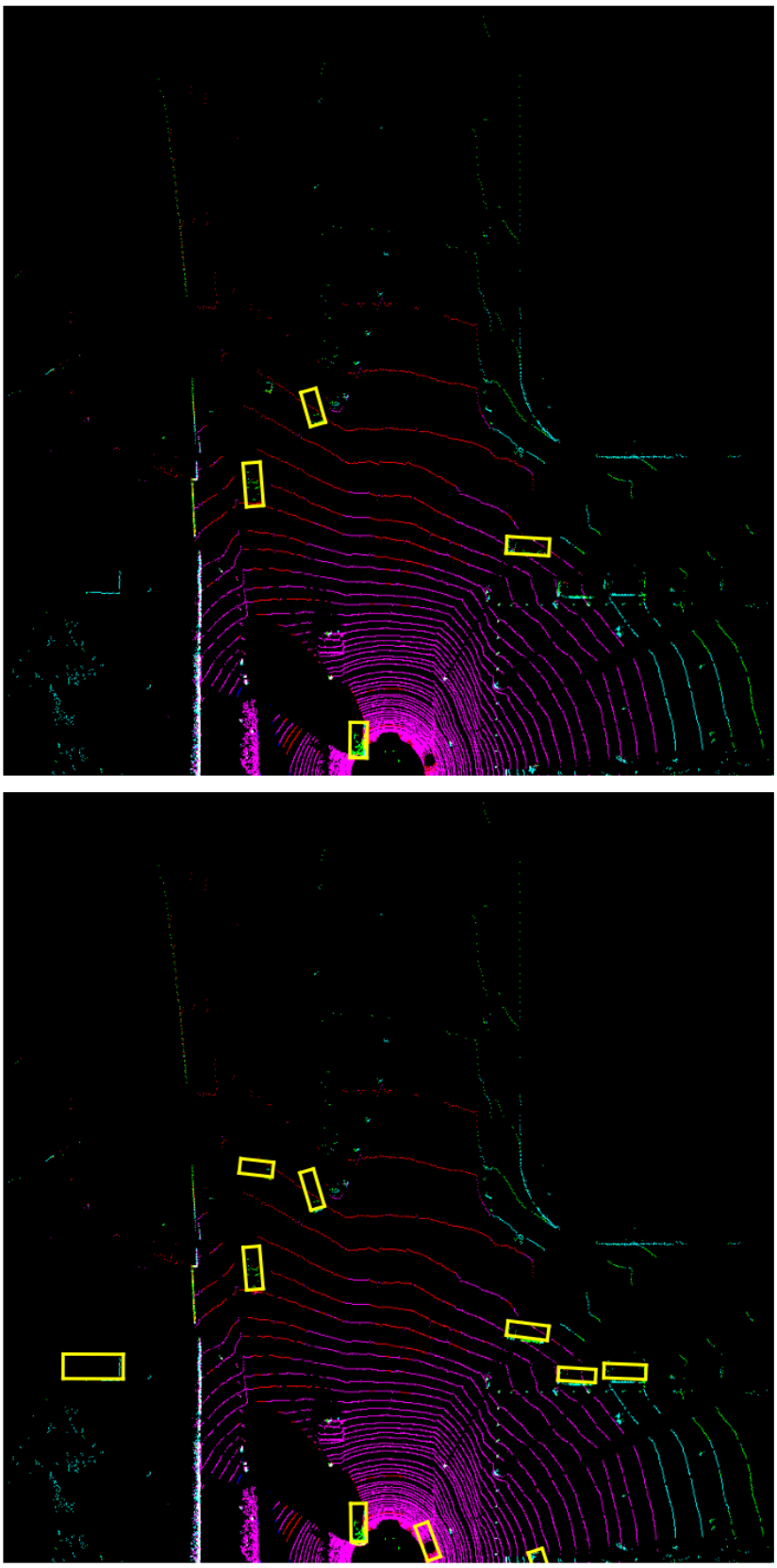


Figure 6.4: BEV detection result using Pointpillars+ on a scene in the KITTI dataset. This scene is the same as figure 6.3. The first figure (top) depcicts the ground truth bounding boxes that are projected into the BEV. The second figure (down) shows the prediction results using our Pointpillars+ on the BEV.

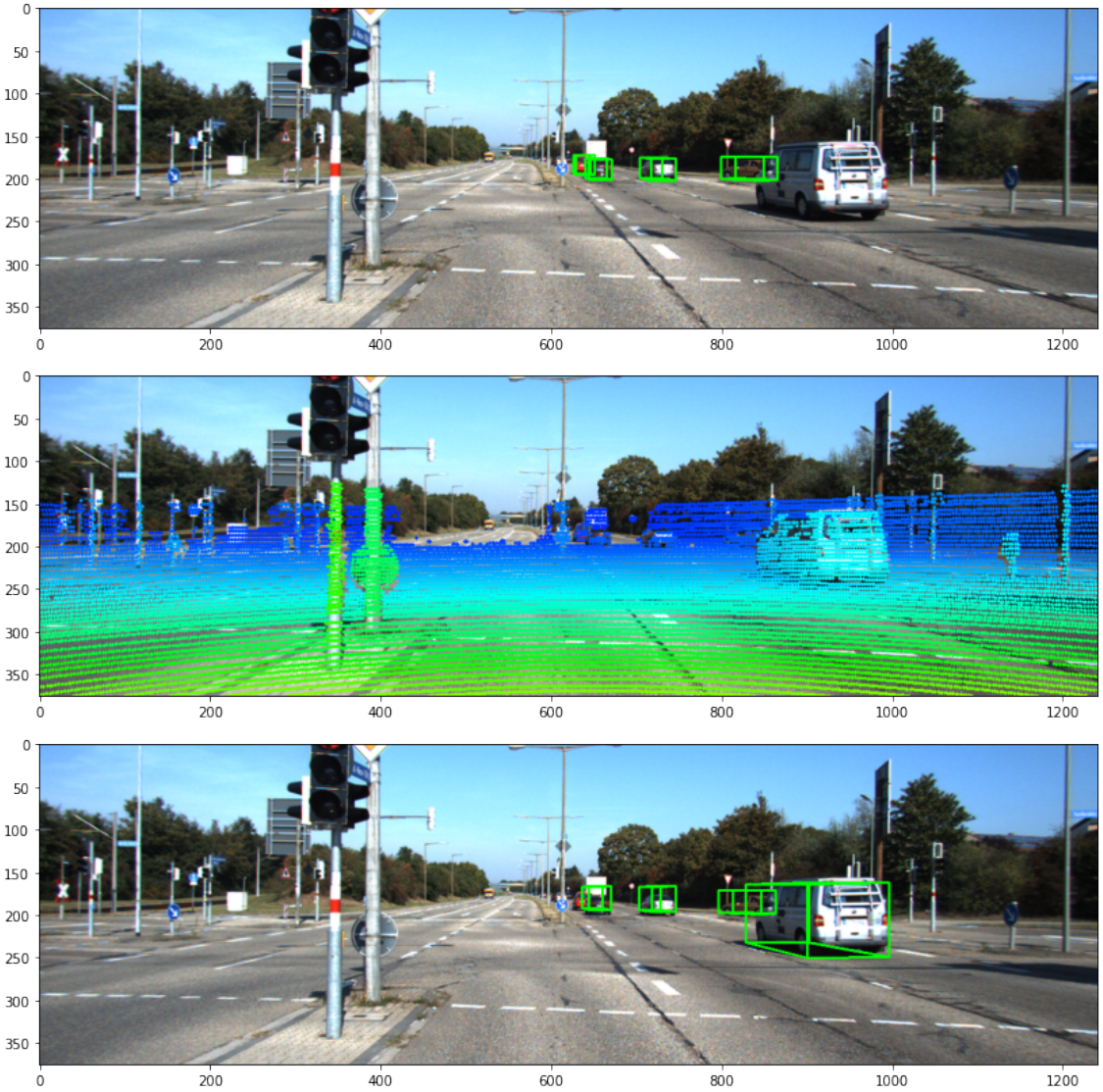


Figure 6.5: 2D detection result using Pointpillars+ on a scene in the KITTI dataset. The first figure (top) shows the ground truth bounding boxes that are projected into the image. In the second figure (middle), we projects scanned LiDAR points into the 2D image. The third figure (down) presents the prediction results using our Pointpillars+, which are projected from detected 3D bounding boxes to 2D boxes. We keep the detected boxes with confidence score over 0.3.

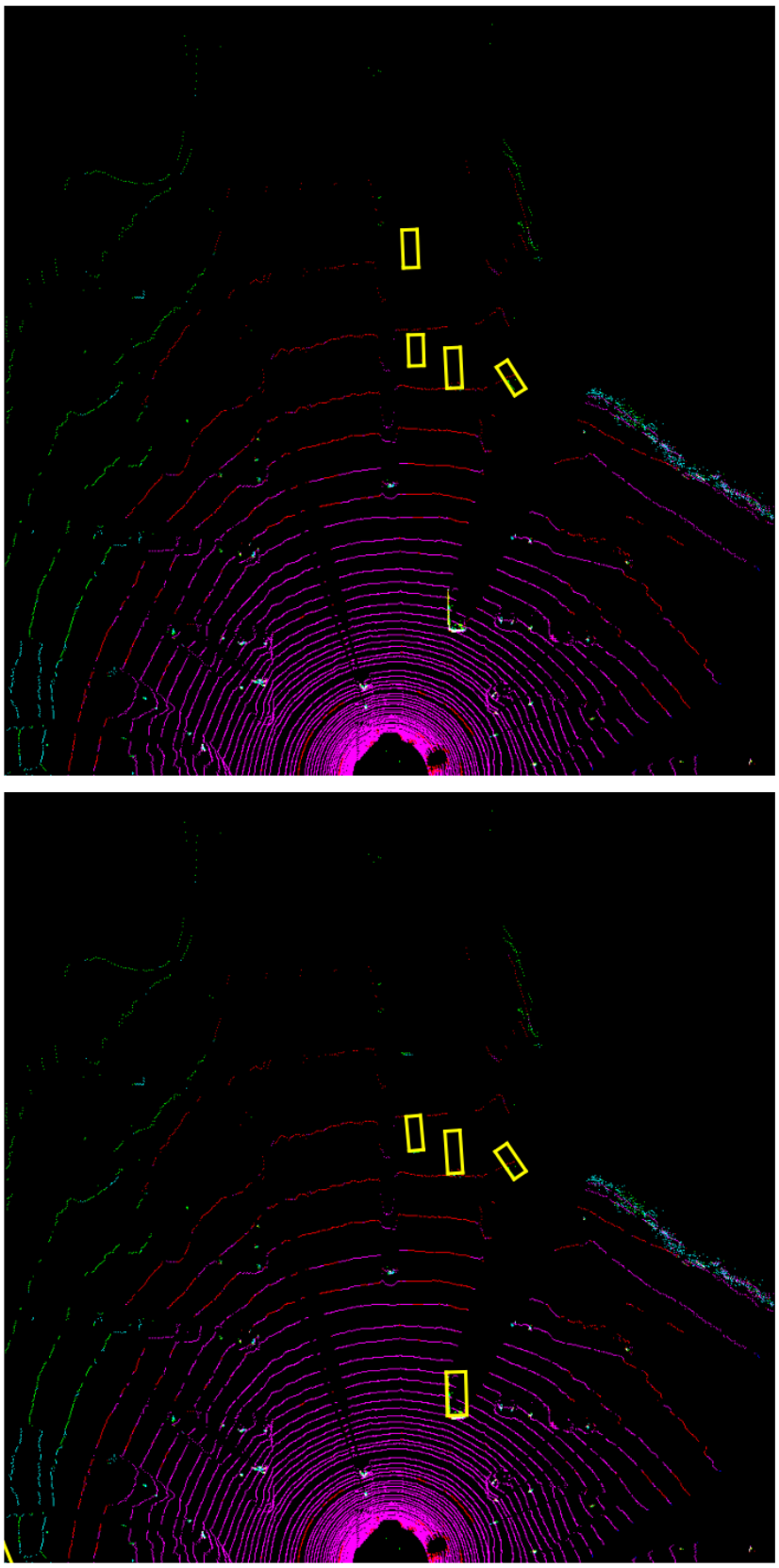


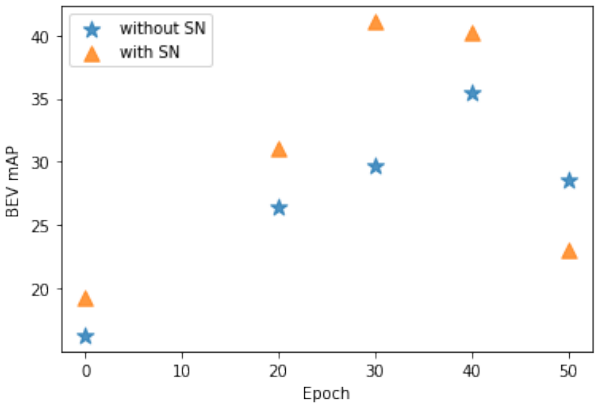
Figure 6.6: BEV detection result using Pointpillars+ on a scene in the KITTI dataset. This scene is the same as figure 6.5. The first figure (top) depcicts the ground truth bounding boxes that are projected into the BEV. The second figure (down) shows the prediction results using our Pointpillars+ on the BEV.

6.2 SAPL Performance on Providentia

We prove the effectiveness of our SAPL algorithm on the Providentia dataset in this section from two aspects: the effect of statistical normalization, the effect of pseudo labeling. As mentioned in section 5.2.1, we evaluate our results on 200 frames Providentia testing dataset.

6.2.1 Effect of Statistical Normalization

We implement statistical normalization following section 5.2.3 and produce a new rescaled KITTI dataset. We train two Pointpilars+ networks on both the original KITTI dataset and rescaled KITTI's dataset after statistical normalization. We report the results of detectors that are purely trained on KITTI's dataset at epoch = 0. We also investigate the performance of few-shot fine-tuning using 100 training samples for two already trained detectors. Table 6.9 and the corresponding figure show the BEV performance of car and van on our Providentia testing dataset during the training process.



Epoch	BEV mAP (%)	
	without SN	with SN
0	16.25	19.29
20	26.46	31.03
30	29.65	**41.07**
40	**35.45**	40.18
50	28.49	23.02

Table 6.9: BEV detection results mAP_{BEV} of fine-tuned model trained on Providentia training dataset for car and van categories with IOU threshold 0.25 with 40 recall positions.

6.2.2 Effect of Pseudo Labeling

We explore the effectiveness of pseudo labeling in our SAPL algorithm from two parts. In table 6.10 we re-train the model, which is already trained on rescaled KITTI's

dataset, on Providentia training sets and pseudo labeling sets. We use the model that is pre-trained on rescaled KITTI's dataset and fine-tuned on 100 Providentia training datasets to produce pseudo labels. We set the confidence threshold to 0.2 to filter low-quality proposals in the generated pseudo labels.

Num Pseudo-Labeled Samples	Num Labeled Samples	Percent Labeled Samples (%)	BEV mAP (%)
0	100	100	41.07
50	100	67	44.77
100	100	50	**45.05**
150	100	40	37.26

Table 6.10: BEV detection results mAP_{BEV} of fine-tuned model trained on Providentia training dataset and generated pesudo labeled dataset with confidence threshold 0.2 for filtering low quality proposals. The BEV results are evaluated on car and van categories with IOU threshold 0.25 with 40 recall positions.

Table 6.11 shows the effect of pseudo-labeling confidence threshold on the final detection performance. We re-trained the model on a mixed dataset with 100 frames Providenia training samples and 100 frames pseudo labels under different confidence thresholds. The BEV mAP in the table for each scenario is the best result during the whole training process.

Conf. Threshold	BEV mAP (%)
0.08	43.45
0.1	**47.56**
0.15	46.09
0.2	45.05

Table 6.11: BEV detection results mAP_{BEV} of fine-tuned model trained on 100 frames Providentia training dataset and 100 frames generated pesudo labeled dataset under different confidence thresholds. The BEV results are evaluated on car and van categories with IOU threshold 0.25 with 40 recall positions.

Figure 6.7 depicts the performance of pseudo labels with a confidence threshold of 0.2. The model for generating pseudo labels is trained on rescaled KITTI's dataset and 100 frames Providentia training sets.

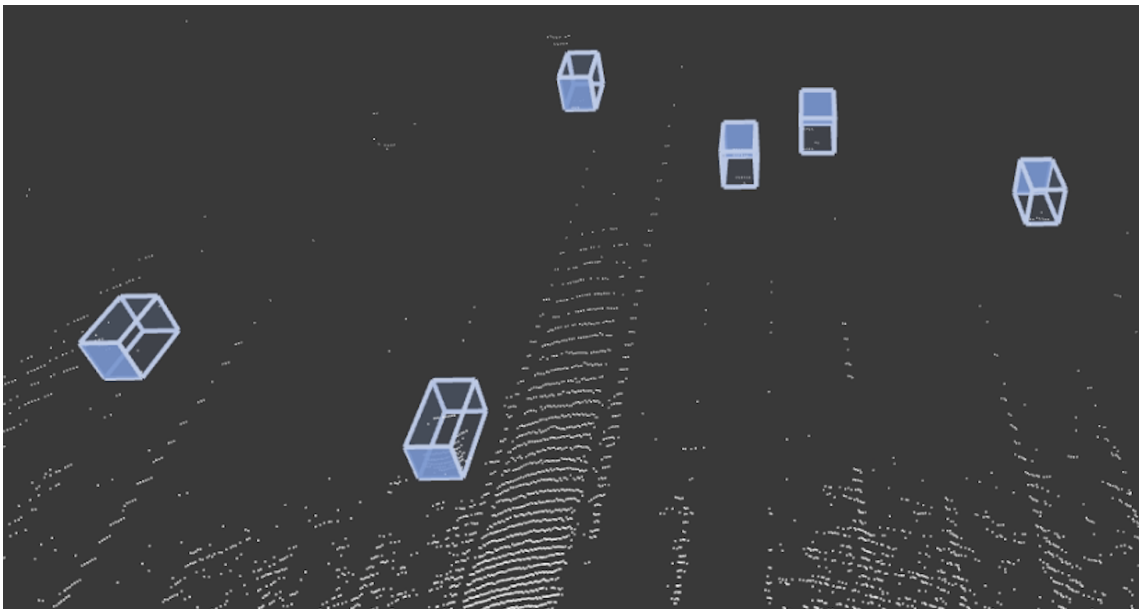


Figure 6.7: Pseudo labeled frames with confidence threshold 0.2

6.2.3 Quantitive Comparision

Table 6.12 summarizes the effect of each component in SAPL algorithm. We display the BEV mAP results on the Car and Van category at IOU threshold 0.25 with 40 recall positions. We use the model that is trained on the original KITTI dataset as the baseline and shows the improvement of three main modules in our SAPL algorithm. The fine-tuning module introduces 100 labeled frames in the target domain, and the pseudo labeling module introduces 100 unlabeled frames in the target domain.

SAPL			BEV mAP (%)
Statistical Normalization	Fine Tuning	Pseudo Labeling	
			16.25
✓			19.29
	✓		35.45
✓	✓		41.07
✓	✓	✓	**47.56**

Table 6.12: Ablation Study of SAPL algorithm. BEV results mAP_{BEV} are evaluated on Providentia testing dataset for Car and Van category at 0.25 IOU threshold with 40 recall positions.

Chapter 7

Discussion

In this chapter, we analyze and discuss the experimental results of Pointpillars+ and SAPL algorithm from chapter 6. We discuss the effect of sparsity-aware part-sensitive warping, the effect of stacked triple attention, the effect of attentive hierarchical backbone in Pointpillars+ and also discuss the performance of double attentive dynamic voxelization and multi-view Pointpillars in section 7.1. In section 7.2, we analyze the effect of statistical normalization, the effect of few-shot fine-tuning, and the effect of pseudo labeling in the SAPL algorithm.

7.1 Detector Analysis

7.1.1 Sparsity-Aware Part-Sensitive Warping

The original PS-Warp is the same as our SAPS-Warp module with k' = 28. Table 6.1 shows that the orignal PS-Warp block boosts the 3D mAP of Pointpillars by (0.72%, 2.16%, 2.59%) on easy, moderate, hard difficulties and boosts the BEV mAP by (0.25%, 1.43%) on easy, moderate difficulties respectively. In comparison with the original PS-Warp, SAPS-Warp increases the 3D mAP by 0.08% and 0.98% on easy and moderate difficulty with k' = 24.

The 3D mAP and BEV mAP results of SAPS-Warp in table 6.1 with k' = 10 are both worse than the original PS-Warp module. One possible reason is that the number of sampled points is too low to represent the whole structure of the bounding box, which leads to large information loss. SAPS-Warp with k' = 24 performs the best among all parameters, which achieves 3D mAP at 79.14% on moderate difficulty. In this case, the most representative parts in the boxes are kept without losing too much structural information. SAPS-Warp (k' = 24) improves the 3D mAP and BEV mAP at all difficulty levels compared to Pointpillars baseline except the BEV mAP on hard samples. The 3D mAP of hard samples increases from 72.25% to 74.38%, while its BEV mAP slightly drops from 86.76% to 86.38%. It can be assumed that the serve occlusion and truncation of hard samples may lead to some re-scoring bias on the bird eye view when we reweight the confidence of the bounding box depending on its part structure. Furthermore, SAPS-Warp (k' = 24) alleviates this structural bias

by discarding the scores of some low-confidence parts and increases BEV mAP from 85.11% to 86.38% for hard samples compared to the original PS-Warp.

7.1.2 Stacked Triple Attention

The results in table 6.2 show that the stacked triple attention module can greatly improve the 3D mAP and BEV mAP of the model by 0.76%, 2.28%, 0.71% on easy, moderate, and hard 3D mAP, and 2.35%, 2.12% on easy and hard BEV mAP respectively in comparison with the baseline. It introduces only 2*ms* additional inference speed latency, which is effective and worthy.

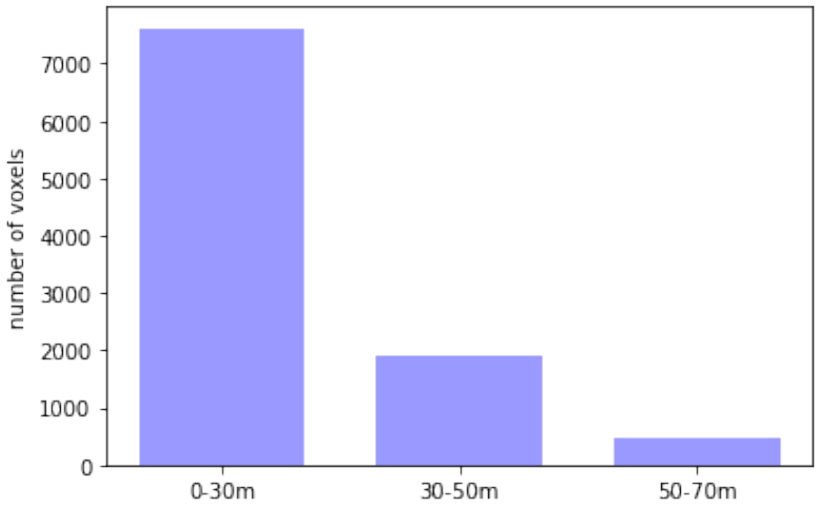


Figure 7.1: Distribution of voxels in different distance range

Table 6.3 shows that the stacked TA module considerably increases the 3D mAP for near objects from 0 to 50 meters by 0.58% and 3.42% on 0-30m and 30-50m respectively compared to the baseline. However, table 6.3 also presents that both the TA module and stacked TA module are not helpful for improving the performance of hard objects over 50 meters. The 3D mAP and BEV mAP drop from 5.85% and 15.58% to 5.14% and 14.19% separately for distant objects between 50 and 70 meters. One possible reason is that the TA module tends to assign a higher weight to the voxels within 50 meters in the voxel-wise attention branch. Figure 7.1 shows the distribution of voxels in three different distance range. It can be seen that the voxels in near areas (within 30 meters) make up the majority of the whole sampled voxels, which dominate the gradient descent during the training process.

The stacked TA module improves the 3D mAP by 0.16% and 0.22% on moderate and hard difficulty, and 0.68% on 30-50m range, respectively, compared to the single residual TA module. However, in comparison to the single TA module, stacking two TA modules does not get an obvious increment on BEV mAP. The effect of two TA modules in the stacked TA module overlaps for improving the bird eye view accuracy.

Overall, the stacked TA module only introduces 1ms extra speed latency compared to the single TA module and brings 3D mAP improvement, which is acceptable.

7.1.3 Attentive Hierarchical Backbone

Table 6.4 presents that our attentive hierarchical (AH) backbone brings promising improvement on both 3D mAP and BEV mAP. The 3D mAP and BEV mAP of the model with AH backbone increases by (0.54%, 2.26%, 1.98%), and (2.08%, 0.13%, 0%) on easy, moderate, and hard difficulty compared to the Pointpillars baseline. The AH backbone introduces no extra inference time when we replace the attentive fusion block with a simple addition or concatenation operation. Surprisingly, it improves the performance at almost all the difficulty levels on both 3D mAP and BEV mAP even without attentive fusion block compared to the Pointpillars baseline. Our AH backbone with single addition operation increases the 3D mAP by (0.14%, 1.76%, 1.32%) on easy, moderate, hard objects and increases the BEV mAP by (2.13%, 0.09%) on easy, moderate objects compared to the default backbone in Pointpillars. As shown in table 6.5, our AH backbone improves the 3D mAP and BEV mAP at all three distance ranges by (0.52%, 2.98%, 0.22%) and (0.14%, 0.21%, 1.38%) on 0-30m, 30-50m and 50-70m separately. Considering the case without attentive fusion block, the AH backbone with addition operation still improves 3D mAP and BEV mAP of the model by (0.13%, 2.66%, 1.17%), and (-0.17%, 0.15%, 0.52%).

As shown in table 6.4, our proposed attentive fusion block brings 2ms speed latency. Compared to the AH backbone with single addition operation, the attentive fusion module improves the 3D mAP by (0.40%, 0.50%, 0.66%) on easy, moderate, and hard difficulties and improves the BEV mAP by (0.04%, 0.08%) on moderate and hard difficulties. Table 6.5 presents that attentive fusion block boosts the 3D mAP by (0.39%, 0.32%) on 0-30m and 30-50m, and boosts the BEV mAP by (0.21%, 0.06%, 0.86%) on 0-30m, 30-50m and 50-70m respectively compared to the single addition operation in AH backbone. The attentive fusion module in the AH backbone brings significant improvement on 3D mAP and a slight improvement on BEV mAP. It probably benefits from the rich spatial and semantic features that are adaptively aggregated and fused.

We compare the performance of two optional operations in the AH backbone: addition and concatenation. Channel-wise concatenation increases output channel dimension from 128 to $128 * 3$. It directly keeps the raw output features and allows the network itself to learn how to fuse features during the training process. Element-wise addition operation fixes the fusion pattern and enriches the information for each element while keeps the output channel dimension unchanged. The results in table 6.4 and table 6.5 empirically proves that addition operation is more suitable for our AH backbone. We also compare two activation functions in our attentive fusion module: sigmoid and softmax function. The sigmoid function allows the network to emphasize multiple features at the same time. The features in the attentive fusion block are

independently activated and then fused. The softmax function normalizes the whole input and assigns weight jointly. It tights the connection between the input features and only emphasizes few elements that are globally important. Table 6.4 and table 6.5 presents that attentive fusion block with softmax as activation function performs far better than sigmoid. Hence, we finally choose the softmax-based attentive fusion block as the fusion method in our AH backbone.

7.1.4 Double Attentive Dynamic Voxelization

We compare the results of Pointpillars with dynamic voxelization and double attentive dynamic voxelization in table 6.6. As shown in the table, the 3D mAP and BEV mAP of Pointpillars with double attentive dynamic voxelization is improved by (0.06%, 0.70%, 0.63%), (0.30%, 0.73%, 0.08%) compared to the baseline on easy, moderate and hard difficulties. Moreover, the proposed double attentive dynamic voxelization boosts the 3D mAP by (0.08%, 0.45%, 0.09%) on easy, moderate, hard samples, and boosts the BEV mAP by (0.20%, 0.59%) on easy, moderate difficulties in comparison with the dynamic voxelization.

The effectiveness of dynamic voxelization is reasonable because it uses all the points within voxels as input and keeps all the sampled voxels for training, which enriches the input features compared to hard voxelization in Pointpillars. However, both dynamic voxelization and double attentive dynamic voxelization introduces too much latency, which is not acceptable for the real-time requirement (25 FPS) of our model. The inference speed of Pointpillars with dynamic voxelization is 45ms, and the additional double attentive module brings 3ms more. Hence, we do not introduce this block to our final Pointpillars+.

7.1.5 Multi-View Pointpillars

As shown in the table 6.7, both Multi-view Pointpillars and Attentive Multi-view Pointpillars improves the performance of baseline by a large margin. The 3D AP and BEV AP of Multi-view Pointpillars increases by (2.24%, 0.81%, 2.3%), (2.29%, 0.41%, 0.56%) compared to Pointpillars baseline for Car category on easy, moderate and hard difficulties separately. The average orientation similarity also gets greatly improved, especially on easy samples from 95.61% to 97.28%. After introducing an attentive mechanism in Multi-view Pointpillars, the 3D AP and BEV AP of the model are further boosted by (0.13%, 1.07%), (0.34%, 1.85%) on easy and moderate difficulties separately.

It can be seen from table 6.7 that both 3D AP and BEV AP drop on hard samples after introducing an attentive mechanism. One possible reason is that the attentive block in the attentive multi-view feature encoding net tends to assign a higher weight to the points that are useful for detecting easy and moderate objects, which are also the majority of all foreground points. The useful points for hard samples only make

up a small proportion, which is easy to be overwhelmed by other points.

Overall, the introduction of multi-view information to Pointpillars greatly improve its 3D and BEV performance. However, multiple gather and scatter operations in multi-view Pointpillars significantly slow down the running speed of the model. Table 6.7 presents that the Multi-view Pointpillars bring 7ms latency, and the additional attentive mechanism also adds extra 1ms. Due to the real-time requirement, we finally do not choose Multi-view Pointpillars as our final model.

7.1.6 Summary

Table 6.8 presents the ablation study for three main components in our Pointpillars+. Our Pointpillars+ achieves 3D mAP improvement by (3.44%, 4.53%, 3.25%) on easy, moderate and hard difficulties respectively compared to the Pointpillars baseline. Our Pointpillars+ adds only 4ms latency which stastifies our real-time speed requirement. Figure 6.2, figure 6.4 and figure 6.6 show that our Pointpillars+ has strong generalization ability, and is able to detect objects that are not labeled in the ground truth.

7.2 SAPL Analysis

7.2.1 Statistical Normalization

Table 6.9 compares the BEV detection results of two models with different pre-trained weights before and after few-shot fine-tuning. Epoch 0 shows the scenario that when we directly evaluate the model that is merely trained on the KITTI dataset, on our Providentia testing sets, the model with pre-trained weight on statistical normalized KITTI outperforms the model with pre-trained weight on the original KITTI dataset by 3.04% BEV mAP. In addition, the fine-tuned model that is pre-trained on rescaled KITTI's dataset still has higher BEV mAP than the model with pre-trained weight on original KITTI. The BEV mAP gap between the two models increases from 3.04% before few-shot fine-tuning to 5.62% after few-shot fine-tuning. One possible reason is that when we employ statistical normalization, we take all 300 labeling frames into consideration, which introduces some prior knowledge of the vehicle's size from Providentia testing sets to the final model. Moreover, it also shows that the vehicle's size distribution in the source domain deeply affects the final detection performance of the target domain even with few-shot fine-tuning.

The corresponding figure of table 6.9 also shows that the model with statistical normalized pre-trained weight reaches the optimal parameters earlier than the model with the original pre-trained weight. The model with SN performs best at around 30 epochs, while the model without SN reaches the best performance at about 40 epochs. It is easy to explain that the rescaled KTTI's dataset has a shorter distribution distance to the target domain than the original KITTI's dataset.

7.2.2 Pseudo Labeling

We analyze the effect of labeled samples' percentage in the mixed training dataset. We choose the model pre-trained on rescaled KITTI dataset and fine-tuned with 100 Providentia training samples as our baseline in table 6.10. The model with 50 and 100 additional pseudo labeled samples outperforms our baseline, while the performance of the model with 150 additional pseudo-labeled samples shows worse results than baseline. It is clear that the produced pseudo labels are not completely accurate, and technically its has a similar effect as data argumentation methods with some labeling noise. Since we use the same Pointpillars+ as both student and teacher models, the improvement of pseudo labeling on the student model should have limited margin benefits. In our case, when the percentage of pseudo-labeled data reaches 60%, the labeling technique loses efficacy, and the performance of the model deteriorates. A number of factors may be accountable for this situation. One possible reason is that the number of noisy frames is too large, which overwhelms the original, accurate labeled samples and dominates the training process. Another possible reason is that our produced pseudo labels are not accurate enough, and there may exist many false-negative objects when we set the confidence threshold too high, where too many noisy objects are then introduced. To sum up, the performance of the pseudo labeling technique reaches the best when the percentage of labeled samples takes half of the whole dataset under the 0.2 confidence threshold in our case.

Table 6.11 shows the effect of the confidence threshold on the performance of pseudo labeling. We re-train the model with pre-trained weight from rescaled KITTI on 100 frames Providentia training sets and 100 pseudo-labeled frames. As shown in the table, the model performs best when we set the confidence threshold to 0.1, and the BEV mAP drops slightly when the confidence threshold is set too high or too set. This is easy to be explained that many false positive objects appear when we set the confidence threshold to a low value, and there exist many false-negative objects if the confidence threshold is set to a high value. In our case, the model reaches a good trade-off between false positive and false negative pseudo labels at the confidence threshold of 0.1.

7.2.3 Summary

Table 6.12 presents the contribution of three main components in our SAPL algorithm: Statistical normalization, few-shot fine-tuning, and pseudo labeling. The results show that all three modules improve the performance of the model in the target domain and few-shot fine-tuning contributes the most among the three modules. After applying the SAPL algorithm, the BEV mAP of the model increases by 31.31% with the help of 100 labeled, 100 unlabeled frames, and the size distribution of the target domain.

Chapter 8

Conlusion

Autonomous driving is deeply involved with our future, and the Providentia project aims at providing intelligent infrastructure which accelerates the development of autonomous driving. As a fundamental part of the Providentia project, our thesis focus on two main problems that commonly exist in the perception field of autonomous driving: design of real-time 3D object detector and label shortage problem. Firstly, we propose a real-time LiDAR-based Pointpillars+ model to detect the 3D positions of vehicles on the highway. Secondly, we present a statistical-aware pseudo labeling algorithm to solve the label-shortage problem in 3D object detection. We experimentally prove that our Pointpillars+ network outperforms the original Pointpillars by a large margin and the SAPL algorithm effectively improves the model's performance in the tagert domain.

Three main modules in Pointpillars+: sparsity-aware part-sensitive warping, stacked triple attention, and attentive hierarchical backbone, all contribute to the improvement of the model with few additional inference time latency. Besides, both double attentive dynamic voxelization and multi-view Pointpillars boost the 3D detection accuracy of the model. However, they are more suitable for the scenario where the model's inference speed is not of great importance. Three main components in the SAPL algorithm: statistical normalization, few-shot fine-tuning, and pseudo labeling also all make contributions to domain adaption. It can be used in the scenario where few labeled samples and size statistics of the target domain are available. Both Pointpillars+ and SAPL algorithms are practical and can be easily applied to other similar problems.

Chapter 9

Future Work

As a fundamental part of the Providentia project, there are still many insufficiencies of our work that can be improved and investigated in the future. We list the possible improvements that are discovered during the implementation and evaluation process as follows:

- As mentioned in section 5.2.1, LiDAR sensor has a limited detection range and can only detect few closely gathered foreground points for distant objects over 50 meters due to its installation position. It is also hard for the LiDAR only network to classify the categories of distant vehicles and estimate their sizes only based on few points. In this case, it is necessary to introduce additional image information for better classification and size estimation performance. We install several RGB camera sensors for data fusion in our Providentia system, which is necessary based on our analysis.

- The proposed sparsity-aware part-sensitive warping module in Pointpillars+ uses Top K' to adaptively select the most representative parts in the predicted bounding boxes. However, we need to adjust the value of K' to find out the most suitable parameter, which is dependent on different models and is a little bit time-consuming. We can design an attentive mechanism in the future to selectively control the importance of each part instead of deprecating all other less useful classification scores.

- As mentioned in the previous chapters, the proposed double attentive dynamic voxelization, and multi-view Pointpillars can also improve the detection performance to some extent. However, their current implementations consume too much time, which can not satisfy real-time requirements. We can re-implement these two algorithms for faster running speed in the future.

- In the SAPL algorithm, we choose Pointpillars+ as both our teacher and student model for the sake of simplicity. However, since we do not have inference time requirements for the teacher model, we can choose some slower but more accurate 3D object detectors such as PV-RCNN as our teachers, which can theoretically produce pseudo labels with better quality and then improve the performance of pseudo labeling in our SAPL algorithm. We may even choose multiple

teacher models to generate pseudo labels separately and then fuse their proposals together for robust results.

Bibliography

[1] Barrera, A., Guindel, C., Beltrán, J., and García, F. *BirdNet+: End-to-End 3D Object Detection in LiDAR Bird's Eye View*. Tech. rep. arXiv: 2003.04188v1.

[2] Behrendt, K. *Boxy Vehicle Detection in Large Images*. Tech. rep. URL: https://boxy-dataset.com/.

[3] Beltrán, J., Guindel, C., Moreno, F. M., Cruzado, D., García, F., and De La Escalera, A. *BirdNet: a 3D Object Detection Framework from LiDAR information*. Tech. rep. arXiv: 1805.01195v1.

[4] Braun, M., Krebs, S., Flohr, F., and Gavrila, D. M. *The EuroCity Persons Dataset: A Novel Benchmark for Object Detection*. Tech. rep. arXiv: 1805.07193v2.

[5] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O., and Company, A. *nuScenes: A multimodal dataset for autonomous driving*. Tech. rep. arXiv: 1903.11027v5.

[6] Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., and Hays, J. *Argoverse: 3D Tracking and Forecasting with Rich Maps*. Tech. rep.

[7] Chen, Q., Sun, L., Wang, Z., Jia, K., and Yuille, A. *Object as Hotspots: An Anchor-Free 3D Object Detection Approach via Firing of Hotspots*. Tech. rep.

[8] Chen, Y., Liu, S., Shen, X., and Jia, J. *Fast Point R-CNN*. Tech. rep. arXiv: 1908.02990v2.

[9] Dalal, N. and Triggs, B. *Histograms of Oriented Gradients for Human Detection*. Tech. rep. 2005. URL: http://lear.inrialpes.fr.

[10] Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., and Li, H. *Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection*. Tech. rep. 2021. arXiv: 2012.15712v2. URL: https://github.com/djiajunustc/Voxel-R-CNN..

[11] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. *Vision meets Robotics: The KITTI Dataset*. Tech. rep. URL: http://www.cvlibs.net/datasets/kitti..

[12] Girshick, R., Donahue, J., Darrell, T., and Malik, J. *Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5)*. Tech. rep. arXiv: 1311.2524v5. URL: http://www.cs.berkeley.edu/%CB%9Crbg/rcnn..

[13] He, C., Zeng, H., Huang, J., Hua, X.-S., and Zhang, L. *Structure Aware Single-stage 3D Object Detection from Point Cloud*. Tech. rep.

[14] Houston, J., Zuidhof, G., Bergamini, L., Ye, Y., Chen, L., Jain, A., Omari, S., Iglovikov, V., Ondruska, P., and Level, L. *One Thousand and One Hours: Self-driving Motion Prediction Dataset*. Tech. rep. arXiv: 2006.14480v2.

[15] Huang, X., Wang, P., Cheng, X., Zhou, D., Geng, Q., and Yang, R. *The ApolloScape Open Dataset for Autonomous Driving and its Application*. Tech. rep. arXiv: 1803.06184v4.

[16] Jeong, J., Lee, S., Kim, J., and Kwak, N. *Consistency-based Semi-supervised Learning for Object Detection*. Tech. rep. URL: https://github.com/soo89/CSD-SSD.

[17] Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. *PointPillars: Fast Encoders for Object Detection from Point Clouds*. Tech. rep. arXiv: 1812.05784v2. URL: https://github.com/nutonomy/second.pytorch.

[18] Lee, D.-H. *Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks*. Tech. rep.

[19] Li, C. R. Q., Hao, Y., Leonidas, S., and Guibas, J. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. Tech. rep. arXiv: 1706.02413v1.

[20] Li, Z., Yao, Y., Quan, Z., Yang, W., and Xie, J. *SIENet: Spatial Information Enhancement Network for 3D Object Detection from Point Cloud*. Tech. rep. arXiv: 2103.15396v2. URL: https://github.com/.

[21] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. *Feature Pyramid Networks for Object Detection*. Tech. rep.

[22] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. *Focal Loss for Dense Object Detection*. Tech. rep.

[23] Lindeberg, T. *Scale Invariant Feature Transform*. Tech. rep. 2012. URL: www.scholarpedia.org/article/Scale%7B%5C_%7DInvariant%7B%5C_%7DFeature%7B%5C_%7DTransform.

[24] Liu, Y.-C., Ma, C.-Y., He, Z., Kuo, C.-W., Chen, K., Zhang, P., Wu, B., Kira, Z., and Vajda, P. *UNBIASED TEACHER FOR SEMI-SUPERVISED OBJECT DETECTION*. Tech. rep. arXiv: 2102.09480v1. URL: https://github.com/facebookresearch/unbiased-teacher..

[25] Liu, Z., Tang, H., Lin, Y., and Han, S. *Point-Voxel CNN for Efficient 3D Deep Learning*. Tech. rep.

[26] Liu, Z., Tang, H., Lin, Y., and Han, S. *Point-Voxel CNN for Efficient 3D Deep Learning*. Tech. rep.

[27] Luo, H., Ji, L., Li, T., Duan, N., and Jiang, D. *GRACE: Gradient Harmonized and Cascaded Labeling for Aspect-based Sentiment Analysis*. Tech. rep. arXiv: 2009.10557v2.

[28] Luo, H., Ji, L., Li, T., Duan, N., and Jiang, D. *GRACE: Gradient Harmonized and Cascaded Labeling for Aspect-based Sentiment Analysis*. Tech. rep. arXiv: 2009.10557v2.

[29] Luo, H., Ji, L., Li, T., Duan, N., and Jiang, D. *GRACE: Gradient Harmonized and Cascaded Labeling for Aspect-based Sentiment Analysis*. Tech. rep. arXiv: 2009.10557v2.

[30] Mohapatra, S., Yogamani, S., Gotzig, H., Milz, S., and Mäder, P. *BEVDetNet: Bird's Eye View LiDAR Point Cloud based Real-time 3D Object Detection for Autonomous Driving*. Tech. rep. arXiv: 2104.10780v1.

[31] Noh, J., Lee, S., and Ham, B. *HVPR: Hybrid Voxel-Point Representation for Single-stage 3D Object Detection*. Tech. rep. arXiv: 2104.00902v1.

[32] Patil, A., Malla, S., Gang, H., and Chen, Y.-T. *The H3D Dataset for Full-Surround 3D Multi-Object Detection and Tracking in Crowded Urban Scenes*. Tech. rep. arXiv: 1903.01568v1. URL: http://www.ros.org/.

[33] Pham, Q.-H., Sevestre, P., Singh Pahwa, R., Zhan, H., Pang, C. H., Chen, Y., Mustafa, A., Chandrasekhar, V., and Lin, J. *A*3D Dataset: Towards Autonomous Driving in Challenging Environments*. Tech. rep. arXiv: 1909.07541v1. URL: https://github.com/I2RDL2/ASTAR-3D.

[34] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. Tech. rep. arXiv: 1612.00593v2.

[35] Ren, S., He, K., Girshick, R., and Sun, J. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. Tech. rep. arXiv: 1506.01497v3. URL: http://image-net.org/challenges/LSVRC/2015/results.

[36] Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., and Li, H. *PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection*. Tech. rep. arXiv: 1912.13192v1.

[37] Shi, S., Wang, X., and Li, H. *PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud*. Tech. rep. arXiv: 1812.04244v2. URL: https://github.com/sshaoshuai/PointRCNN..

[38] Shrivastava, A., Gupta, A., and Girshick, R. *Training Region-based Object Detectors with Online Hard Example Mining*. Tech. rep. arXiv: 1604.03540v1.

[39] Sohn, K., Zhang, Z., Li, C.-L., Zhang, H., Lee, C.-Y., and Pfister, T. *A Simple Semi-Supervised Learning Framework for Object Detection*. Tech. rep. arXiv: 2005.04757v2. URL: https://github..

[40] Song, S., Lichtenberg, S. P., and Xiao, J. *SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite*. Tech. rep. URL: http://rgbd.cs.princeton.edu.

[41] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhao, S., Cheng, S., Zhang, Y., Shlens, J., Chen, Z., and Anguelov, D. *Scalability in Perception for Autonomous Driving: Waymo Open Dataset*. Tech. rep. arXiv: 1912.04838v7. URL: http://www.waymo.com/open..

[42] Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. *Attention Is All You Need*. Tech. rep. arXiv: 1706.03762v5.

[43] Wang, H., Cong, Y., Litany, O., Gao, Y., and Guibas, L. J. *3DIoUMatch: Leveraging IoU Prediction for Semi-Supervised 3D Object Detection*. Tech. rep. arXiv: 2012.04355v2. URL: http://thu17cyz.github.io/3DIoUMatch.

[44] Wang, Y., Chen, X., You, Y., Li, L. E., Hariharan, B., Campbell, M., Weinberger, K. Q., and Chao, W.-L. *Train in Germany, Test in The USA: Making 3D Object Detectors Generalize*. Tech. rep. URL: https://www.best-selling-cars.com/germany/.

[45] Wang, Y., Fathi, A., Kundu, A., Ross, D. A., Pantofaru, C., Funkhouser, T., and Solomon, J. *Pillar-based Object Detection for Autonomous Driving*. Tech. rep. URL: https://github.com/WangYueFt/pillar-od..

[46] Xue, J., Fang, J., Li, T., Zhang, B., Zhang, P., Ye, Z., and Dou, J. *BLVD: Building A Large-scale 5D Semantics Benchmark for Autonomous Driving*. Tech. rep. arXiv: 1903.06405v1. URL: https://github.com/VCCIV/BLVD/..

[47] Yan, Y., Mao, Y., and Li, B. "SECOND: Sparsely Embedded Convolutional Detection". In: (). DOI: 10.3390/s18103337. URL: www.mdpi.com/journal/sensors.

[48] Yang, B., Luo, W., and Urtasun, R. *PIXOR: Real-time 3D Object Detection from Point Clouds*. Tech. rep. arXiv: 1902.06326v3.

[49] Yang, Z., Sun, Y., Liu, S., and Jia, J. "3DSSD: Point-Based 3D Single Stage Object Detector". In: (2020), pp. 11037–11045. DOI: 10.1109/cvpr42600.2020.01105. arXiv: 2002.10187.

[50] Yang, Z., Sun, Y., Liu, S., Shen, X., Jia, J., and Lab, Y. *STD: Sparse-to-Dense 3D Object Detector for Point Cloud*. Tech. rep. arXiv: 1907.10471v1.

[51] Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., and Darrell, T. *BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning*. Tech. rep. arXiv: 1805.04687v2. URL: https://www.getnexar.com.

[52] Zarzar, J., Giancola, S., and Ghanem, B. *PointRGCN: Graph Convolution Networks for 3D Vehicles Detection Refinement*. Tech. rep. arXiv: 1911.12236v1.

[53] Zhao, N., Chua, T.-S., and Lee, G. H. *SESS: Self-Ensembling Semi-Supervised 3D Object Detection*. Tech. rep. arXiv: 1912.11803v3. URL: https://github.com/Na-Z/sess..

[54] Zhao, X., Bai, X., Liu, Z., Huang, T., Hu, R., and Zhou, Y. *TANet: Robust 3D Object Detection from Point Clouds with Triple Attention Object Detection; Scene Text Recognition; Deep Learning View project Adversarial Learning Based Saliency Detection View project TANet: Robust 3D Object Detection from Point Clouds w*. Tech. rep. arXiv: 1912.05163v1. URL: www.aaai.org.

[55] Zheng, W., Tang, W., Chen, S., Jiang, L., and Fu, C.-W. *CIA-SSD: Confident IoU-Aware Single-Stage Object Detector From Point Cloud*. Tech. rep. 2021. arXiv: 2012.03015v1. URL: https://github.com/Vegeta2020/CIA-SSD..

[56] Zhou, Y., Sun, P., Zhang, Y., Anguelov, D., Gao, J., Ouyang, T., Guo, J., Ngiam, J., and Vasudevan, V. *End-to-End Multi-View Fusion for 3D Object Detection in LiDAR Point Clouds*. Tech. rep.

[57] Zhou, Y. and Tuzel, O. *VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection*. Tech. rep. arXiv: 1711.06396v1.

[58] Zimmer, W. *3D Bounding Box Annotation Tool (3D-BAT) Point cloud and Image Labeling*. 2020. URL: https://github.com/walzimmer/3d-bat.