3D Object Detection for Self-Driving Vehicles

by

Kiwoo Shin

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Fall 2020

3D Object Detection for Self-Driving Vehicles

Abstract

3D Object Detection for Self-Driving Vehicles

by

Kiwoo Shin

Doctor of Philosophy in Engineering - Mechanical Engineering

University of California, Berkeley

Professor Masayoshi Tomizuka, Chair

3D object detection systems based on deep neural network become a core component of self-driving vehicles. 3D object detection helps to understand the geometry of physical objects in 3D space that are important to predict future motion of objects. While there has been remarkable progress in the fields of image based 2D object detection and instance segmentation, 3D object detection is less explored in the literature.

This dissertation is concerned with various challenges in 3D object detection for self-driving vehicles. We mainly discuss how to improve the performance of 3D object detection system as well as the computational efficiency of the detection pipeline.

The scope of this research lies in the field of 2D camera image vision, 3D LiDAR point clouds processing, sensor-fusion based detection method, efficiency of detection pipeline, and novel data augmentation method for 3D LiDAR point clouds. While the primary research focus is on improvement of the detection performance in terms of precision and recall which are core metrics for object detection task, also emphasized is the importance of practicality of the proposed methods.

In Chapter 2, we discuss sensor-fusion based 3D object detection system. Sensor-fusion based detection system for self-driving vehicles becomes an essential component for safety of self-driving. It becomes crucial to find out optimal combination of multiple sensors to build more accurate and efficient detection system. We mainly discuss how to use 2D camera vision system and 3D LiDAR sensor together for accurate and efficient 3D object detection system.

We first explore various methods in monocular pose estimation. A monocular pose estimation is one of the research areas in computer vision. It predicts the location of objects in 3D space by using single camera image. While most monocular pose estimation based 3D object detection systems have not shown enough performance, the proposed approach in Chapter 2 utilizes the geometrical consistency assumption to narrow down the huge 3D search space

into smaller one. Then, the proposed approach applies PointNet for further refinement of a 3D bounding box coordinates.

The proposed approach in Chapter 2, **RoarNet**(: **R**egi**O**n **A**pproximation **R**efinement), shows one of the best performances in 3D object detection task in KITTI dataset, which is a standard benchmark for the self-driving vehicle detection.

In Chapter 3, we discuss about LiDAR point clouds based 3D object detection pipeline. We analyze the weakness of the most 3D object detection systems which use end-to-end detection pipeline for training and testing. One interesting observation on recent 3D object detection systems is that even though predicted objects are proximal to ground truth objects, many of those predictions are classified as false positive due to low quality in box regression task.

In this chapter, we introduce a practical method to improve the performance of 3D object detection system. The proposed approach, **epBRM_V1**(: **e**nd **p**oint **B**ox **R**egression **M**odule), aims at improving the quality of 3D bounding box regression task, thus increasing the overall performance of 3D object detection task. The proposed approach requires less than 1 hour of training time and only 12ms of additional latency to improve the performance of standard detection methods to the state-of-the-art 3D object detection methods.

In Chapter 4, we develop **epBRM_V2** which overcomes the limitation of epBRM_V1 and further improve the recall performance of the 3D object detection system. We improve the previous baseline, which is epBRM_V1, in two aspects: 1) building a more sophisticated network structure for box regression task to improve the representation power of the 3D LiDAR point clouds feature, 2) introducing a novel data augmentation method for 3D LiDAR point clouds using tracklet information. These two simple modifications enable us to get approximately 95% of recall and 80% of mAP which are 10.0% improvement in recall and 2.7% improvement in mAP compared to epBRM_V1.

To my family

Contents

List of Figures

List of Tables

Acknowledgments

I appreciate everyone for their support during my PhD at University of California, Berkeley. I have been fortunate to receive precious help and support from my advisor, committee members, Samsung Scholarship, internship companies, friends and my family.

First of all, I would like to thank my advisor Prof. Masayoshi Tomizuka. For last 5.5 years, he has been always supporting and motivating my PhD studies.

Also, I would like to thank my qualifying exam committees, Prof. Kameshwar Poolla, Prof. Anil Aswani, Prof. Mark Mueller, and Prof. Somayeh Sojoudi. I also would like to thank Prof. John Canny and Prof. Francesco Borrelli for serving as my dissertation committees.

I have been fortunate to be a member of Mechanical Systems Control lab and thank every member in MSC lab.

I greatly appreciate Samsung Scholarship for their financial support which covers my tuition and stipend for last 5 years.

I was fortunate to have internship opportunities at Aptiv and Lyft Level 5. I would like to thank my Aptiv internship collaborators Oscar Beijbom, Alex Lang, Sourabh Vora and Donghyun Won. I also would like to thank my Lyft Level 5 internship collaborators Jack Hwang and Kiril Vidimce.

I would like to thank my friends who have been with me for more than 5 years at Berkeley: John J Kim, Jinsol Kim, Hyung-taek Kim, Brian Shin, Tim Lee, Narae Lee, Yeojun Kim, Eric Choi, Saehong Park, Donggun Lee, Emmanuel Shin, Paul Kwon, Jongsang Suh, Keunhong Jeong, Kwangho Kim, Jeong-ah Lee, Chaewon Baek, Kyueui Lee, Hyojin Kim, Wonhee Koh, Kyu-nam Kim, Myounghwan Oh, Jaeyoung Choi, Seung-ah Choo, Soochan Chung, Kyeongsik Nam, Miyeon Lee, Wonjun Jo, Dohyung Kim, Joonwoo Kim, Hyungmook Kang, Yangjoon Kim, Sunyoung Kwon, Dongkeun Han, Hokeun Kim, Hyun-dong Ha, Philjun Kang, Sangjae Bae, Chanwoo Oh, Kwangmin Ryu, Joonhoon Kim, Hyemin Jeon, Seonah Moon, Keunwoo Lee, Minok Park, Donggyu Lim, Seokwoong Choi, Deokhwang Kwon, Jewon Kim, Hyangkeun Yoo, Dongwook Kim.

Finally, I would like to thank my father Dae-yeol Shin, mother Young-soon Park, and two sisters Seol-mi Shin and Seol-ah Shin. This dissertation is dedicated to you.

Chapter 1

Introduction

A self-driving technology is one of the most active research area in both academia and industry. A number of researchers in the fields of control, computer vision, artificial intelligence, signal processing, and software/hardware architecture have been developing cutting-edge methodologies for safe self-driving vehicles. Car makers, chip manufacturer, sensor component makers, and many software companies are heavily investing to become the leader of autonomous driving.

Recently, a remarkable progress in the field of artificial intelligence leads the rapid development of self-driving vehicles. The computer vision society has developed super-human level detection and segmentation methods which work in real-time. The human-robot interaction researches help to understand the intention of on-road agents such as vehicles, pedestrians, and cyclists and this helps to predict the future motion of each agent. The planning system based on model predictive control and reinforcement learning is used to improve the safety of self-driving systems.

Among the various components in self-driving system described above, we will mainly discuss detection systems for self-driving vehicles. The scope of our research lies in object detection, which classifies objects of interest and background by using camera image and/or LiDAR sensor. Depending on the system requirement and computing resources, the ontology of objects of interest could be different at each detection system. For example, a simple detection system considers only vehicles, pedestrian, and cyclists class while a more sophisticated detection system might use a fine-grained classification rule which discriminates van, car, truck, bus, etc. In this dissertation, we are interested more in the former case for research purpose.

The object detection is generally categorized into 2D object detection and 3D object detection. The main difference between 2D object detection system and 3D object detection system is that the latter one predicts the 3D geometry of the objects in 3D space while the former one detects the objects only on 2D image space. To predict the location of surrounding objects and plan a future trajectory of the ego-vehicles, the 3D object detection systems are more useful for self-driving vehicles than the 2D object detection systems.

There are generally 7 coordinates which are predicted from 3D object detection systems:

(x, y, z, height, width, length, heading). The **(x, y, z)** represent the location of objects in 3D space. Depending on the coordinate system, depth is represented by x (in LiDAR coordinate system) or z (in camera coordinate system). In this dissertation, we follow LiDAR coordinate system. The **(height, width, length)** represent the dimension of the objects. The **heading** represents the direction of the objects. The (z, height) is often omitted in a detection system which works in bird's eye view space.

There are three main sensors which are used for the 3D object detection systems: camera, LiDAR, and Radar. While we admit that a Radar is precious sensor for estimating the distance and velocity of moving objects, we focus only on camera sensor and LiDAR sensor for research purpose in this dissertation. These two sensors are different in many aspects, so we briefly summarize camera based detection system and LiDAR based detection system.

Camera based 3D object detection system

The camera based detection uses a camera as primary perception sensor. The camera is widely used for perceiving visual information. The camera sensors are capable of providing high resolution images and inexpensive. Therefore, the camera based detection systems have been explored by a number of computer vision researchers for a long time, which brings the biggest advantage of using camera based detection system. Not only the detection tasks which aim to predict tight rectangular bounding boxes around the objects, the high resolution images from camera sensor make it available to use segmentation techniques which is pixel-level classification task.

However, a detection system which relies only on camera images shows weakness in predicting the location of objects in 3D space which is critical to on-road self-driving. The **monocular pose estimation** is one of computer vision research area which predicts the 3D geometry of objects by using single camera image. From 60's, there have been numerous methodologies studied in the field of monocular pose estimation. However, those monocular pose estimation methods have not shown good performance in a 3D object detection benchmark.

Recently, more researchers focus on using two or more cameras together to overcome the limitation of monocular pose estimation methods. For example, two or more carefully calibrated cameras are used together to approximate the depth information of the objects based on triangular geometry. More recently, there are several researches to generate psuedo-lidar point clouds from two images based on deep neural networks and apply LiDAR-based detection method for 3D object detection task.

LiDAR based 3D object detection system

The LiDAR based object detection system uses 3D LiDAR point cloud which is reflected from obstacles. The 3D LiDAR point cloud is an un-ordered set of 3D LiDAR points. Each 3D LiDAR point is generally represented as (x, y, z, i) where (x, y, z) represent the 3D coordinates of the point and i represents the intensity of beam reflection.

The biggest advantage of using a LiDAR sensor is that the LiDAR provides highly precise spatial information which can greatly improve the localization performance. In fact, once you get a set of LiDAR points which belong to an object, you can approximate the location of the object by simply taking a median and/or mean value of those LiDAR points.

Another advantage of LiDAR sensors is that it provides consistent output regardless of light status. Unlike to the camera sensors which give dark images during the night time, the light status doesn't affect on the output of LiDAR sensors.

However, a sparsity of 3D LiDAR point clouds makes classification task difficult. A density of LiDAR point clouds becomes very sparse outside of 70-100m boundary from the LiDAR sensor. Therefore, only a few number of LiDAR points are reflected from the objects in far fields. Moreover, for the small objects like pedestrian and cyclist, a number of LiDAR points reflected from the objects is less than 20-50 in most cases even though those objects are located proximal to the LiDAR sensors. As a result, it is difficult to discriminate traffic pole and pedestrian using LiDAR point clouds even though they are located only 20-30m away from the ego-vehicle.

Sensor-fusion based 3D object detection system

As described above, a camera sensor and a LiDAR sensor cannot be the perfect sensor for self-driving vehicles when each sensor is used alone. The sensor-fusion based detection systems tries to overcome the limitation of each sensor by using both sensors together in a way that they can be supplement to each other.

For example, a high resolution images from camera sensor can be more useful for classifying object and non-objects than sparse LiDAR point clouds. On the other hand, a set of 3D LiDAR points can provide more precious information for regression task which predicts the location of object than camera images. Therefore, more researchers focus on sensor-fusion based detection system especially on how to build a better object detection pipeline using multiple sensors together.

Chapter 2. Sensor Fusion based 3D Object Detection for Self Driving Vehicles

In Chapter 2, we explore the sensor-fusion based detection systems for self driving vehicles. There can be various approaches of using multiple sensors together for making combined prediction. What is the best suite of sensors for accurate and efficient detection systems for self driving vehicles is still a controversial topic in both research academia and industry. This chapter is focusing on how to combine 2D camera vision and 3D LiDAR point clouds for predicting the precise coordinates of 3D cuboid of the objects.

We first explore the characteristics of each camera sensor and LiDAR sensor. While camera sensors can provide very dense and clear appearances of the objects, the lack of spatial information makes it difficult to infer precise location of the objects in 3D space.

In other hands, LiDAR sensors contain exact spatial information which makes it straightforward to predict the location of the objects while the sparsity of LiDAR point clouds may lead to mis-classification between objects and non-objects.

Focusing on these characteristics, we explore the various methods to build an unified detection system that each sensor can supplement the weakness of the other sensor. To bridge the 2D detection from camera vision and 3D detection from LiDAR sensor, we discuss the theoretical background behind the monocular pose estimation which uses geometrical consistency between 2D bounding boxes and their corresponding 3D cuboids.

In this chapter, we introduce three major contributions for sensor-fusion based detection system for self driving vehicles. First, we combine RoI pooling based feature extraction with monocular pose estimation algorithm. This modification removes the need to do iterative calculations required by the original monocular pose estimation, thus leads to faster monocular pose estimation.

Second, considering the prediction error from 2D object detector, we provide mathematical approach to get geometrically feasible locations where the objects can be located. This greatly narrows down the search space that the LiDAR point clouds based 3D object detectors are responsible for.

Lastly, we compare the proposed approach with other state-of-the-art 3D object detection systems. Not only the normal status when the multiple sensors are well synchronized, but we also consider more realistic status when the camera vision sensor and LiDAR sensors are not temporally synchronized. In both cases, the proposed approach shows the better performance than previous state-of-the-art 3D object detectors.

Chapter 3: In-depth Analysis on 3D Object Detection Pipeline and Practical Method to Improve the Quality of 3D Bounding Box Regression.

In Chapter 3, we discuss about how to improve the detection performance by increasing the quality of 3D bounding box regression. The 3D object detection for self driving vehicle is defined by the multi-tasks learning which predicts the location of objects as well as 3D bounding box coordinates of those objects.

Most recent 3D object detection systems which are based on deep neural networks use the end-to-end pipeline for training and evaluation. While building an end-to-end detection pipeline is usually preferable for faster inference time and training efficiency, in-depth analysis on comparing the localization performance and overall detection performance shows that the many of predicted objects which are located proximal to the ground truth are classified as false positive due to the low quality of bounding box regression. In other words, even though the detection network predicts the location of object pretty well, the overall detection performance is degraded due to low performance of box regression task.

In this chapter, we focus on developing a practical method to get the performance of existing detection networks improved. The practicality of the proposed approach can be highlighted from the three aspects: a) the proposed approach should not impose heavy computation in order to keep the real-time detection pipeline, b) the proposed approach should be easily attachable on top of existing detection networks. c) the network of the proposed approach should be easily trainable.

Based on the analysis which shows the box regression task performs poorly, we separate the detection pipeline into a localization module, which is responsible for predicting the location of objects and probability of objects' existence, and a box regression module which is responsible for predicting the precise coordinates of 3D bounding boxes. We use the detection results from the existing detection networks as localization purpose and we measure how much performance gain can we achieve by designing a box regression network.

To keep the box regression network small and computationally light weighted, the proposed approach adopts spatial transformation mechanism into the regression pipeline. The spatial transformation mechanism is the core idea to reduce the complexity of the task in order to make even a small sized network be capable of predicting the precise coordinates of the 3D bounding boxes. Runtime analysis shows that it takes less than 12ms for up to 20 objects around the ego-vehicle.

Since the proposed approach builds a small network for box regression task, it brings us another benefit such that training a network takes really short time. We show that 30 minutes of training is enough for improving the detection performance to the stat-of-the-art level.

The proposed approach can be attached to various existing detection networks which have all different sensor configurations. We evaluate the proposed approach by using Frustum-PointNet, AVOD-FPN, and PointPillars which all have different approach for 3D object detection systems. Our analysis on validation set shows that the proposed approach performs better than the state-of-the-art detection networks with various sensor configurations.

Chapter 4: Novel Data Augmentation Method for 3D LiDAR Point Clouds using Tracklet information for Improved Recall Performance.

In Chapter 4, we discuss a novel data augmentation method for 3D LiDAR point clouds to improve the performance of 3D object detection system. It is expensive to get ground truth labels from 3D LiDAR point clouds due to its sparsity and huge 3D space. While self-driving companies and research institutes release several open dataset to the public, the number of labels we can utilize are still very limited.

A data augmentation method is one of core techniques for training a deep neural network when the number of training samples is limited. By applying several simple transformation methods on input images and labels, a data augmentation method increases the number of training samples which leads to generalization of the trained network.

While there have been numerous data augmentation methods explored for camera images, a data augmentation method for 3D LiDAR point clouds has been less explored in the literature. Most commonly used data augmentation methods for 3D LiDAR point clouds are horizontal/vertical flipping, global translation/rotation, jittering, frustum-cutout.

In this chapter, we introduce a novel data augmentation method for 3D LiDAR point clouds which greatly improves the performance of 3D object detection task. The proposed method utilizes the tracklet information which tracks a single object for multiple time stamps. By using tracklet information, the proposed method can build very dense point clouds of the objects.

The core idea behind the proposed data-driven augmentation method is based on mixup between dense set of point clouds with ground truth labels and the sparse set of point clouds sampled from training dataset. The proposed method can create a new set of 3D LiDAR point clouds which has realistic pattern of 3D LiDAR point clouds as well as ground truth label to be used as training samples.

We show by experiments that the proposed data augmentation method can improve the box regression performance. And combined with the new localization method, the 3D object detection performance of the proposed approach can become a comparable to the state-of-the-art 3D object detection systems which rely on very complicated network structures.

Conclusion

The main focus of this thesis is to explore various methods in 3D object detection for self-driving vehicles. Throughout the thesis, we explore the sensor-fusion based detection network which uses 2D camera vision and 3D LiDAR sensor together, 3D LiDAR point clouds based detection network, and data augmentation method for 3D LiDAR point clouds to enhance feature representation and generalization of the detection networks. We hope our research can contribute to the future research in 3D object detection for self-driving vehicles.

Chapter 2

Sensor Fusion for 3D Object Detection

2.1 Introduction

In this chapter, we mainly discuss about sensor-fusion based 3D object detection method. We present **RoarNet**, a novel approach for 3D object detection based on sensor-fusion between 2D image and 3D LiDAR point clouds. Based on two stage object detection framework ([16, 44]) with PointNet [39] as our backbone network, we explore several novel ideas to improve 3D object detection performance.

The first part of our method, RoarNet_2D, estimates the 3D poses of objects from a monocular image, which approximates where to examine further, and derives multiple candidates that are geometrically feasible. This step significantly narrows down feasible 3D regions, which otherwise requires demanding processing of 3D point clouds in a huge search space.

Then the second part, RoarNet_3D, takes the candidate regions and conducts in-depth inferences to conclude final poses in a recursive manner. Inspired by PointNet, RoarNet_3D processes 3D point clouds directly without any loss of data, leading to precise detection.

We evaluate our method in KITTI, a 3D object detection benchmark. Our result shows that RoarNet has superior performance to state-of-the-art methods that are publicly available. Remarkably, RoarNet also outperforms state-of-the-art methods even in settings where Lidar and camera are not time synchronized, which is practically important for actual driving environment.

2.2 Contribution

Recently, 3D object detection has become a crucial component in various fields such as mobile robots and autonomous vehicles. 3D object detection helps to understand the geometry of physical objects in 3D space that are important to predict future motion of objects. While there has been remarkable progress in the fields of image based 2D object detection and instance segmentation, 3D object detection is less explored in the literature.

3D detection algorithms can be divided into two groups: LiDAR-based approaches and sensor-fusion-based approaches. LiDAR-based approaches rely solely on 3D point clouds, where 3D point clouds may be used directly [73], or manipulated including projections onto ground plane (i.e., bird's eye view), depth maps (i.e., perspective view) [67, 34, 11, 51], etc.

In contrast, sensor-fusion-based approaches exploit both image and 3D point clouds [**iv18cwang**, **iv18zwang**, 38, 24, 7, 31]. Sensor-fusion-based approaches have several advantages: (1) Using complimentary sensors can be more robust since each sensor has its own weakness. (2) Within autonomous driving applications, computer vision algorithms conduct many more tasks including lane line detection, traffic light recognition, object detection, etc. Cameras are regarded as essential sensors, and there is little to no cost in using the images. (3) Cameras typically output images around at 30 fps, whereas LiDAR outputs point clouds around at 100 fps. Considering the safety aspect of autonomous driving, latency is very important, and image-based object detection is inevitable. A good 3D detection algorithm should exploit such advantages.

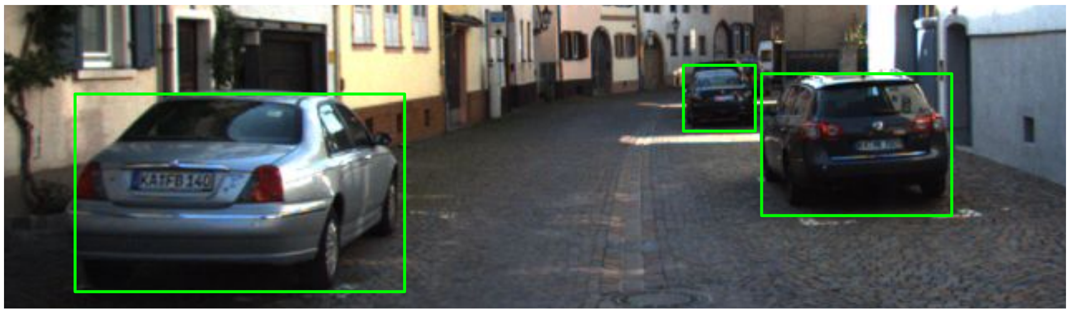
In this work, we propose a robust 3D fusion-based detector, named **RoarNet** (**R**egi**O**n **A**pproximation **R**efinement Network), which helps to improve 3D object detection performance and reduce problems caused by sensor synchronization issue. RoarNet consists of two parts: RoarNet_2D and RoarNet_3D.

The detection pipeline of our model consists of three components as in Figure 2.1. Inspired by geometric interpretation for monocular images in [36], RoarNet_2D estimates the 3D poses of objects from a monocular image and derives multiple candidate locations that are geometrically feasible, where the candidates are the input for RoarNet_3D. This scheme significantly narrows down feasible 3D regions, which otherwise requires demanding processing of 3D point clouds in a huge search space (Section 2.5).

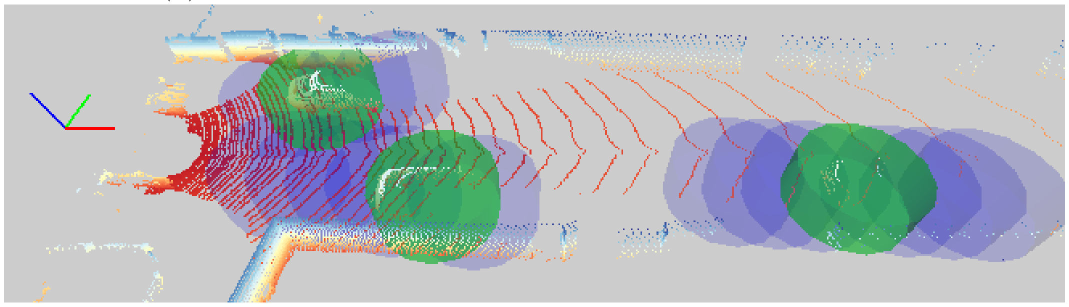
Obtaining 3D region proposals predicted from 2D image, RoarNet_3D, a two-stage 3D object detector, gradually refines a search space making its training process efficient. The architecture of our model is analogous to standard two stage object detectors for 2D image such as Fast-RCNN and Faster-RCNN [16, 44], and we adopt several modifications in order to make training of each stage easier (Section 2.6).

The key difference compared to [38] is that our model does not filter out point clouds by using 2D bounding box. Instead, our model takes the whole point clouds that are located inside region proposals which have the shape of standing cylinders. This leads to our model being more robust to sensor synchronization than state-of-the-art methods. We compare our method to other state-of-the-art 3D detection models in both synchronized and asynchronized conditions in Section 3.8.

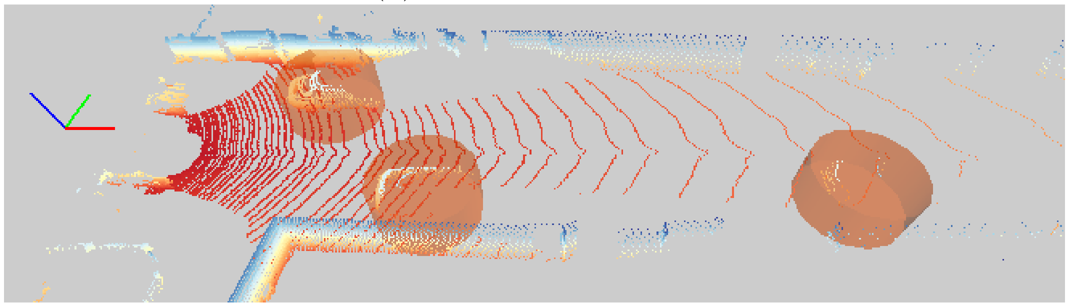
We evaluate our model on the 3D object detection task, provided by the KITTI benchmark, and our experiments show that RoarNet outperforms the state-of-the-art fusion-based 3D object detection methods that are publicly available. We also evaluate our model in settings where camera and the Lidar are not time synchronized and the result shows that our model consistently performs better in these challenging settings.



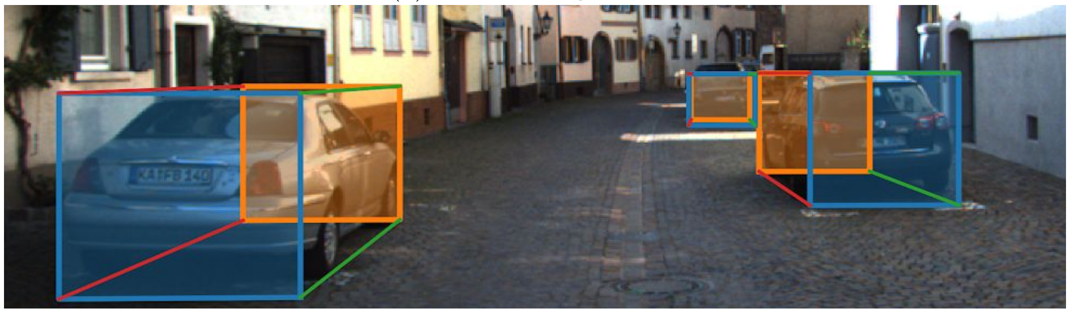
(a) geometric agreement search on 2D object detection



(b) 3D region proposals



(c) 3D box regression



(d) resulting 3D bounding boxes

Figure 2.1: Detection pipeline of RoarNet. Our model (a) predicts region proposals in 3D space using geometric agreement search, (b) predicts objectness in each region proposal, (c) predicts 3D bounding boxes, (d) calculates IoU (Intersection over Union) between 2D detection and 3D detection.

2.3 Related work

Monocular pose estimation

Due to the projection characteristics of camera sensors, monocular 3D pose estimation is very challenging. To overcome such difficulty, previous works often rely on domain knowledge or external data/information. [5] trains a network to predict 36 control points per each vehicle that conveys 3D shape information.

However, this method requires additionally annotating the auxiliary control point, which are very expensive to obtain. [36] proposes a novel method to predict physical dimensions (i.e, height, width, length in meters) and an orientation of vehicle without any additional data. Then, it can predict the location of object (i.e., X, Y, Z in the world coordinate) by solving an over-constrained system of linear equations system. Since we find this method useful, we explore the method in more detail in Section 2.5 where we modify the method to be more computationally efficient.

3D point clouds processing

Since autonomous driving applications require very high level of accuracy in 3D pose estimation that monocular algorithms cannot provide, many algorithms using Lidar sensors are proposed. There are three popular representations to handle unstructured point clouds: (1) The first representation is using a 3D voxel grid [71, 28, 11, 45, 65, 25, 46, 4, 43, 30, 68, 54, 10, 53, 51]. In autonomous driving applications, however, sparse points clouds generally make voxel representation computationally redundant. (2) The second is to project an point cloud onto one or more 2D planes [34, 51, 67]. These representations are usually compact and efficient, and can be treated as images. However, information loss by projection is inevitable. (3) The third one is to use the point clouds directly without any structured form.

PointNet [39, 40] showed how to digest point clouds directly for object classification and segmentation, and Frustum PointNet (F-PointNet) [38] selects only necessary 3D points utilizing 2D detection results (i.e., 3D points within a frustum region that a camera position and a 2D bounding box make), and conducts detection using a PointNet scheme.

F-PointNet [38] and Aggregate View Object Detection (AVOD) [24] show the state-of-the-art performance on the public KITTI dataset leader board. RoarNet outperforms these methods in the standard 3D object detection, and our analysis shows that RoarNet shows better robustness in an even more general setting.

2.4 Designing a RoarNet detector

The main idea behind RoarNet is to construct sequential networks that gradually refines a search space at each step in order to assign each network a simple task, and thus leads to efficient training and prediction.

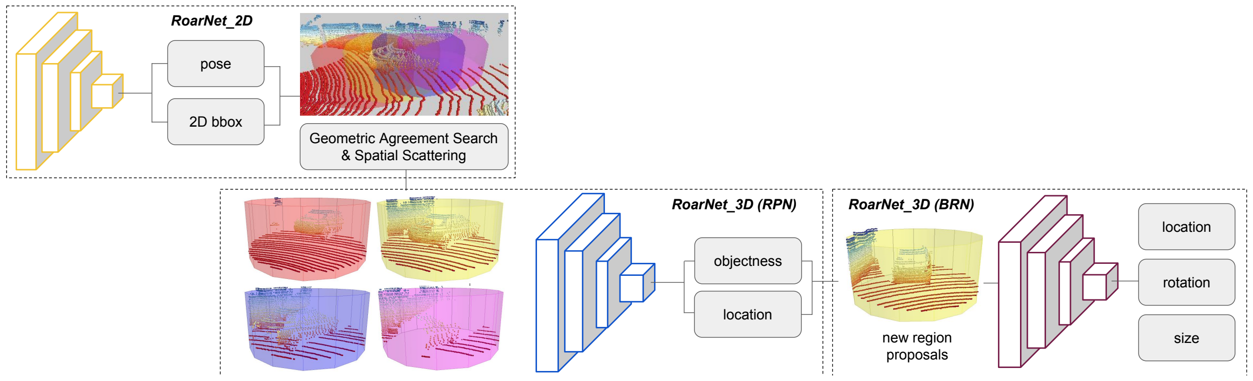


Figure 2.2: Architecture of RoarNet

RoarNet is structured to utilize advantage of both 2D image and 3D point clouds, those contain supplementary information for 3D object detection. On the one hand, 2D image with high resolution provides reliable objectness information. On the other hand, point clouds provides exact spatial information of the objects. We design our model based on these characteristics.

Figure 2.2 shows the architecture of RoarNet. The model first predicts the 2D bounding boxes and a 3D poses of objects from a 2D image. For each 2D object detection, geometric agreement search is applied to predict the location of object in 3D space.

The inverse projection method is essentially based on a monocular pose estimation algorithm and we discuss a mathematical formulation behind the inverse projection method in a more detail in Section 2.5.

Centered on each location prediction, we set region proposal which has a shape of standing cylinder. Taking the prediction error in bounding box and pose into account, there can be multiple region proposals for a single object.

Each region proposal is responsible for detecting a single object. Taking the point clouds sampled from each region proposal as input, our model predicts the location of an object relative to the center of region proposal, which recursively serves for setting new region proposals for the next step.

Our model also predicts objectness score which reflects the probability of an object being inside the region proposal. Only those proposals with high objectness scores are considered at the next step.

At a final step, the model sets new region proposals at previously predicted locations. Our model predicts all coordinates required for 3D bounding box regression including location, rotation, and size of the objects.

2.5 RoarNet_2D

Geometric agreement search

For our initial seeds of 3D region proposals, we utilize a method suggested by [36] for monocular pose estimation, which we call *geometric agreement search*:

Given that the 3D pose of an object can be represented by seven degrees of freedom (localization in the camera coordinate X, Y, Z, physical dimensions of width, height and length W, H, L, and heading angle Θ), a 2D bounding box window and the projection of its 3D pose (i.e., 3D box formed by X, Y, Z, W, H, L, Θ and camera projection matrix P) should agree.

[36] showed that (1) a network can regress $\{W, H, L, \Theta\}$ per object, (2) there are only finite number of possible combinatorial configurations that a 3D box can locate to tightly fit a given 2D box, and (3) at each configuration, translation X, Y, Z can be solved from known (regressed) W, H, L, Θ using an over-constrained system of linear equations. Then, the best configuration that minimizes projection error is selected.

More formally, for an object, let b_{2D} be its 2D bounding box (from a 2D detector). At each configuration c, one can calculate a 3D bounding box candidate b^c_{3D} as

$$b^c_{3D} = B(W, H, L, \Theta; c, b_{2D}) \tag{2.1}$$

where B is the over-constrained linear equation system aforementioned. The best configuration c^* can be obtained by checking the agreement between b_{2D} and the projection of 3D box b^c_{3D}.

$$b^c_{PROJ} = T(b^c_{3D}; P) \tag{2.2}$$

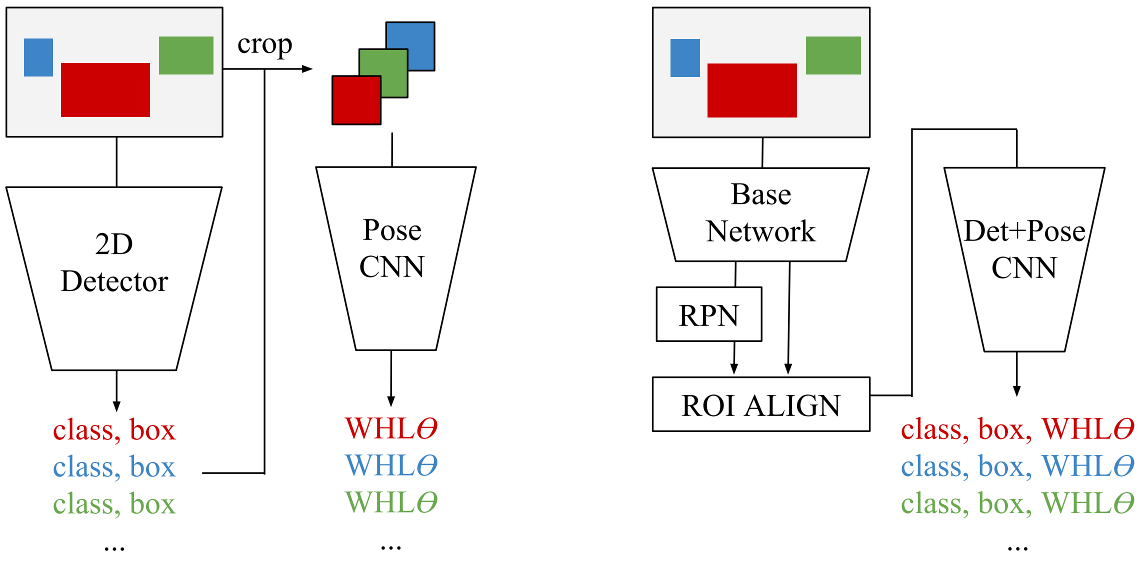
$$c^* = \arg\max_{c \in C} \text{IoU}(b_{2D}, b^c_{PROJ}) \tag{2.3}$$

where T is projective transformation onto the image coordinate, IoU is a widely-used intersection-over-union measure, and C is the finite configuration set.[1]

We illustrate the network structure of the original geometric agreement search by [36] and our improved version in Figure 2.3. One drawback of [36] is that the $\{W, H, L, \Theta\}$ inference and inverse projection process should be done after running a separate 2D object detection and should be conducted for each detected vehicle (Figure 3.5a).

In other words, when an image includes k objects, there should k-time computation of the network. Aiming better computation efficiency, we build an unified network that combines the 2D object detection and $\{W, H, L, \Theta\}$ inference as illustrated in Figure 3.5c. The 2D bounding boxes and $\{W, H, L, \Theta\}$s of k objects can be inferred with only one forward

[1]We refer [36] for the details about the configuration set C, and the over-constrained system of linear equations B.



(a) Architecture in [36]

(b) RoarNet_2D architecture

Figure 2.3: Architecture of RoarNet_2D

calculation of the unified network. This improvement is comparable with the improvement made by Faster-RCNN [44] over Fast-RCNN [16].

Specifically, the original algorithms takes extra 20ms (when $k = 3$) $\sim$ 72ms (when $k = 9$) for pose prediction, whereas the counterpart of our architecture only takes around 1$\sim$2ms to its 2D Faster-RCNN detection regardless of k. [2]

Spatial scattering

Note that the role of RoarNet_2D, as a 3D region proposer, is to provide proposals of higher recall. Since the monocular pose estimation suffers from limited accuracy, it is necessary to scatter our initial monocular pose estimation in order to increase the number of feasible pose candidates, and therefore, increase recall: For each object (i.e., its bounding box b_{2D}, regressed pose $XYZWHL\Theta$, and the best configuration c^*), we first set a *scattering range* by considering two extreme cases where the true physical size could actually be $1 - s$ times smaller and $1 + s$ times larger than the regressed size WHL ($0 < s < 1$), which results in differently located 3D boxes by Equation (2.1):

[2]Admittedly, the exact computation time comparison may be ambiguous since [5] uses a separate 2D detector, and one can use a very light-weight detector, whereas our unified architecture contains a built-in FasterRCNN detector. For this reason, So we discuss only the overhead part of the pose inference part.

$$b_{3D}^{c^*\ small} = B((1-s)W,(1-s)H,(1-s)L,\Theta;c^*,b_{2D}) \tag{2.4}$$

$$b_{3D}^{c^*\ large} = B((1+s)W,(1+s)H,(1+s)L,\Theta;c^*,b_{2D}) \tag{2.5}$$

Recall that Equation (2.1) means the geometric constraint that the projection of the 3D box of an object should match with its 2D box, i.e., for the same 2D bounding box, smaller 3D boxes result in closer locations to the camera origin. Given these two extreme boxes, we divide the line of their two center points, $p1$ and $p2$, into an equal stride distance m. RoarNet_2D detector finally provides $\lceil \|p_1 - p_2\|/m \rceil$ 3D points per object for RoarNet_3D to start.[3]

We visualize the process of RoarNet_2D detector in Figure 2.4. RoarNet_2D detector predicts 2D bounding boxes (Figure 2.4a) as well as their physical sizes WHL and heading angles Θ, which lead to calculate their positions XYZ (color-filled boxes in Figure 2.4b). For each object, we consider two extreme deviations (non-filled boxes in Figure 2.4b), and collect the uniform linear subdivision between the center points of the extreme poses (colored dots in Figure 2.4b).

Note that the geometric agreement search and spatial scattering scheme significantly narrows down feasible 3D regions into a few linear regions, which otherwise requires a huge search space. Moreover, by virtue of geometric agreement constraints, our resulting proposals natively distribute (1) along the projection rays of the camera, and (2) in larger areas for more challenging further objects without bells and whistles.

2.6 RoarNet_3D

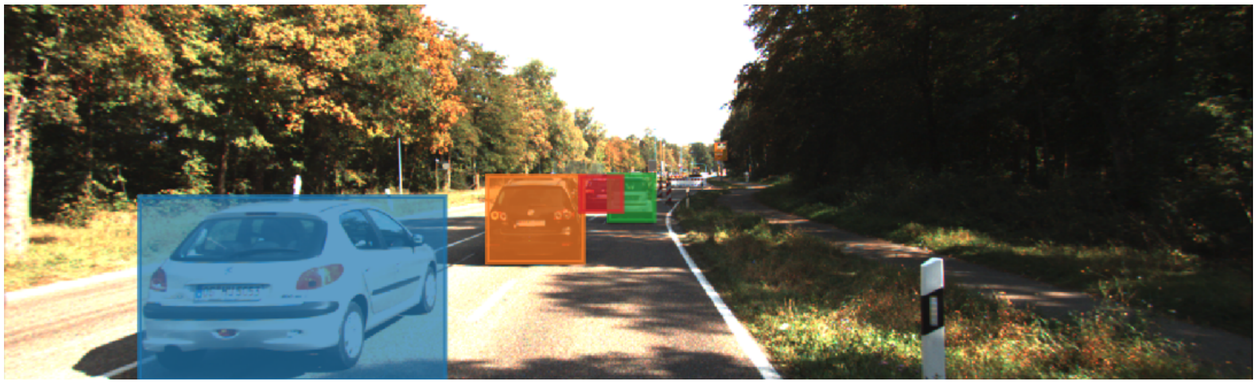
Network architecture

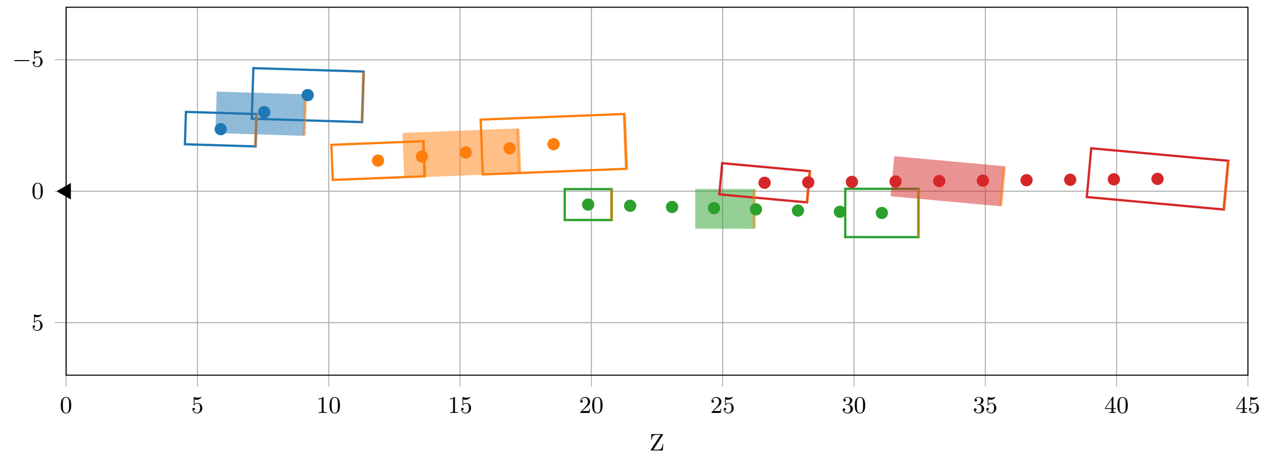
The RoarNet_3D is designed to predict a 3D bounding box that optimally fits for a given object by using point clouds. While building RoarNet_3D as a two-stage object detector, the backbone network is inspired by the PointNet[39], which uses max-pooling layers in the middle to get a global feature directly from unstructured point clouds. For more details, we refer readers to [39, 38, 40]. In this work, we use a simplified version of PointNet shown in Figure 2.5.

RoarNet_3D consists of two networks, called RPN (region proposal network) and BRN (box regression network), those have same structure except for the number of output as shown in Figure 2.5 and Table 2.1.

The location is predicted by 3 coordinates (t_x, t_y, t_z) for (x, y, z) directions which is relative to center of region proposals. If a center of region proposal is offset from the origin

[3] $s = .5, m = 1.6$ for experiments; $s = .2, m = 1.25$ for Figure 2.3.


(a) 2D detection



(b) Geometric agreement search and spatial scattering

Figure 2.4: RoarNet_2D. An unified architecture detects 2D bounding boxes and 3D poses illustrated as color-filled boxes in (a) and (b), respectively. For each object, two extreme cases are shown as non-filled boxes, and final equally-spaced candidate locations as colored dots in (b).

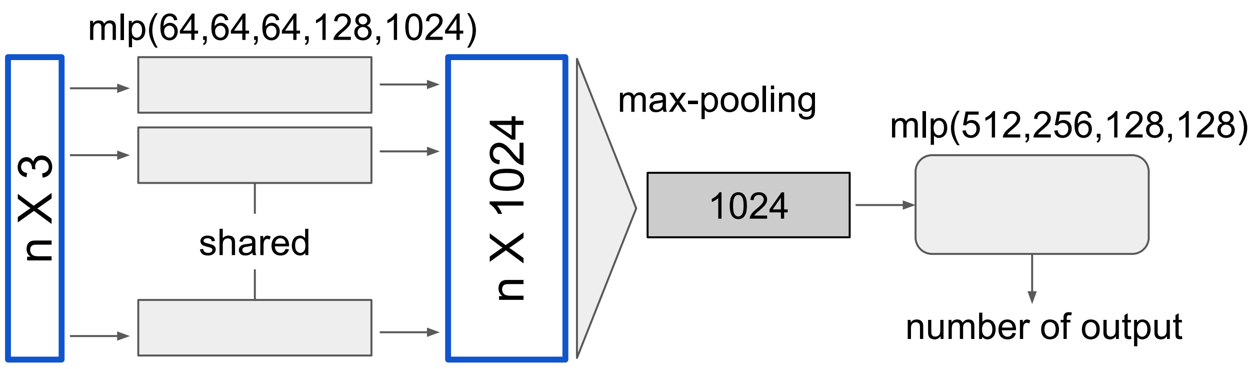


Figure 2.5: Our backbone network is a simplified version of PointNet without T-Net in the original paper [39].

Number of Outputs	RPN	BRN
location	3	3
rotation	0	2*N_R
size	0	4*N_C
objectness	1	0

Table 2.1: Number of output at each network

by (c_x, c_y, c_z), then the location prediction corresponds to:

$$x = c_x + 2 * (\sigma(t_x) \text{ - } 0.5) * m_x \tag{2.6}$$

$$y = c_y + 2 * (\sigma(t_y) \text{ - } 0.5) * m_y \tag{2.7}$$

$$z = c_z + 2 * (\sigma(t_z) \text{ - } 0.5) * m_z \tag{2.8}$$

We constrain the location prediction be bounded by (m_x, m_y, m_z) from center of region proposal.

The rotation angle is predicted by 2*N_R coordinates $(t_{r\_\text{cls}(i)}, t_{r\_\text{reg}(i)})_{i=1}^{N_R}$ which is a hybrid formulation of <cls+reg> structure. We equally divide [0, pi) to N_R bins.

The size is predicted by 4*N_C coordinates, $(t_{size\_cls(i)}, t_{h(i)}, t_{w(i)}, t_{l(i)})_{i=1}^{N_C}$ which is also a hybrid formulation of <cls+reg> structure. We use K-Means method to get N_C clusters.

The objectness is predicted by the output t_o which reflects the probability of object or not object for each region proposal. We use sigmoid function to bound its value in a range of [0.0, 1.0).

2.7 Training and prediction

During training each network, we optimize the following multi-task loss for RPN and BRN:

$$L_{\text{RPN}} = \lambda_{\text{obj}} * L_{\text{obj}} + \mathbb{1}^{\text{obj}}[L_{\text{loc}}] \tag{2.9}$$

$$L_{\text{BRN}} = \mathbb{1}^{\text{3D IoU}<0.8}\Big[L_{\text{loc}} + L_{\text{rot-cls}} + \mathbb{1}^{\text{rot-cls}}[L_{\text{rot-reg}}] + L_{\text{size-cls}} + \mathbb{1}^{\text{size-cls}}[L_{\text{size-reg}}]\Big] \tag{2.10}$$

$L_{\text{loc}}, L_{\text{rot-reg}}$, and $L_{\text{size-reg}}$ are regression loss for location, rotation and size, which are represented as huber loss. $L_{\text{obj}}, L_{\text{rot-cls}}$, and $L_{\text{size-cls}}$ are classification loss for objectness, rotation and size, which are represented as cross-entropy loss. $\mathbb{1}^{\text{obj}}$ denotes if objectness is true for a given region proposal. $\mathbb{1}^{\text{3D IoU}<0.8}$ is used for improving prediction performance for more general case.

We train each network with batch of 512 for 500k iterations. Learning rate is 5e-3 for initial 100k and 5e-4 for rest of steps. It takes about two days for training each network with Titan X (not pascal).

Non-maximal suppression (NMS) is used to reduce redundant prediction at testing. We apply NMS on bird's eye view boxes with threshold of 0.05 to remove overlapping objects.

2.8 Dataset

We conduct our experiments in KITTI dataset, the 3D object detection benchmark. It provides synchronized 2D images and 3D LiDAR point clouds with annotations for car, pedestrian, and cyclist class. In this work, we focus on car class which has most training examples.

The detection results are evaluated based on three difficulty levels: easy, moderate, and hard and we evaluate on moderate level, a standard metric for performance evaluation.

3D object detection performance is evaluated at 0.7 IoU threshold. Following [38, 24, 3], we split training set into train set of 3,717 frames and val set of 3,769 frames such that frames in each split belong to different video clips.

2.9 Experiment settings

We evaluate our method in two settings.

First, we evaluate our method in the original KITTI evaluation setting where the Lidar and the camera are well-synchronized each frame. This is a standard metric for ranking in KITTI benchmark leaderboard.

Second, we evaluate our method in a more general case where the two sensors are not synchronized. As we discussed in Introduction, in actual driving situations, images are consumed for various tasks including traffic light recognition, lane line detection, object

Method	Easy	Moderate	Hard
MV3D [7]	71.09	62.35	55.12
VoxelNet [73]	77.47	65.11	57.73
UberATG-ContFuse [31]	82.54	66.22	64.04
F-PointNet (v2) [38]	81.20	70.39	62.19
AVOD (FPN) [24]	81.94	71.88	66.38
RoarNet	**84.25**	**74.29**	**59.78**

Table 2.2: 3D object detection performance publicly available on the KITTI *test* set, with 3D IoU threshold of 0.7

detection, object tracking, etc. at their maximum frequency, and one can not assume a camera sensor and a Lidar sensor are perfectly synchronized to each other. Typically camera frequency is 30 fps, which is higher than that of Lidar. So there could be at most around 30 ms time discrepancy between a set of point clouds and its corresponding image.

To simulate such cases, we randomly translate the whole point clouds and re-generate ground truth labels according to the amount of translation of point clouds. This means that we regard the Lidar as the primary sensor. We constrain the translation of point clouds within 0.8m for x, y axis (i.e., parallel to the ground plane) and 0.2m for z axis (i.e., orthogonal to the ground plane).

2.10 Experiment results

Comparison of the 3D object detection performance

First, we evaluate RoarNet in a setting where the Lidar and the camera are synchronized, and compare it to publicly available 3D object detection methods on the KITTI benchmark. Table 2.2 shows that RoarNet shows state-of-the-art performance for 3D object detection in both easy and moderate level metric.

Second, we compare RoarNet to the two state-of-the-art methods, AVOD (FPN) and F-PointNet (v1) in a setting where sensors are *not* synchronized.

Figure 2.6 shows that RoarNet performs better than two state-of-the-art methods when

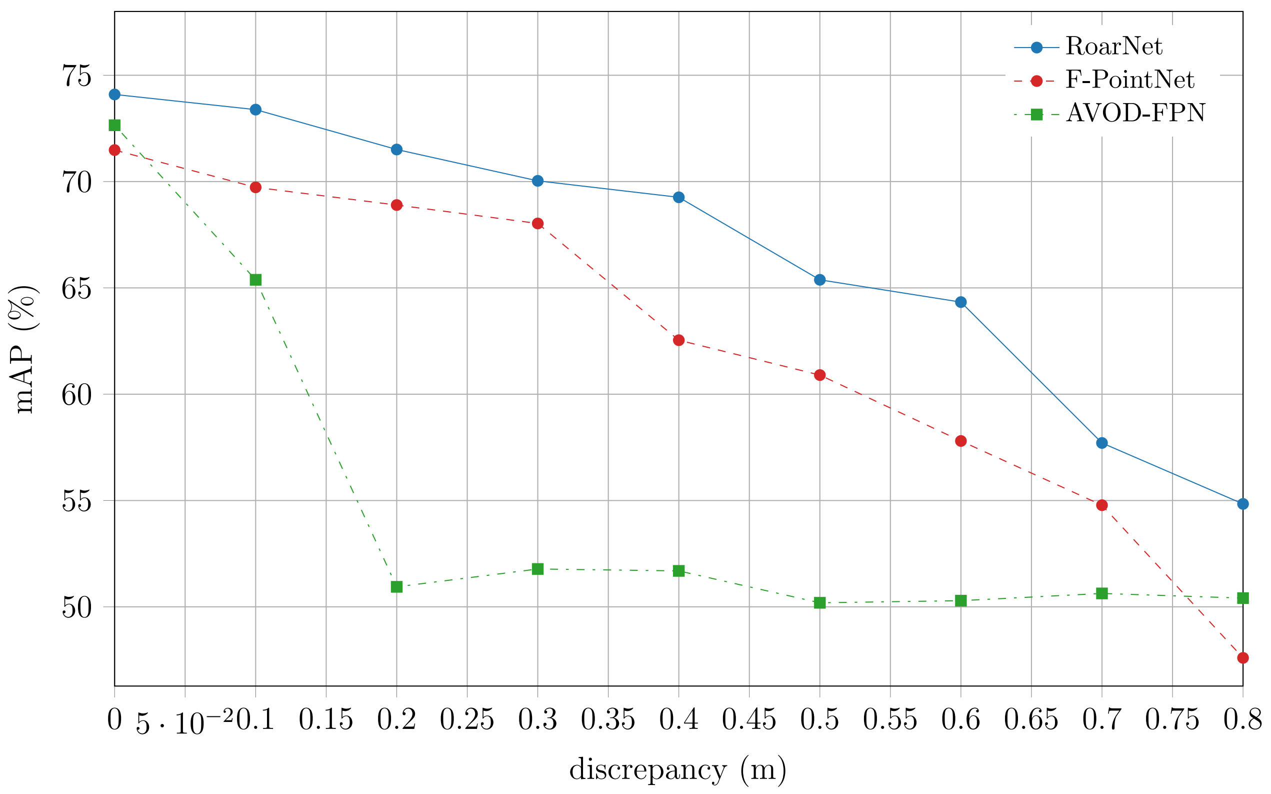


Figure 2.6: A comparison of the 3D object detection performance in where Lidar and camera are not time synchronized.

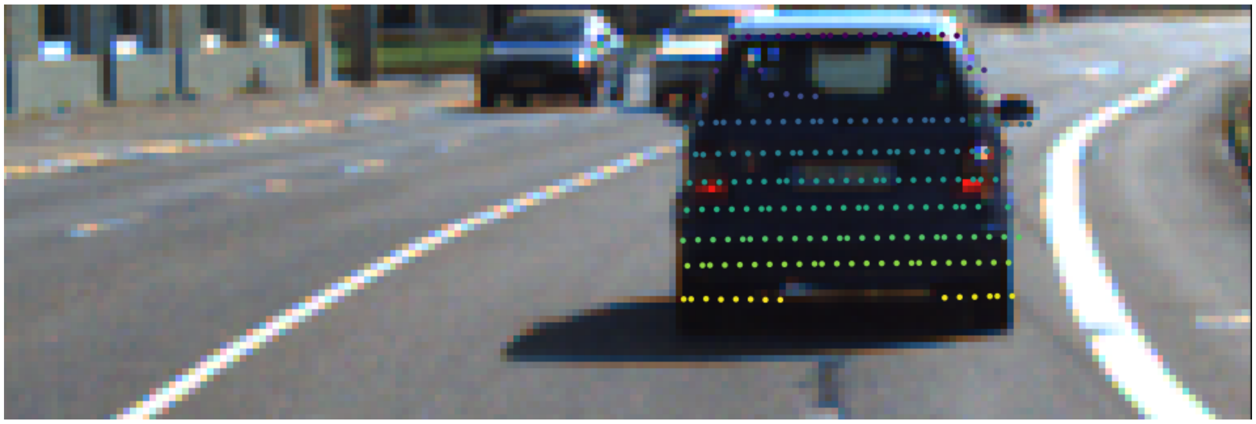
two sensors are not synchronized. When sensors are synchronized, all three methods show the recall of 82.5%. When two sensors are a-synchronized by 0.8m, the recall of our model degrades to 72.5%, while the recall of F-PointNet degrades to 67.5% and the recall of AVOD (FPN) degrades to 65%.

Region proposals analysis

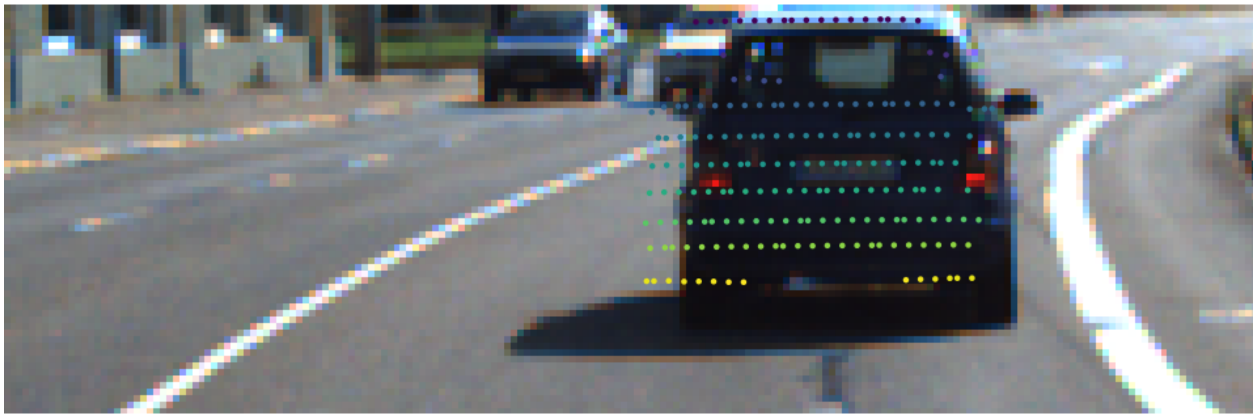
In this section, we analyze the effect of spatial scattering parameter s and objectness threshold in RoarNet_3D (RPN) for refining a search space, as shown in Figure 2.8.

The smaller the value s, the higher confidence we have on monocular pose estimation. However, only 26.3% of objects are captured in region proposals when we predict the location of object directly from monocular pose estimation ($s = 0$). As we increase s, more objects are captured in region proposals, but number of region proposals are also linearly increased, which becomes the bottleneck of our detection pipeline. Aiming high recall, we use $s = 0.5$ in our implementation.

The search space is further refined by RoarNet_3D (RPN). In our implementation, we use



(a) synchronization between camera and LiDAR timestamp



(b) asynchronization between camera and LiDAR timestamp

Figure 2.7: Example of simulation for time discrepancy between camera and LiDAR sensors

objectness threshold of 0.25, that gives 83.2% of recall with less than two region proposals per ground truth object.

Network design analysis

In this section, we compare three network architectural designs shown in Figure 2.9.

Figure 2.9(a) represents a single stage 3D object detector, which predicts 3D bounding box along with objectness in a single step. This approach is inspired by YOLO detector [42, 41], which shows promising results in a 2D object detection. However, (a) shows the recall of 67.5% and mAP of 54.3%.

Without any further training step, we only modify the detection pipeline of (a) to use location predicted at current step as region proposals for the next step. This simple mod-

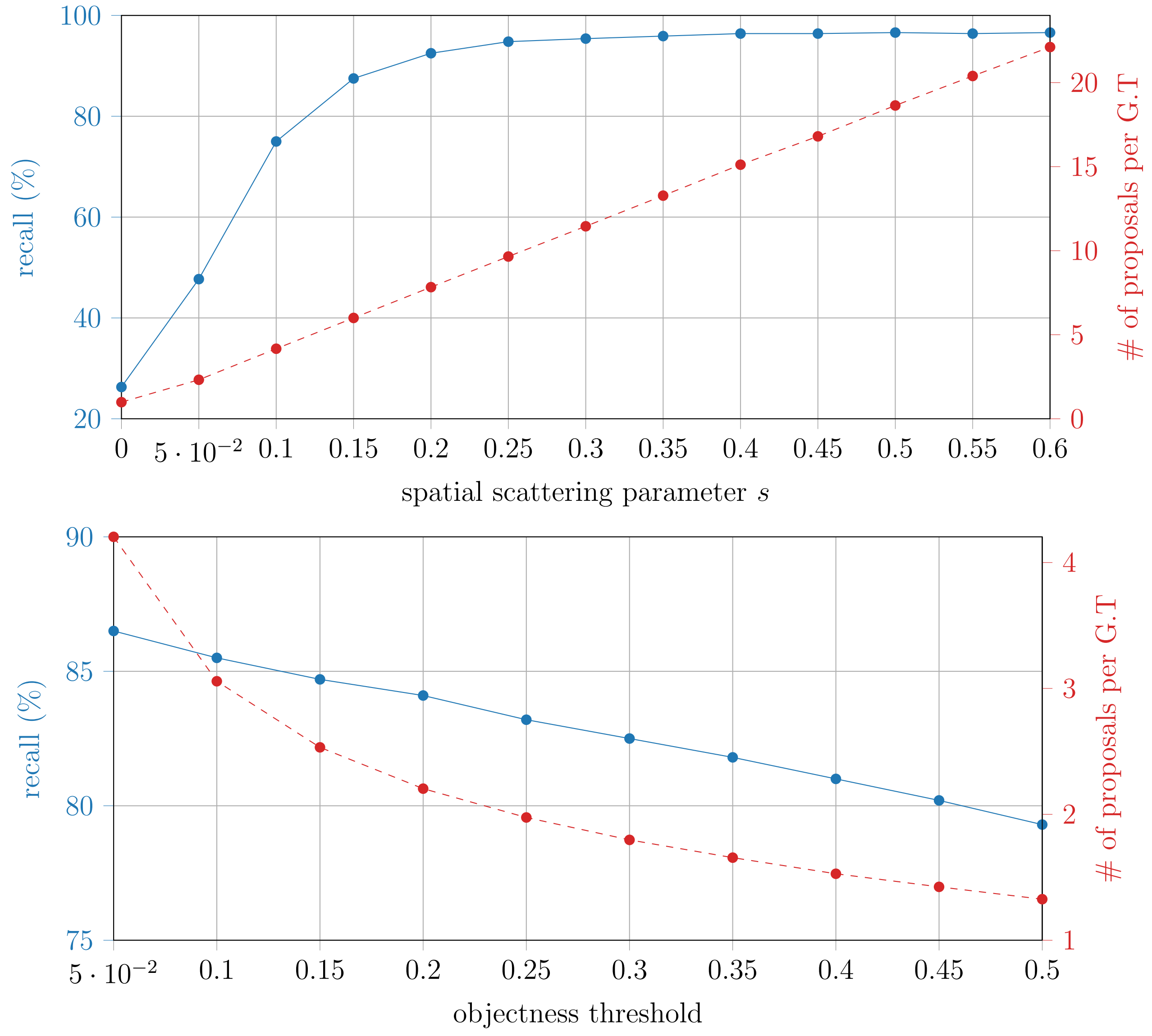
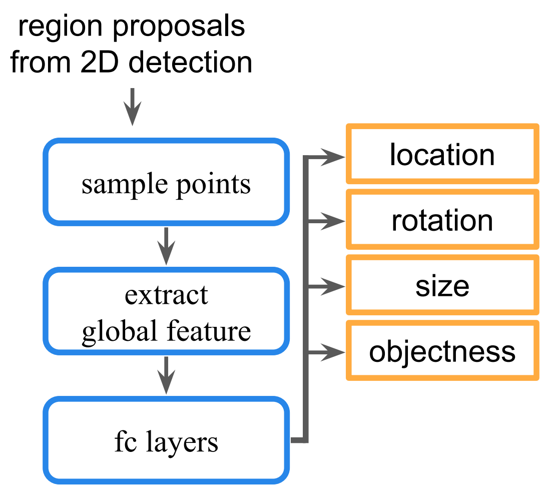


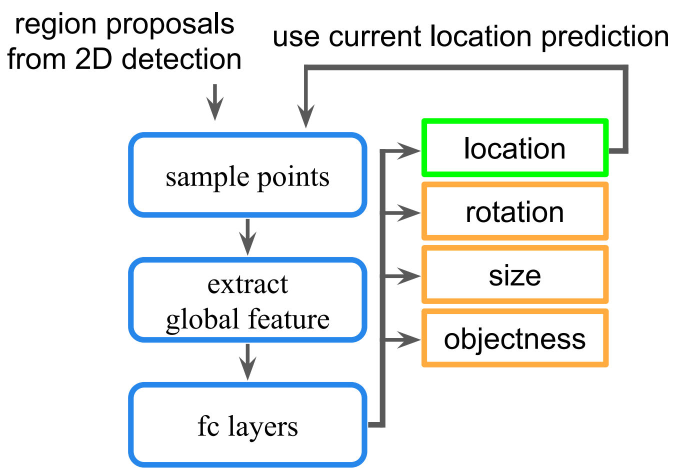
Figure 2.8: The effect of spatial scattering parameter s and objectness threshold

ification immediately improves the performance to 59.9% with an increase of 5.6% from (a).

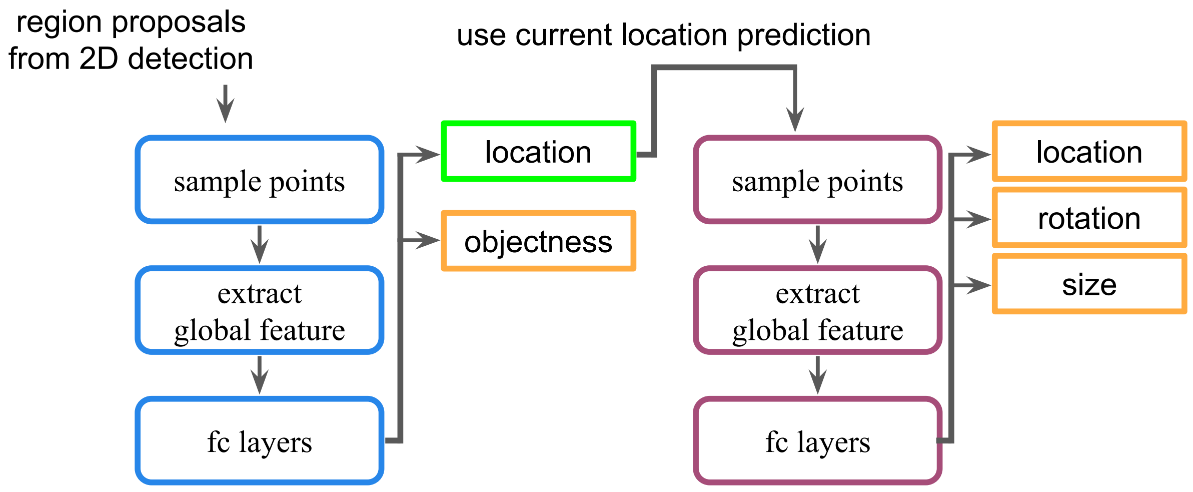
This result inspires us to build our final model, RoarNet_3D in Figure 2.9(c) that specializes each detection step to a specific task and remove redundant predictions. This modification leads significant performance improvement such that recall is 82.5% and mAP is 74.02%.



(a) single stage 3D detector



(b) run (a) twice



(c) our final model, RoarNet_3D

Figure 2.9: A detection pipeline of several network architectures

Runtime analysis

First, we use ResNet-50 as a backbone network for RoarNet_2D and it takes around 43 ms per frame. Second, RoarNet_3D (RPN) samples points from each region proposal predicted from RoarNet_2D and batch processes sampled points to predict objectness score and location.

Runtime required for this step highly depends on spatial scattering parameter s. For fast detection, we use $s = 0.2$ with 7.3 region proposals per object and it takes 35ms. For submission to KITTI server, we use $s = 0.5$ and it takes 65ms. Third, RoarNet_3D (BRN) samples points from region proposals with high objectness score and batch processes to predict 3D bounding box. It takes only 20ms as there are less number of region proposals to be considered.

Average runtime per frame takes around 100–130ms, which may vary according to choice of network structure and hyper parameters. Concurrently to this work, we develop new technique to unify RoarNet_3D (RPN) and RoarNet_3D (BRN) into one for more efficient computation and we will leave this for future work.

Weakness of RoarNet

RoarNet_2D sets region proposals in 3D space by using pose and bounding box predicted from 2D images. This imposes weakness on detection pipeline when RoarNet_2D misses to detect object from 2D images and/or pose prediction has more error than spatial scattering method can compensate.

First, RoarNet_2D occasionally misses to detect object as shown in figure A. In this case, no region proposals will be generated in 3D space.

Second, RoarNet_2D may generate false positive 2D bounding box predictions which results in setting unnecessary region proposals in 3D space.

Third, geometric agreement search misbehaves when objects are located in the boundary of images as 2D bounding box can be clipped out and only small part of full 2D bounding box is considered in calculating region proposals.

Forth, RoarNet_2D sometimes make more error in pose prediction, especially for heading angle prediction, than spatial scattering method can compensate. In an extreme case, difference between ground truth heading angle and predicted heading angle is 90' which leads to generating large number of region proposals in wrong locations, as shown in Figure D.

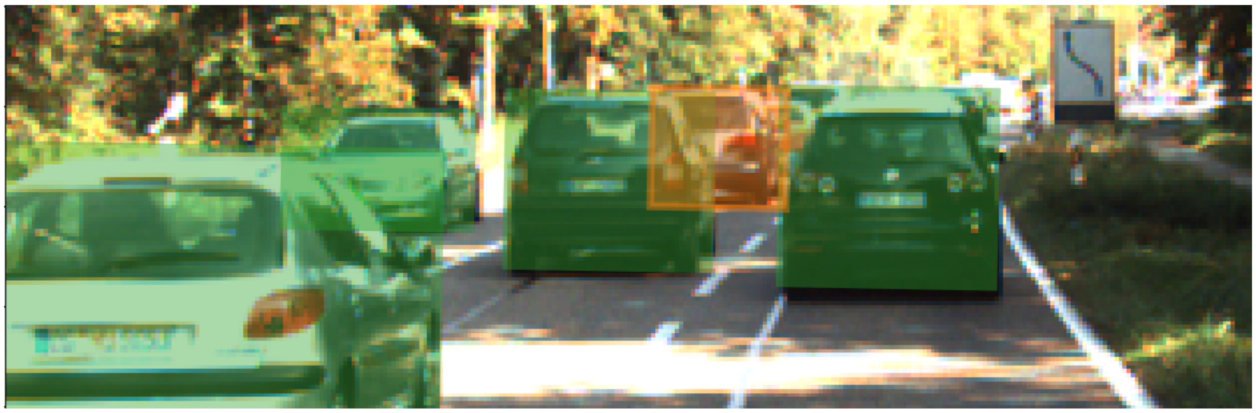
We think this implies a future research direction to improve detection performance.

2.11 Discussion and future research direction

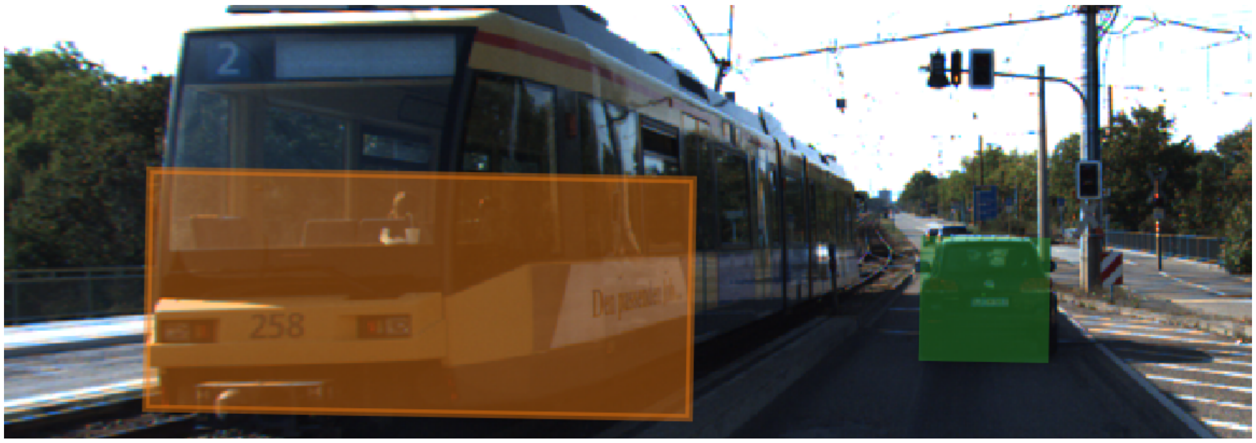
We have proposed RoarNet, a new approach for 3D object detection from an 2D image and 3D Lidar point clouds. RoarNet refines search space recursively at each step in order to make training and prediction efficient.

We first estimate 3D poses from a monocular input image, and derives multiple geometrically feasible candidates nearby the initial estimates. We adopt a two-stage object detection framework to further refine search space effectively from 3D point clouds. Our model shows superior performance to state-of-the-art methods in KITTI, a 3D object detection benchmark.

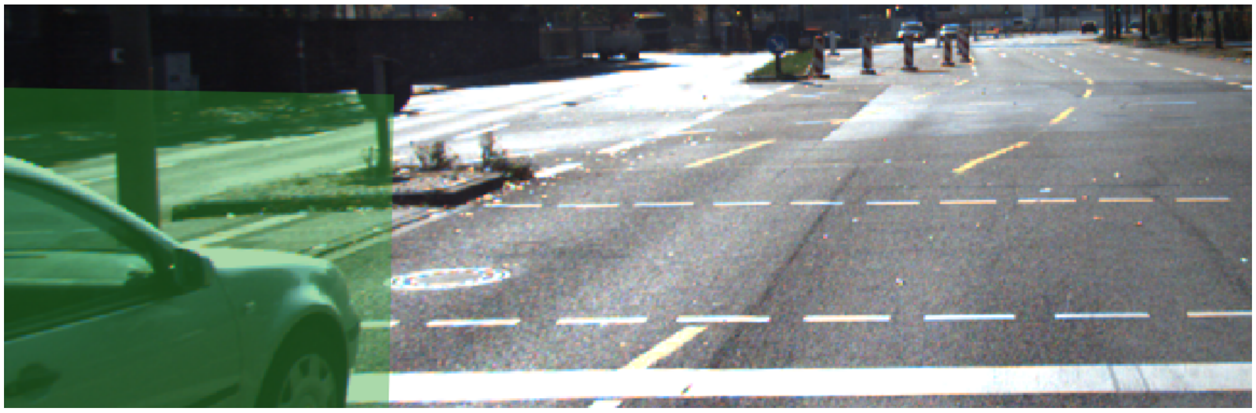
RoarNet outperforms even in the setting where Lidar and camera are not time synchronized, which is practically important results in order to extend current single frame based detection into video frame based detection in the future research.



(a) Object missed from 2D camera detector. Yellow box is false negative which is missed from the 2D detector.



(b) Tram is classified as vehicle. Yelow box is false positive detection from the 2D detector



(c) 2D bounding box is located at the boarder of image. This makes it difficult to apply geometrical agreement search.

Figure 2.10: Example of failure cases from the RoarNet.

Chapter 3

Efficient LiDAR 3D Object Detection Algorithm

3.1 Introduction

In this chapter, we discuss an efficient method for LiDAR-based 3D object detection system. We present an **epBRM** (**e**nd-**p**oint **B**ox **R**egression **M**odule), which is designed for predicting precise 3D bounding boxes using LiDAR 3D point clouds. The proposed epBRM is built with sequence of small networks and is computationally lightweight.

Our approach can improve a 3D object detection performance by predicting more precise 3D bounding box coordinates. Surprisingly, it takes less than 3 hours for training and imposes small overhead on inference time, less than 12 ms latency for up-to 20 objects.

Instead of building unified structure for whole detection pipeline, this work focuses on predicting 3D bounding box more precisely.

Separating a whole detection pipeline into a localization module and a box regression module brings us two benefits: 1) we can build a network specifically designed for box regression task which enhances efficiency of both training and inference, and 2) the design of 3D object detection system becomes more flexible.

We conduct in-depth analysis of the effect of various spatial transformation mechanisms applied on LiDAR 3D point clouds and the effect of localization module on 3D object detection performance.

We evaluate our approach on KITTI dataset[15], a standard 3D object detection benchmark for autonomous vehicles. The results show that the proposed epBRM enhances the overlaps between ground truth bounding boxes and detected bounding boxes, and improves 3D object detection. We also evaluate the proposed method in KITTI test server, and the results show the approach outperforms current state-of-the-art approaches.

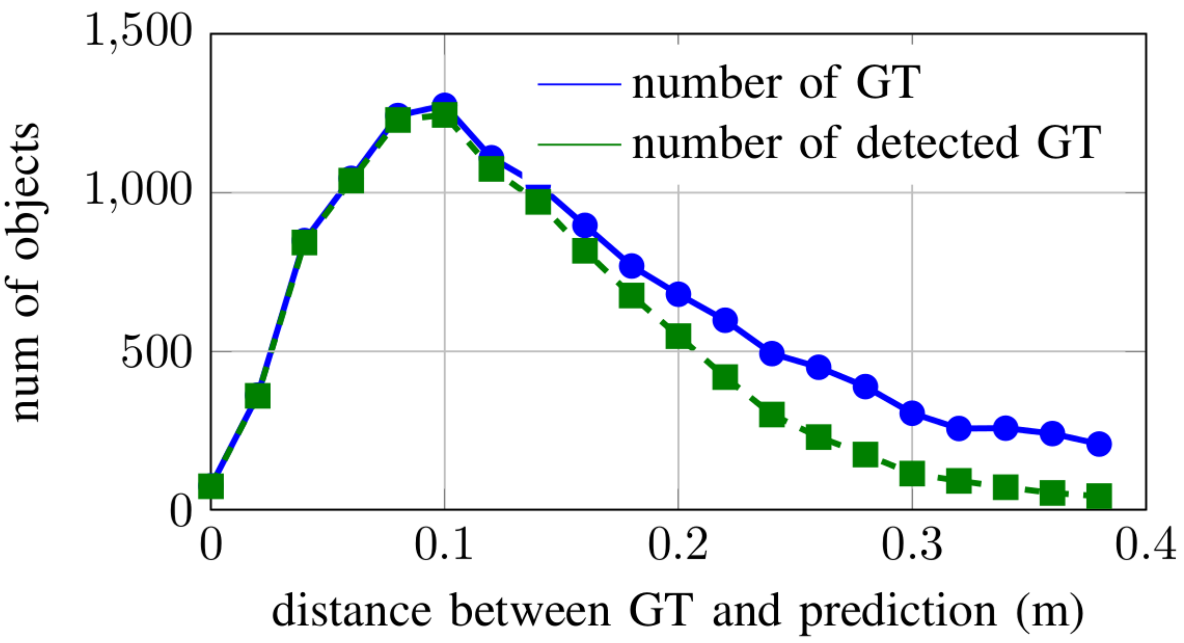


Figure 3.1: Many of predictions located close to ground truth objects(: GT) have 3D overlap lower than 70%. 'Detected' means that overlap between GT and prediction is greater than 70%.

Contribution

The main goal of this work is to improve quality of bounding box predictions in order to increase recall of 3D object detection from LiDAR 3D point clouds. One interesting observation on recent 3D object detection systems is that even though predicted objects are located very close to ground truth objects, many of those predictions are classified as false positive, meaning that overlap between ground truth object and prediction is lower than a specific requirement.(see Figure 3.1)

This is mainly due to the characteristics of the domain where those detection systems are applied to. In autonomous driving, the size of search space is usually 80m(side) X 70m(forward) from ego vehicle, which is very huge compared to the size of objects, such as car, pedestrian and cyclist.

Moreover, the number of object, which can be used as training samples, is also limited. Thus, the majority of search space is occupied by non-object area. During training process, the network is mostly trained to classify object/non-object areas(localization task) and less effort is made for predicting precise 3D bounding boxes(box regression task).

There have been several technical attempts to overcome such difficulty by tuning a weight

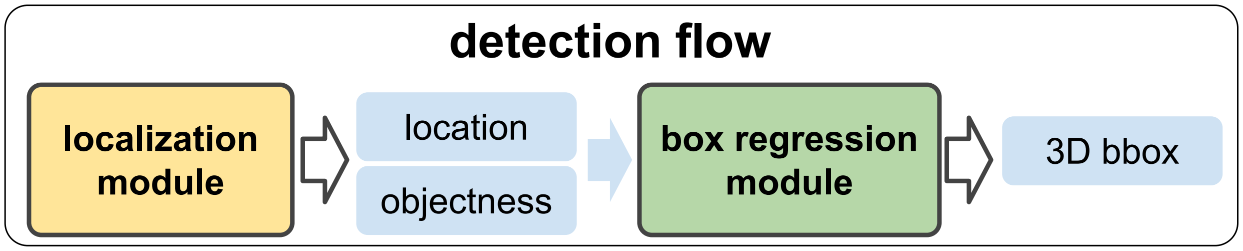


Figure 3.2: Decomposing a detection pipeline into localization module and box regression module allows us a flexibility in building detection system as well as enhanced efficiency in training box regression task.

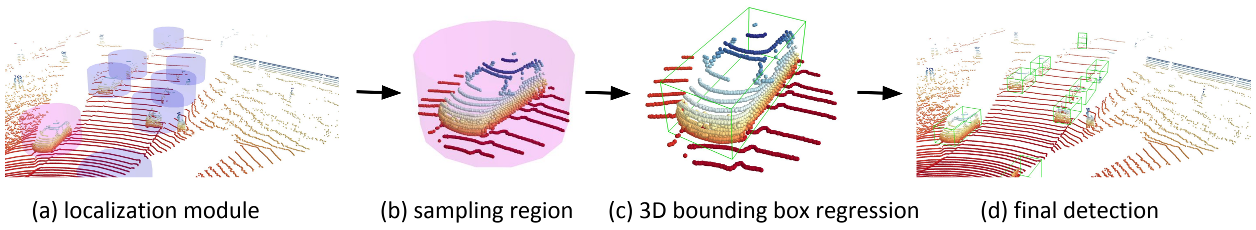


Figure 3.3: 3D bounding box regression pipeline. (a) The localization module predicts the location of objects. (b) samples point clouds at each predicted location, (c) and predicts 3D bounding box which fits to the object. (d) shows final detection results.

of each task and/or adopting focal loss to dynamically control weight of each task[33], but those approaches have not been very successful.

In this work, instead of trying to find the best hyper-parameter by tuning process, we turn our focus on more fundamental question. If having an unified network structure for both localization task and box regression task degrades quality of detection, why not decomposing those tasks into two networks and train them independently?

The main idea behind our proposed detection system is to decompose the whole detection pipeline into a localization task and a box regression task and train each task independently(Figure 3.2). This approach has several advantages over having an unified structure for both tasks.

First, a box regression network can be trained more efficiently, 1) with a network specifically designed for regression task, and 2) using training samples containing information about 3D bounding boxes only. In this work, we design new network structure specially designed for box regression task(Section 3.3). Moreover, the proposed box regression network no longer needs to be trained with samples containing no information about objects(Section 3.5).

Second, a design of 3D object detection system becomes more flexible, which means that any source of information about the location of objects can be used for localization purpose. For example, not only deep neural network based approaches, but any sensor such as Radar or camera can be used directly for localization purpose, or multiple approaches can be combined together to generate more reliable location prediction.

While we mainly evaluate the proposed method using previous works in 3D object detection, which use deep neural network as their main structure, we expect other sources can provide valuable information about the location of objects. And we believe this is more practical in actual driving environment.

Despite it is more beneficial to decompose a whole detection pipeline into localization task and box regression task and train them independently, most 3D object detection systems rely on building an unified network structure for whole detection pipeline. The main reason behind building unified structure is to improve the detection speed because transitioning between two independent networks slows down the detection speed.[**shin2018roarnet**, 38]

Therefore, to make the proposed approach more practical, the overhead imposed by adopting box regression network should be little for real time object detection. To meet such requirement, we design the epBRM with a small size network, which imposes only 12ms overhead for up to 20 objects(Section 3.6).

Moreover, by designing a small sized network, the network is capable of using more training samples in a batch and this leads to stable convergence of network parameters during training. Most of current detection systems which use whole 3D scene as input to the network are capable of using only batch of 2 or 3 training samples per GPU due to lack of memory. The proposed approach is capable of using batch of 512 or 1024 training samples together during training.

We evaluate the proposed method by applying it to the result of current 3D object detection systems, AVOD(FPN)[24], F-PointNet[38] and PointPillars[26], each of which is used as localization module in our detection pipeline.

In this work, a localization module is responsible for predicting the location of object and its corresponding confidence score. And epBRM is responsible for predicting all coordinates required for a 3D bounding box, including location, rotation and size of object.

The evaluation result shows that the proposed approach improves the 3D object detection performance immediately after 20k of training iteration, which takes only 80 minutes using Titan X GPU(not pascal) and i7-6700k CPU.

3.2 Related work

In this section, we review recent works on 3D object detection system from point clouds and image.

Bird's eye view based approaches

In early works in 3D object detection system, input feature is represented by projecting LiDAR 3D point clouds onto 2D planes and used CNN-based detection network for prediction[34, 51, 67]. A core problem shared by these approaches is a loss of information by projection. Thus, these methods showed low performance in detecting small objects such as pedestrian and cyclist.

Sensor fusion based approaches

The density of LiDAR 3D point clouds is sparse when the object is far from the LiDAR. Several sensor fusion based approaches have been proposed to utilize 2D RGB images that can provide higher resolution than LiDAR[7, 24, 31, 12, 17, 14, 66, 62, 32].

These methods concatenate features extracted from 2D images and LiDAR point clouds to generate input feature for the region proposal network. These approaches improved detection performance for small objects, but they required additional task for synchronization and calibration between multi-sensors.

3D LiDAR point clouds based approaches

One approach in 3D object detection system is to convert Lidar 3D point clouds into voxels and extracts voxelwise features for predicting 3D bounding boxes[73, 47, 27, 18, 60, 69, 61, 52, 8, 48]. Several works proposed feature representation based on 3D CNN, but they required high computation[63, 29].

Another line of work in 3D object detection systems can be categorized between sensor fusion based approach and 3D based approach, which is to use 2D object detection results as region proposals and apply PointNet[39, 40] to predict 3D bounding boxes[**shin2018roarnet**, 38].

One drawback behind these approaches is a slow inference speed due to the need for two sequentially connected pipelines, one for 2D image detection and the other one for 3D point clouds detection.

3.3 3D Bounding Box Regression Module(epBRM)

The main goal of our model is to improve the quality of bounding box regression task in order to increase the recall of detection.

Most of recent object detectors are composed of two main tasks: localization task and box regression task. In localization step, a detector predicts location of objects in a rough manner, in a form of objectness score. Then, in regression task, the detector precisely regresses bounding box that fits to the object.

Our observation on various 3D object detectors recently developed for autonomous vehicles is that the performance of box regression task is seemingly worse than the performance of localization task. There can be multiple reasons explaining such phenomenon, but one reasonable explanation would be that the number of objects and/or the size of object is very small compared to the huge search space generally considered for autonomous driving.

Role and goal of epBRM

In this section, we define a role of epBRM as a component of 3D object detection pipeline. Since our focus is to improve a performance of box regression task, we utilize several previous works in 3D object detection system for localization purpose, by using only location prediction and ignoring all other predictions such as size and rotation of objects.

Note that the main purpose of epBRM is not discovering objects that are missed from localization module, but predicting precise 3D bounding box coordinates by using point clouds sampled around the location predicted from the localization module. Therefore, we need to set a target distance($dist_{\text{bound}}$), which represents the maximum distance between ground truth object and predicted location of object from the localization module that we are aiming for further refining a 3D bounding box. For objects located out of $dist_{\text{bound}}$ from the ground truth object, we do not aim for such detections to be further regressed.

There are two main factors for deciding a value of $dist_{\text{bound}}$: 1) a performance of localization module, and 2) a representation ability of epBRM.

First, the value of $dist_{\text{bound}}$ is in inverse relationship with the precision of localization module. If localization module is capable of predicting precise location of objects, then epBRM only needs to work well for samples with small error. In this case, we assign small value for $dist_{\text{bound}}$. Opposite to such case, if the localization module performs poorly, then epBRM should consider samples with large location prediction error and we set a large value of $dist_{\text{bound}}$.

Second, the value of $dist_{\text{bound}}$ is also closely related to a representation ability of network used for epBRM. If we increase the value of $dist_{\text{bound}}$, it also increases the complexity of task that epBRM is responsible for. If we constrain a representation ability of epBRM aiming for computational efficiency, the value of $dist_{\text{bound}}$ should also be decreased according to that.

In Section 3.5, we describe how we generate training samples for each value of $dist_{\text{bound}}$. In Section 3.8, we evaluate the effect of $dist_{\text{bound}}$ on various localization modules and its effect on detection performance.

epBRM with spatial transformation mechanism

In this section, we explain reasons for adopting spatial transformation mechanism. Aiming better computational efficiency and real time inference, we restrict the size of network for epBRM at the cost of representation ability of network. This arises a need for additional method to simplify box regression task in order for a network with low representation ability be capable of predicting precise 3D bounding boxes.

Our approach is closely related to RoI pooling and anchor structure[44, 16], which are core components in most of two stage object detectors. The main idea behind RoI pooling and anchor structure is to extract features at each pre-defined location in a similar form in order to gradually simplify the task. Unlike 2D RGB image or voxel structure which have dense representation of input feature, we directly use sparse representation of unordered set of point clouds, which has a form of $\mathbb{R}^{\mathrm{Nx3}}$ for N point clouds with x,y,z coordinates.

To avoid voxelizing[73] and/or grouping 3D point clouds[40] which require extra processing time, we insert spatial transformation mechanism[22] into epBRM to get similar effect with pooling based approaches. In essential, our approach first transforms input point clouds into a similar form each other and then predicts 3D bounding boxes. After that, we inverse transform on predicted 3D bounding box to the original coordinate before transformation applied to input point clouds.

We are aiming to predict 3D bounding box in a original coordinate unaffected by transformation mechanism applied to input point clouds. For example, assume that we first translate input point clouds by (x_0, y_0, z_0) and a location of the object after translation is predicted by (x, y, z). Then, the location prediction of the object in the original coordinate should be adjusted as $(x\text{-}x_0, y\text{-}y_0, z\text{-}z_0)$. Therefore, for each transformation mechanism, its corresponding inverse transformation mechanism should be defined to get prediction at the original coordinate. If multiple spatial transformation mechanisms are sequentially applied, then their corresponding inverse transformation mechanisms should be applied to the final prediction in reverse order.

Furthermore, as we predict 3D bounding box that fits to the given object, we focus on transformation mechanisms that preserve the structure of rectangular cuboid.

In this work, we are focusing on four spatial transformation mechanisms, translation, rotation, scaling, and centering. They take the input point clouds $P \in \mathbb{R}^{\mathrm{Nx3}}$ with N point clouds, and 3 dimensions for x, y, z coordinates and output θ, which is the transformation parameter to be applied to P.

Translation and Centering The output of transformation mechanism is defined by (t_x, t_y, t_z) which represents the translation of P by x_0, y_0, z_0 as following equations.

$$\begin{aligned} x_0 &= 2 * (\sigma(t_x) \text{ - } 0.5) * T_x, \\ y_0 &= 2 * (\sigma(t_y) \text{ - } 0.5) * T_y, \\ z_0 &= 2 * (\sigma(t_z) \text{ - } 0.5) * T_z \end{aligned} \tag{3.1}$$

where $\sigma(\cdot)$ is sigmoid activation function. This has a same form with Equation (3.4) except for the hyper parameters, (T_x, T_y, T_z) which constrain the maximum distance of translation. In this work, we set (T_x, T_y, T_z) to the same values with $dist_{\text{bound}}$.

The difference between translation mechanism and centering mechanism is at whether there exists target transformation parameter to be used during training. The spatial transformation mechanisms introduced in [22] are learned without having target transformation parameters. For those transformation mechanisms, the parameters are learned to extract most informative feature for final prediction. Unlike such mechanisms, centering mechanism aims to directly predict the center of object and relocate the object to the origin.

Rotation The output of rotation mechanism is defined by t_r which represents a clockwise rotation of P around z axis by rot_z as following equation.

$$\text{rot}_z = 2 * (\sigma(t_r) \text{ - } 0.5) * T_r \tag{3.2}$$

We set a value of T_r as $\pi/4$ to avoid excessive rotation of input point clouds.

Scaling The output of scaling mechanism is defined by $(t_{s,xy}, t_{s,z})$ which represent the scaling of P on ground plane, scale_{xy} and on z axis, scale_z[1] as following equation.

$$\begin{aligned} \text{scale}_{xy} &= \text{pow}(T_{s,xy}, 2 * (\sigma(t_{s,xy}) \text{ - } 0.5)), \\ \text{scale}_z &= \text{pow}(T_{s,z}, 2 * (\sigma(t_{s,z}) \text{ - } 0.5)) \end{aligned} \tag{3.3}$$

We set a value of $T_{s,xy}, T_{s,z}$ as 2.0, thus constrain scaling factors within $[\frac{1}{2}, 2]$ to avoid distorting input point clouds.

Differentiable point sampler

Another important component of spatial transformer network[22] is differentiable image sampling mechanism which allows backpropagation of loss through sampler into input features.

To avoid point segmentation task which disconnects input features and output loss[38], we implement point clouds sampler by exploiting a characteristic of unordered set of point clouds such that if one or many elements of a set are repeated, the set remains the same.

One possible approach is to replace all point clouds outside sampling region with any point cloud inside sampling region. However, this requires additional computation and still blocks backpropagation of loss through sampler.

In this work, we simply relocate all point clouds outside sampling region into the origin, thus those point clouds become repeated elements in set of point clouds. This operation is simple and fast, but may create new point cloud at the origin. We assume that the single element at the origin has negligible effect on final prediction.

3D bounding box regression

After transformation applied to input point clouds, epBRM finally predicts coordinates for 3D bounding box, such as location, rotation and size of objects. We use confidence score predicted from each localization module for scoring our predicted 3D bounding boxes.

Location is predicted by 3 coordinates (t_x, t_y, t_z) for x, y, z direction relative to the origin as following equation:

$$\begin{aligned} x &= 2 * (\sigma(t_x) \text{ - } 0.5) * d_x, \\ y &= 2 * (\sigma(t_y) \text{ - } 0.5) * d_y, \\ z &= 2 * (\sigma(t_z) \text{ - } 0.5) * d_z \end{aligned} \tag{3.4}$$

[1]perpendicular to the ground plane

The (d_x, d_y, d_z) represents a maximum offset of center of an object from the origin. Aiming finer prediction of location, we set values for (d_x, d_y, d_z) as follows:

$$[d_x, d_y, d_z]^{\mathrm{T}} = 0.5 * [T_x, T_y, T_z]^{\mathrm{T}} \tag{3.5}$$

Rotation is predicted based on hybrid formulation of classification and regression methods. We equally divide [0', 180') to N_R bins and the rotation angle is predicted by 2*N_R coordinates $(t_{r\_\mathrm{cls}(i)}, t_{r\_\mathrm{reg}(i)})_{i=1}^{N_R}$. We use $N_R = 9$, and each bin is responsible for 20.0' of rotation angle.

Size is predicted by 3 coordinates, (t_h, t_w, t_l), for height(h), width(w), and length(l). We set anchor of size coordinates, (h_a, w_a, l_a). We use (1.50m, 1.57m, 3.33m) for car, (1.73m, 0.6m, 0.8m) for pedestrian and (1.73m, 0.6m, 1.76m) for cyclist. Then, the size of object is predicted by:

$$h = h_a e^{t_h}, \quad w = w_a e^{t_w}, \quad l = l_a e^{t_l} \tag{3.6}$$

3.4 Network structure

In this section, we describe a network structure for both transformation mechanism and 3D bounding box regression task. We design epBRM as sequence of multiple PointNet by using it as building block. Figure 3.4 shows a structure of the building block based on PointNet . This structure is shared by transformation mechanism and 3D bounding box regression task except for the number of output at the last layer.

In case of transformation mechanisms, we transform point clouds using output of network, which is a transformation parameter to be applied to input point clouds(Section 3.3) and sample point clouds inside sampling region(Section 3.3). Since we simply relocate point clouds outside of sampling region into the origin, we again get n point clouds with 3 coordinates with some repeated point clouds. During training, we use a half number of point clouds used for testing as Table 3.1.

class	training	testing
car	256	512
pedestrian & cyclist	128	256

Table 3.1: number of point clouds sampled for training and testing.

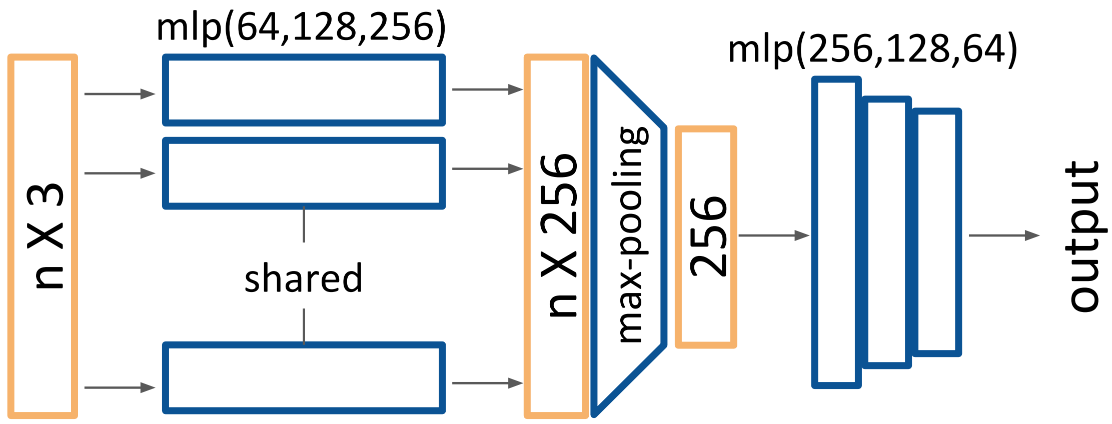


Figure 3.4: the modular network structure used as a building block of epBRM. The same structure is used for both transformation mechanism and the 3D bounding box regression task.

3.5 Dataset: training samples

In this section, we describe how we generate training samples for epBRM including choice of sampling region shape and data augmentation methods applied to input point clouds.

Unlike CNN based approaches which require rectangular shape and dense representation of input feature, we directly use sparse representation of LiDAR 3D point clouds as input to the network. Therefore we have more flexibility in selecting a shape of sampling region.

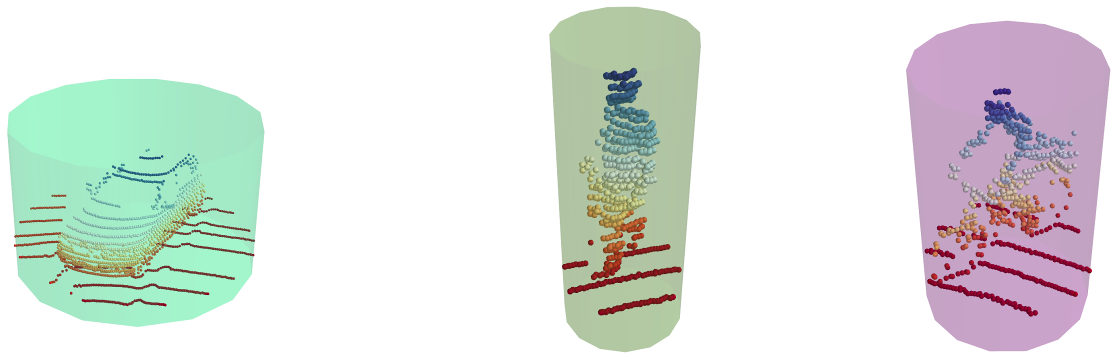
We aim for setting sampling region as tight as possible around object in order to exclude redundant point clouds that are not belong to the object. Also, as we only have location prediction from localization module, we need rotation-invariant sampling region. To satisfy both requirements, we use cylinder shaped sampling region in this work.

Considering a prediction error from localization module, we set size of sampling region slightly larger than ordinary size of objects. We sample point clouds inside a cylinder defined by radius r and min/max value of point clouds (z_{min}, z_{max}) along z axis. (Table 3.2)

class	radius(r)	min height(z_{min})	max height(z_{max})
Car	2.4	-0.5	2.5
Pedestrian	0.35	-0.5	2.5
Cyclist	0.8	-0.5	2.5

Table 3.2: the size of sampling region for each class

Now we describe data augmentation methods for generating training samples. Assume



(a) Car (b) Pedestrian (c) Cyclist

Figure 3.5: shape of sampling region for each class

that the 3D bounding box coordinates of the ground truth object are given by $\mathrm{loc}_{\mathrm{gt}} = (\mathrm{loc}_{\mathrm{gt,x}}, \mathrm{loc}_{\mathrm{gt,y}}, \mathrm{loc}_{\mathrm{gt,z}})$, $\mathrm{size}_{\mathrm{gt}} = (\mathrm{h}_{\mathrm{gt}}, \mathrm{w}_{\mathrm{gt}}, \mathrm{l}_{\mathrm{gt}})$ and $\mathrm{rot}_{\mathrm{gt}}$.

First, we sample all point clouds inside sampling region centered at $\mathrm{loc}_{\mathrm{gt}}$. Then, we translate sampled point clouds by subtracting $\mathrm{loc}_{\mathrm{gt}}$ from each point and rotate them by -$\mathrm{rot}_{\mathrm{gt}}$ to align the object along y axis.

We change the size of object by multiplying s_x, s_y, s_z for each x, y, z axis of point clouds independently. s_x, s_y, s_z are independently sampled from uniform distribution on the interval [0.9, 1.1]. Then, the target value of size of object now becomes

$$\mathrm{size}_{\mathrm{target}} \leftarrow (\mathrm{h}_{\mathrm{gt}} * s_z, \mathrm{w}_{\mathrm{gt}} * s_y, \mathrm{l}_{\mathrm{gt}} * s_x) \tag{3.7}$$

After applying size augmentation, we rotate it back to original heading angle plus small angle r_z randomly sampled from uniform distribution on the interval $[-pi/8, pi/8]$. Then, the target value of rotation of object now becomes:

$$\mathrm{rot}_{\mathrm{target}} \leftarrow \mathrm{rot}_{\mathrm{gt}} + r_z \tag{3.8}$$

Finally, we sample $\mathrm{loc}_{\mathrm{x}}, \mathrm{loc}_{\mathrm{y}}, \mathrm{loc}_{\mathrm{z}}$ independently from uniform distribution on the interval [-$dist_{\mathrm{bound}}, dist_{\mathrm{bound}}$] according to the target of our box regression model.

$$\mathrm{loc}_{\mathrm{target}} \leftarrow (\mathrm{loc}_{\mathrm{x}}, \mathrm{loc}_{\mathrm{y}}, \mathrm{loc}_{\mathrm{z}}) \tag{3.9}$$

3.6 Loss, training and inference

During training each network, we optimize the following multi-task loss:

$$\begin{aligned} Loss = L_{\mathrm{loc}} + L_{\mathrm{rot\text{-}cls}} + \mathbb{1}^{\mathrm{rot\text{-}cls}}[L_{\mathrm{rot\text{-}reg}}] + L_{\mathrm{size}} && (3.10) \\ + L_{\mathrm{loc\_center}} && (3.11) \end{aligned}$$

If we use transformation mechanism such as translation, rotation and scaling, then we use Equation (3.10) as a loss function. If we adopt centering mechanism in epBRM, then we use Equation (3.11) as a loss function.

$L_{\text{loc\_center}}$, L_{loc}, $L_{\text{rot-reg}}$, and L_{size} are regression loss for intermediate center location prediction, location, rotation and size of bounding box regression, which are represented as huber loss. $L_{\text{rot-cls}}$ is classification loss for rotation, which is represented as cross-entropy loss.

For training a network, we use samples generated at Section 3.5. We use batch of 512 samples with fixed learning rate of 5e-4. We don't apply non-maximum suppression on the predictions.

Up to 20 objects, it takes 6.5ms for sampling point clouds at each location which is predicted from localization module. When we adopt one transformation mechanism and predict 3D bounding box, it takes 5.5ms for network inference.

3.7 Experiment setup

Dataset

We use KITTI dataset, the 3D object detection benchmark, to evaluate our approach. It provides synchronized 2D RGB images and 3D LiDAR point clouds, carefully calibrated with annotations on car, pedestrian, and cyclist class.

We mainly evaluate our method on the car class which has the most training samples. For pedestrian and cyclist, we discuss in Section 3.8.

To evaluate our method, we split whole training set into train set of 3,717 frames and val set of 3,769 frames frames. Frames in train set and frames in val set are extracted from different video clips. 3D object detection performance is evaluated at 0.7 IoU threshold.

Localization module

To evaluate epBRM in 3D object detection benchmark, we utilize previous 3D object detection systems as localization modules. We select three 3D object detection systems publicly available as open source: F-PointNet, AVOD(FPN), and PointPillars. Shown in Table 3.3, each method is different in use of sensors and network structure.

Our model uses only the location and score predictions from those detectors and ignores size and rotation predictions.

While our refinement model is agnostic to any type of 3D object detectors, the refinement performance can be different at each detector. We discuss about this at Section 3.8

Therefore, we evaluate our method by using detection results from AVOD(FPN)[24], F-PointNet(v1)[38], and PointPillars[26] which are introduced in earlier section.

We follow official instruction provided by authors of AVOD-FPN and PointPillars to train each detection network. We use pre-trained model for F-PointNet(v1) provided from author[cite]. We compare our result with all three classes: car, pedestrian and cyclist.

method	AVOD(FPN)[24]	F-PointNet[38]	PointPillars[26]
sensors	LiDAR, camera	LiDAR, camera	LiDAR only
region proposals	fused LiDAR and image feature	pretrained 2D detector	CNN based

Table 3.3: Comparison of 3D object detection systems. Explanations on AVOD(FPN) and F-PointNet are from [13]

Evaluation metric

To evaluate our approach, we mainly focus on whether epBRM improves a quality of 3D bounding box prediction. As well as recall and mAP of detection which are general evaluation metrics for 3D object detection benchmark, we also measure **ratio** of detected ground truth objects defined as follows:

$$\text{ratio} := \frac{\text{num. of detected GT objects}}{\text{num. of all GT objects}} \tag{3.12}$$

where the term 'GT' represents ground truth and 'detected' means that the overlap between ground truth 3D bounding box and predicted 3D bounding box is greater than 70%.

This measurement considers all ground truth object regardless of its difficulty level and visible size of object in RGB image plane, thus provides more reliable measurement.

3.8 Experiment results

Comparisons of transformation mechanism

As motivating experiment for our approach, we first evaluate various spatial transformation mechanisms applied to input point clouds and their effect on 3D bounding box regression task. In this experiment, we utilize PointPillars[26] as a localization module.

We train epBRM with each transformation mechanism for 20k iteration and reports the evaluation results.

Table 3.4 shows the evaluation results of each transformation mechanism adopted in epBRM as well as no transformation mechanism applied to input point clouds. We also report performance of baseline 3D object detector, PointPillars[26] which is used as our localization module.

Table 3.4 indicates that among the various transformation mechanisms applied, the centering mechanism is the most effective in improving 3D object detection performance.

Figure 3.6 shows the validation result of centering mechanism at every 5k until 55k iterations. Note that epBRM immediately surpasses the baseline detector after 10k of training. It takes approximately 40 minutes for each 10k iteration of training.

transformation mechanism	recall(†)	mAP(†)	ratio(‡)
none	85.0	75.52	73.23
translation	85.0	76.65	75.30
center	85.0	77.28	76.62
translation+rotation	82.5	73.73	72.97
center+rotation	85.0	75.90	75.45
center+scale	85.0	76.95	75.34
PointPillars	82.5	76.29	73.17

Table 3.4: a comparison of transformation mechanisms. The recall and mAP are evaluated in moderate level(†). The ratio is evaluated for all difficulty levels(‡) following Equation (3.12).

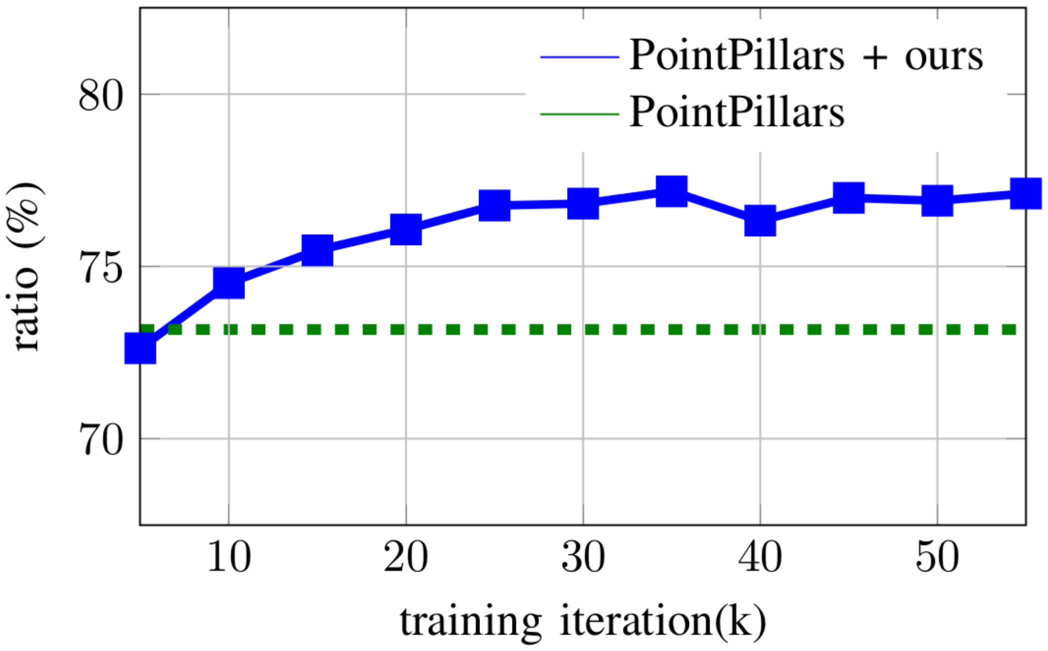


Figure 3.6: Training progress of epBRM. Evaluation of car class in KITTI val set.

Effect of $dist_{\text{bound}}$

In this experiment, we compare effect of $dist_{\text{bound}}$ on several localization modules. The purpose of this experiment is to find optimal value of $dist_{\text{bound}}$ that maximizes the performance of epBRM when epBRM is applied to different localization modules. We again train epBRM with centering mechanism for 20k iteration.

Figure 3.7 visualizes the evaluation results when epBRM is applied to each localization module at different $dist_{\text{bound}}$. For PointPillars and AVOD(FPN), the value of ra-

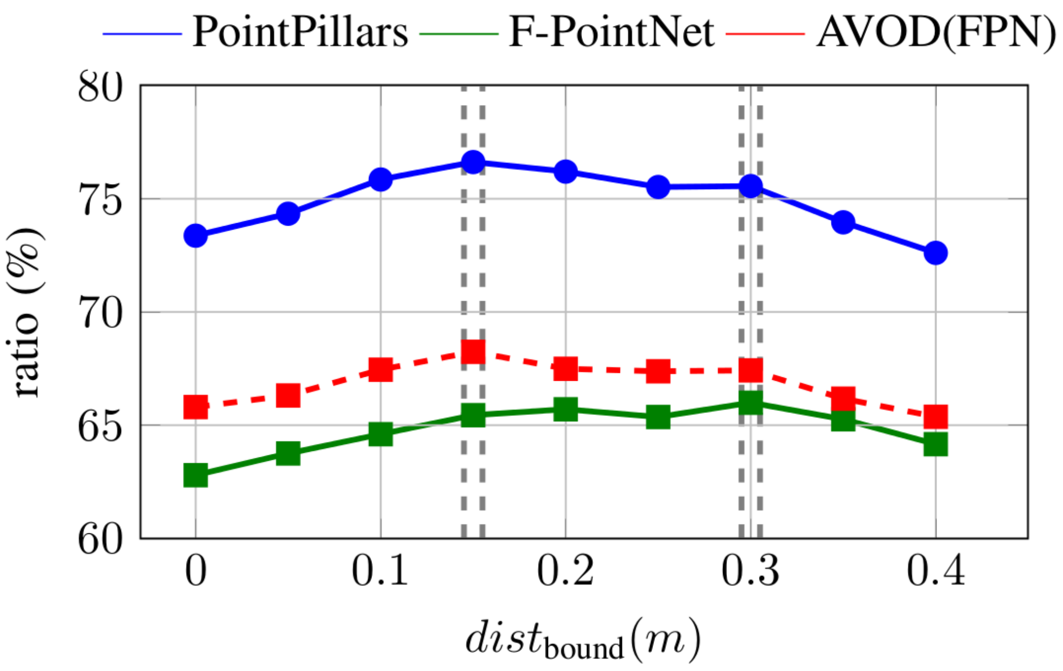


Figure 3.7: Effect of $dist_{\text{bound}}$ on each localization module.

tio(Equation (3.12)) peaks when $dist_{\text{bound}}$ is 0.15 and then gradually decreases after that. For F-PointNet, the value of ratio peaks at $dist_{\text{bound}}$ of 0.30 and then starts to decrease.

Table 3.5 reports the performance gains when epBRM is applied to each localization module at optimal $dist_{\text{bound}}$ found by this experiment.

Note that the performance improvement gained by applying epBRM is different at each localization module. For example, epBRM greatly improves the performance of AVOD(FPN) while less improvement is observed from F-PointNet. This result infers that 3D bounding box prediction function of AVOD(FPN) is comparably worse than the F-PointNet.

Smaller objects: pedestrian and cyclist

Our approach is also applicable for predicting precise 3D bounding boxes for smaller objects such as pedestrian and cyclist. Again, we train epBRM using fixed value of $dist_{\text{bound}}$ as 0.15 for 20k iteration and evaluate the performance by applying it to each localization module. Here, we omit the process of finding optimal value of $dist_{\text{bound}}$.

Evaluation using KITTI test set

We also evaluate the proposed approach by using KITTI *test* set by submitting detection result to KITTI test server. Among several localization modules we evaluate so far, Point-

methods	car				pedestrian				cyclist			
	dist (m)	recall (%)	mAP (%)	ratio (%)	dist (m)	recall (%)	mAP (%)	ratio (%)	dist (m)	recall (%)	mAP (%)	ratio (%)
AVOD(FPN)	-	77.5	68.50	63.18	-	52.5	39.78	41.36	-	55.0	37.14	45.80
+ ours	0.15	82.5	77.02	68.24	0.15	55.0	41.54	44.04	0.15	57.0	37.61	48.38
F-PointNet	-	82.5	71.36	62.06	-	72.5	55.40	59.12	-	75.0	53.85	62.91
+ ours	0.30	82.5	72.90	65.99	0.15	77.5	63.06	66.84	0.15	77.5	59.92	68.09
PointPillars	-	82.5	76.29	73.17	-	77.5	58.95	69.30	-	77.5	61.22	62.60
+ ours	0.15	85.0	77.28	76.62	0.15	82.5	61.65	73.29	0.15	77.5	61.07	65.00

Table 3.5: Comparisons of 3D object detection performance on car, pedestrian and cyclist class before/after applying epBRM to each localization module. The performances are evaluated at KITTI *val* set. The recall and mAP are evaluated in moderate difficulty level and the ratio is evaluate in all difficulty level.

	easy		moderate		difficult	
car	mAP	recall	mAP	recall	mAP	recall
PointPillars(†)	78.17	90.0	68.71	80.0	65.91	75.0
+ ours	83.95	92.5	75.79	82.5	67.88	77.5
PointPillars(‡)	79.05	90.0	74.99	85.0	68.30	80.0
pedestrian	mAP	recall	mAP	recall	mAP	recall
PointPillars(†)	45.67	62.5	38.65	52.5	36.16	50.0
+ ours	50.38	67.5	43.90	60.0	40.91	57.5
PointPillars(‡)	52.08	67.5	43.53	57.5	41.49	55.0
cyclist	mAP	recall	mAP	recall	mAP	recall
PointPillars(†)	70.92	90.0	55.57	75.0	49.95	65.0
+ ours	70.52	90.0	56.94	75.0	51.70	67.5
PointPillars(‡)	75.78	92.5	59.07	77.5	52.92	67.5

Table 3.6: Evaluation results for car, pedestrian and cyclist at KITTI *test* set. (†) represents reproduced result from our side. (‡) represents performance of PointPillars reported by Lang et al. [26]

Pillars[26] shows the best performance in all categories, thus we apply epBRM to prediction result from PointPillars. We train epBRM for 40k iterations, which takes approximately 160 minutes.

Table 3.6 reports evaluation result on KITTI *test* set before/after epBRM is applied. We also reports original PointPillars result(‡) reported by Lang et al.[26] for reference. Note that epBRM improves the recall of detection in most cases.

3.9 Discussion and future research direction

In this work, instead of building a entire detection pipeline, we mainly focus on a box regression task, one of core components in detection pipeline that aims to predict precise 3D bounding box for objects. This approach brings us several benefits.

First, we can build a small network that specifically designed for predicting 3D bounding box with short inference time.

We can also increase the number of training samples in a batch, which stabilizes training process. As a result, our proposed approach, epBRM, greatly improves 3D object detection performance within short time of training.

To build an efficient bounding box regression pipeline, we adopt transformation mechanisms into epBRM. We evaluated various transformation mechanisms including translation, rotation, scaling and centering. Among them, centering and translation mechanisms improved the detection performance while rotation and scaling mechanisms didn't improve the performance.

For each baseline 3D object detector, which is used as localization module in our work, we explored effect of value of $dist_{\text{bound}}$ on 3D object detection. We observed that the optimal values of $dist_{\text{bound}}$ and performance improvement are different at each localization module. For a 3D object detection system which has good localization performance, but underperforms in box regression task, we can expect overall performance improvement by replacing box regression function by epBRM.

Decomposing a whole detection pipeline into a localization task and a box regression task brings us flexibility of design of detection system while imposing little overhead in inference time. epBRM requires that the localization module provides reliable information about the location of objects.

If the localization module fails to localize objects precisely, then no further attempt to predict 3D bounding boxes will be made by epBRM. This also indicates our future research direction.

Chapter 4

Data augmentation for 3D LiDAR point clouds

4.1 Introduction

In this chapter, we mainly discuss a novel data augmentation method for 3D LiDAR point clouds. By using the proposed data augmentation method and the improved network structure of epBRM_V1 which is discussed in Chapter 3, the performance of 3D object detection is greatly improved compared to epBRM_V1.

The data augmentation method is a core technique for training a deep neural network. The data augmentation refers to a set of simple transformations which can be applied to input features and ground truth labels. By applying several data augmentation methods, the deep neural network can be trained by using more diversified training samples which improves a generalization of the network as a result.

While a number of data augmentation methods for 2D camera images have been actively explored, the data augmentation methods for 3D LiDAR point clouds have been less explored in the literature. The horizontal/vertical flipping, global translation/rotation, point-level jittering, frustum cutout are often used data augmentation methods for 3D LiDAR point clouds.

More recently, researchers are trying to build a sophisticated LiDAR simulator to generate pseudo LiDAR point clouds that can be used for training a deep neural network. However, not only it requires deep understanding behind the physical characteristic of LiDAR sensor and noises, the performance of detection networks trained on newly generated dataset have not been promising.

Instead of building a LiDAR simulator, we develop a data-driven data augmentation method for 3D LiDAR point clouds. The proposed method utilizes the tracklet information which tracks a single object for multiple time stamps. The tracklet information helps to densify a sparse 3D LiDAR point clouds. Then, the proposed approach mixes the dense 3D LiDAR point clouds with the sparse 3D LiDAR point clouds from the original training dataset to generate a new training dataset.

Additionally, we improve the representation ability of epBRM_V1 using skip-connected layers and deeper layers. These two simple modifications, one from a novel data augmentation for 3D LiDAR point clouds and the other one from network structure, greatly improve the detection performance.

4.2 Contribution

The main goal of this work is to improve the detection performance of epBRM_V2 throughout in-depth analysis on localization module and box-regression module. In this work, the proposed approach mainly aims at improving the recall performance of overall detection task while the epBRM_V1 mainly explored the methodology to improve the quality of bounding box regression.

In this work, we further improve the detection performance based on in-depth analysis on both localization module and box regression module. While maintaining the overall detection pipeline similar to epBRM_V1, the proposed approach is based on three major modifications which lead to significant improvement in detection performance: a) in-depth analysis on a network specifically designed for predicting only the location of the objects, b) a wide and deep layers with skip-connected path for better feature representation, c) a novel data augmentation method for 3D LiDAR point clouds based on mixup methodology.

In Section 4.4, we explore the localization performance as standalone module. Instead of using prediction results from the existing detection networks, the proposed approach trains a network specifically designed for localization purpose only. This reduces the complexity of the task that the network is responsible for.

The epBRM_V1 uses the existing detection networks such as AVOD-FPN[24], Frustum PointNet[38] and PointPillars[26] for localization purpose. This approach helps epBRM_V1 to evaluate how much performance can be improved by using box regression network.

In epBRM_V2, the proposed approach builds a localization module which is trained only for predicting location of the objects. This approach enables us to carefully analyze the recall and precision performance of the localization network as standalone module.(Section 4.6)

In Section 4.4, inspired by the recent improvement in structure of detection network, such as StarNet[37] and skip-connected layers[19], the proposed approach builds an epBRM_V2 which has better representation ability for box regression task. This approach improves the recall performance of detection task to state-of-the-art.

In Section 4.6, we develop a novel data augmentation method for 3D LiDAR point clouds based on mixup method. We provide detailed process of creating a new training sample. Our proposed data augmentation method uses a tracklet information which is prevalent in most public datasets for 3D object detection task such as KITTI[15], Waymo[55], nuScenes[2], and ArgoVerse[6], etc.

4.3 Related works

Box regression module for detection pipeline

epBRM_V1

The proposed approach is based on our previous work on epBRM_V1[50]. The main objective of the epBRM_V1 is to improve the detection performance by building a network which is specifically designed for box regression task. The epBRM_V1 is inspired by the analysis(Figure 3.1) which shows that many of predicted bounding boxes which are proximal to the ground truths are regarded as false positive due to their low quality in regressing the coordinates of 3D bounding boxes.

Instead of pursuing training a whole detection network in an end-to-end manner, the epBRM_V1 divides a detection pipeline into localization task and box regression task and focuses only on training a box regression module. It uses location predicted from other 3D detectors such as Frustum-PointNet[38], PointPillars[26] and AVOD-FPN[24] for localization purpose.

In epBRM_V2, which is the improved version of epBRM_V1, we make further improvements based on three approaches: a) instead of using existing detection networks for localization purpose, the proposed approach builds a network specifically designed for localization task, b) deeper and wider network structure with skip-connected layers enables the box regression module to predict the high quality 3D bounding box coordinates, c) we develop a new data augmentation method for 3D LiDAR point clouds which uses mixup methods[72] and tracklet information from the dataset.

StarNet

Another milestone work which focuses on box regression module is StarNet[37]. StarNet also separates the whole detection pipeline into proposal generation process and box regression process. To generate proposal from 3D LiDAR point clous with high recall, StarNet uses sampling based approach which uses the distribution of 3D LiDAR point clouds as sampling parameters.

StarNet compares three sampling based approaches which generate proposals from 3D LiDAR point clouds, a) random uniform sampling, b) fathest point sampling and c) hybrid of two methods. While random uniform sampling method increases the probability of points in a densely populated regions to be selected, farthest point sampling method maximizes the spatial coverage of region proposals.

Unlike to the most modern detection networks which uses deep learned network for proposal generation process, StarNet heavily relies on simple mathematical algorithm for generating proposals. Instead of training a network for localizing objects, StarNet focuses more on box regression module which is based on PointNet.

While epBRM_V1 and StarNet share the similar network architecture, there are two main differences between them, a) deeply connected layers between global features of PointNet, b) mean pooling operation instead of max pooling operation to remove the order of LiDAR point clouds feature. Figure 4.2 shows the differences between two networks.

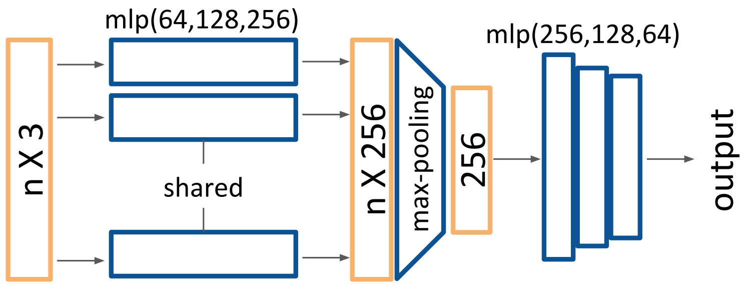


Figure 4.1: Network structure of epBRM_V1.

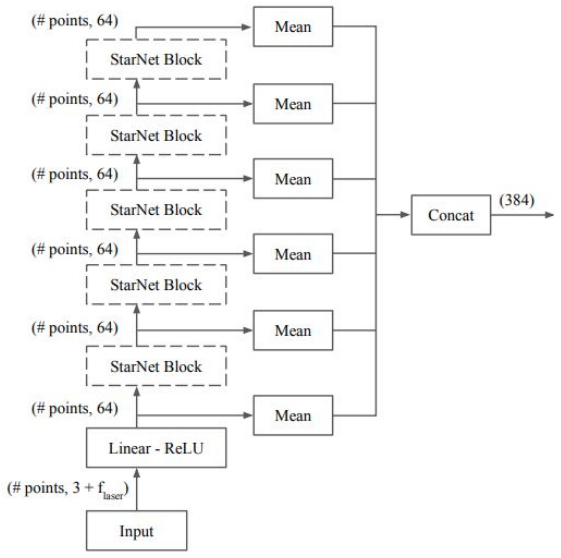


Figure 4.2: StarNet architecture[37]. Global features from multiple layers are concatenated together.

Synthesizing 3D LiDAR point clouds

Mixup of samples for enhancing generalization

Mix-up[72, 9, 23, 1, 20, 64, 56, 21, 59, 70, 58, 57] based approaches introduce a simple data augmentation method which mixes two different samples into one to improve the generalization and robustness of large neural networks. Given two training samples x and $x^{'}$ and their corresponding labels y and $y^{'}$, Mix-up first samples λ from asymmetric beta distribution and mixes x and $x^{'}$ using the following equation.

$$x_{new} = x * \lambda + x^{'} * (1 - \lambda) \tag{4.1}$$

The mathematical derivation induced by the [72] proves that binary cross entropy loss for classification task can use y as ground truth label for x_{new}. In other word, $y^{'}$ which is a ground truth label for $x^{'}$ could be omitted under this setting.

The parameter λ sampled from beta distribution plays a critical role in this result, while we omit the derivation of cross entropy loss for mixed training samples because it is out of scope in this paper. Intuitively, as $\lambda \sim$ Beta(0.1,1.1) is not symmetric, roughly 80% of sampled λ has value higher than 0.9. This makes x_{new} looks more similar to x than $x^{'}$ and mixing with $x^{'}$ works as noise being applied on x.

The proposed approach in epBRM_V2 takes the similar idea of mixing up two different samples into one with a bias on one of those samples. Unlike to the [72] which focuses on classification task, the proposed approach in epBRM_V2 focuses on box regression task. As 3D LiDAR point clouds defines the shape of objects, we cannot directly apply mixup method by concatenating two set of 3D LiDAR point clouds into one.

Instead of mixing two samples by concatenating two set of 3D LiDAR point clouds, we develop a novel approach for mixup process based on data-driven LiDAR simulator. What we mix in the proposed approach are a) dense 3D LiDAR point clouds with ground truth label which we name as 'prototype' of 3D LiDAR point clouds and b) sparse 3D LiDAR point clouds sampled from training set which provides the realistic pattern of LiDAR beam. By mixing these two sources of information, the proposed approach can create new training samples with a) ground truth label and b) realistic pattern of LiDAR point clouds.

4.4 epBRM_V2

In this section, we discuss three major modifications from epBRM_V1 to epBMR_V2. The modification can be summarized as followings: a) while epBRM_V1 uses existing detection networks for localization purpose, epBRM_V2 uses the network trained only for localization objective. b) the network structure of epBRM_V2 is deeper and wider than the network structure of epBRM_V1. c) we develop a novel data augmentation method for 3D LiDAR point clouds based on mix up method.

Localization module based on PointPillars

While epBRM_V1 improves the detection performance by building an end-point box-regression network on top of localization module, it uses the existing detection networks as its localization module. Even though this approach makes it easy to analyze how much detection performance gain does each baseline detection network get by box-regression network, the fact that each detection network is not trained only for localization task degrades the their localization performance.

The epBRM_V1 uses several detection networks such as AVOD-FPN, Frustum-PointNet and PointPillars as localization module. The epBRM uses the post-processed detection results from each detection network. The main part of post-processing is NMS operation which removes the redundant predictions by calculating the similarity between each prediction. In general, the similarity between predicted boxes is measured by the overlap between those boxes. If intersection over union between two boxes are higher than threshold, then we take only one prediction which has the highest confidence score among those boxes.

In this work, we focus on PointPillars which shows the best detection performance among the baseline detection networks which epBRM_V1 uses as localization modules. Instead of using the final detection output of PointPillars, we modify the network structure of PointPillars so it predicts only the location of objects and probability of the objects existence. In other word, the localization module for epBRM_V2 is not trained for predicting dimension and heading angle. This modification brings us important advantage in training a network: a complexity of localization task is simpler than the complexity of whole detection task which includes localization task as well as bounding box coordinates, so we can expect much faster training and higher localization performance by removing redundant output and loss function during the training.

However, as we remove the bounding box coordinate from the output of network, we cannot use box overlap based NMS operation. Instead, the proposed approach uses proximity based NMS operator which measures the similarity between predictions by the distance between predicted locations.

Proximity based NMS

Non-maximal suppression (NMS) operation is one of post-processing methods which is used to reduce the number of redundant predictions at validation and testing. Most detection networks which use anchor based approach typically generate multiple predictions at each anchor. In general, the distance between nearby anchors is a set as small to get high recall performance. Therefore, the number of output of detection network before NMS operation is usually very large.

It is well-known problem for most anchor based detection networks that several predictions overlap each other, which degrades the overall detection performance. For example, there might exist several predictions which correspond to a single ground truth. To reduce such redundant predictions, NMS operation sorts all predictions using their confidence scores and removes boxes one-by-one if each box has low confidence score and highly overlapped with other predicted boxes with higher confidence scores.

The localization module for epBRM_V2 predicts only the location of objects and their corresponding confidence scores. Therefore, the proposed approach cannot measure the similarity between predictions based on overlap between boxes. Instead of using overlap between boxes, we use distance between two predicted locations to measure the similarity between predictions.

Comparison of epBRM_V1 and epBRM_V2

We explore network structures recently developed by 3D object detection researchers. While keeping a simple structure of the epBRM_V1, the proposed approach has several modifications adopted from the recent improvements.

Network structure

The proposed approach adopts skip-connection layers between global features of PointNet for better representation ability of point clouds feature. The network structure of StarNet[37] inspires us to build dense skip connection layers between each global feature output. The skip-connection layer between very deep layers has been proven to be very effective to improve detection performance. Figure AA shows the comparison between the epBRM_V1 and the new epBRM_V2 network. As shown in Figure AA, epBRM_V2 has wider and deeper network structure with skip-connection between each global feature aiming for better represenation learning.

Target distance

We conduct in-depth analysis on the effect of target distance. The target distance is defined in a same way as it is first defined in epBRM_V1. Here, we simply explain about target distance of epBRM_V1 and epBRM_V2. For more detailed definition of target distance, please refer Section 3.3.

The target distance represents the maximum distance between predicted locations from the localization module and its nearest ground truth that we aims for further refining a 3D bounding box by using a box regression module. Therefore the target distance sets a limit of what box regression module is responsible for.

There are several things to consider for setting a proper value for target distance. First, the target distance is affected by the performance of localization module. If we increase the value of target distance, it means that we expect that even the prediction which is located far from the ground truth can be refined by the box regression module. However, if the localization module predicts the location of the object precisely, then a large value for the target distance would be redundant.

Second, the target distance is affected by the representation capability of box regression module. Increasing the target distance means that the complexity of the box regression task increases as well. If we constrain a representation ability of box regression module in a preference on a computational efficiency, the value of target distance should also be decreased according to that.

The epBRM_V1 uses the location predicted as a part of detection result from the AVOD-FPN[24], Frustum-PointNet[38, 39, 40], and PointPillars[26]. This approach makes us possible to easily analyze how much detection performance gain we can get by building a box regression module on top of various detection networks. However, we observe some predictions located proximal to the ground truth are removed during the Non-Maximal Suppression operation not because of wrong location predictions, but because of dimension and heading which are not related to localization of objects.

In this work, the proposed approach is aiming to increase the value of target distance by using the network with better representation power, in order to solve more complex bounding box regression task using an epBRM_V2. This modification leads epBRM_V2 to a better box regression performance for the objects which are located comparably far from the ground truth location compared to epBRM_V1.

To summarize, 1) epBRM_V2 aims at increase the recall performance of bounding box regression task. 2) To increase the recall, the proposed approach also needs to consider the predictions which are located far from the ground truth. 3) This requirement leads to increase the value of target distance. 4) As target distance is increased, the network encounters more complicated samples. With limited representation power of epBRM_V1, this leads to degrade the performance of bounding box regression task. 5) To precisely regress the bounding boxes which are located far from the ground truth, the proposed approach needs to build a new network, epBRM_V2, which has better representation power. Aiming better representation learning, we adopt deeper and wider network structure for epBRM_V2 than the structure of epBRM_V1.

Loss function for box regression task

The final output of the box regression module is the coordinate of 3D bounding box of each prediction which is represented as (x, y, z, l, w, h, yaw). In epBRM_V1, the loss functions for bounding box regression task are smooth L1 for (x, y, z, l, w, h) and hybrid of classification and regression loss for yaw prediction. In epBRM_V1, given a ground truth $(\hat{x}, \hat{y}, \hat{z}, \hat{l}, \hat{w}, \hat{h}, y\hat{a}w)$ and a prediction (x, y, z, l, w, h, yaw), the loss for bounding box regression task is given as follows:

$$\begin{aligned} Loss_{center} &= L_{smooth}(x, \hat{x}) + L_{smooth}(y, \hat{y}) + L_{smooth}(z, \hat{z}) + L_{smooth}(l, \hat{l}) + L_{smooth}(w, \hat{w}) \\ &\quad + L_{smooth}(h, \hat{h}) + L_{CE}(\text{yaw-cls}, \text{ya}\hat{\text{w}}\text{-cls}) + \mathbb{1}^{\text{ya}\hat{\text{w}}\text{-cls}}[L_{smooth}(\text{yaw-reg}, \text{ya}\hat{\text{w}}\text{-reg})] \end{aligned} \tag{4.2}$$

Note that loss for each coordinate is independent to the loss of other coordinates. This approach assumes that each of coordinate is not correlated with others which is actually a wrong assumption. Each corner of the 3D bounding box is calculated as a combination of the center location of the object, (x, y, z) and dimension/heading angle of the object, (l, w, h, yaw). To address the correlation between each coordinate, [24] uses an objective function which uses the difference between 8 corners of predicted bounding box and ground truth box.

Unlike to [24] which directly regresses the corner of the 3D bounding box, our proposed approach predicts the center based bounding box encoding method, which represents box as (x, y, z, l, w, h, yaw). However, unlike to loss function of epBRM_V1(Equation (4.2)), the epBRM_V2 uses the difference between 8 corners of predicted boudning box and ground truth box as augmented loss function, $Loss_{corner}$.

$$Loss_{corner} = \sum_{c \in corners(8)} (L_{smooth}(x_c, \hat{x_c}) + L_{smooth}(y_c, \hat{y_c}) + L_{smooth}(z_c, \hat{z_c})) \tag{4.3}$$

where (x_c, y_c, z_c) is c th corner of predicted 3D bounding box and $(\hat{x_c}, \hat{y_c}, \hat{z_c})$ is c th corner of ground truth 3D bounding box for $c = 1, 2, ..., 8$.

The overall loss function for epBRM_V2 is sum of $Loss_{center}$ and $Loss_{corner}$ where λ is hyper-parameter which set as 1.0 in this work.

$$Loss = Loss_{center} + \lambda * Loss_{corner} \tag{4.4}$$

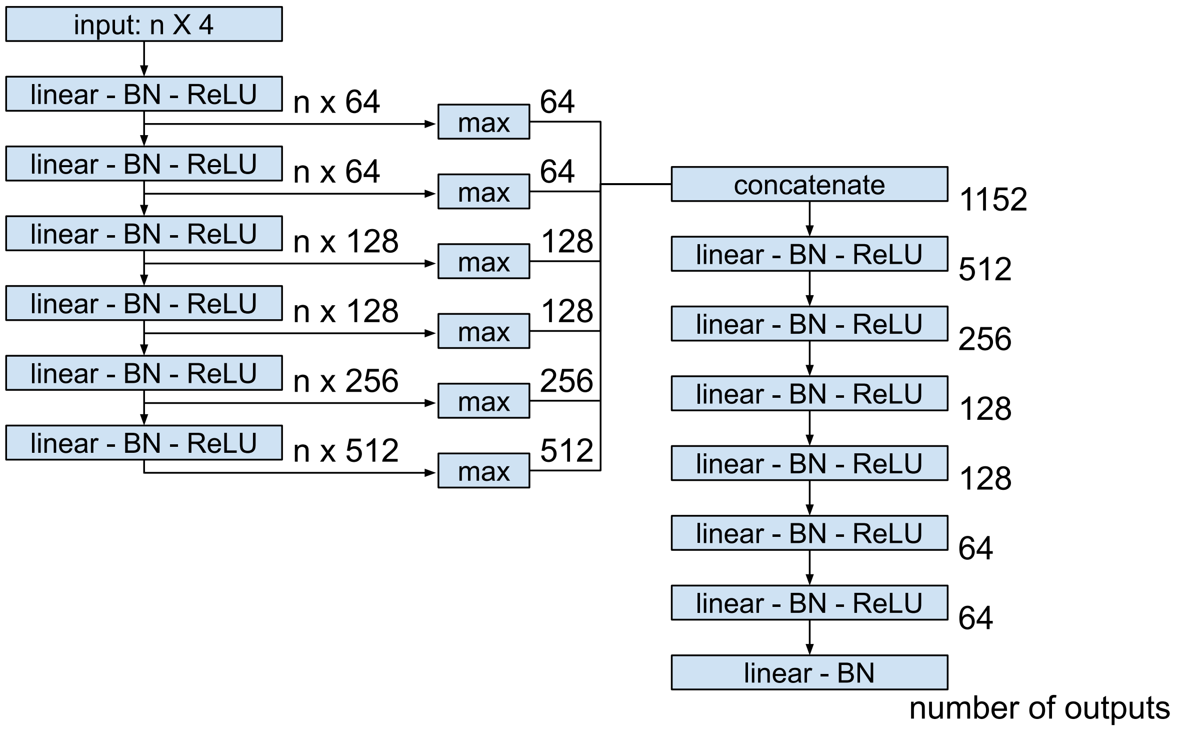


Figure 4.3: Network structure of module in epBRM_V2. As in epBRM_V1, we use spatial transformation based approach for better representation capability. To use spatial transformation based approach, we take modular approach as shown in Figure 4.4. We apply same network structure for two epBRM modules in Figure 4.4.

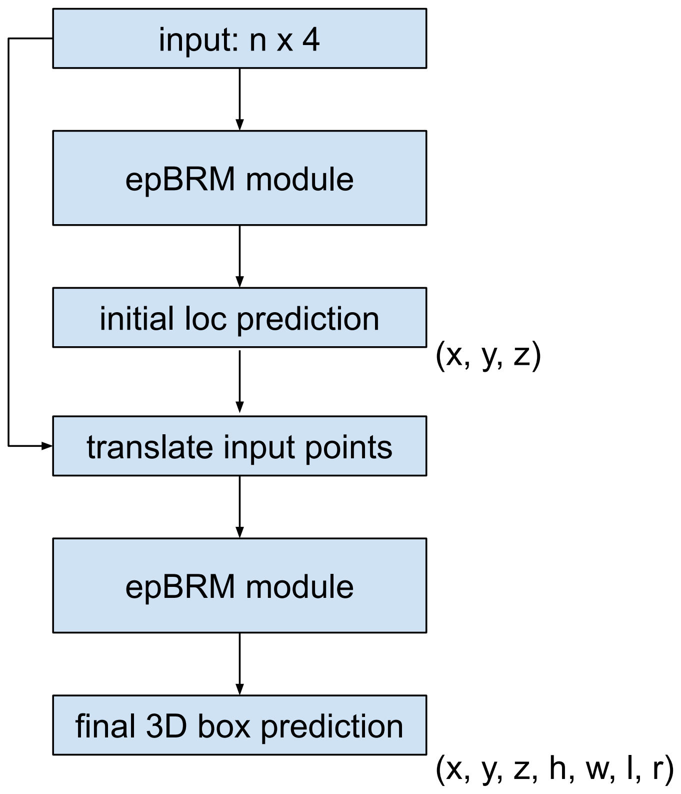


Figure 4.4: Network structure of epBRM_V2. The proposed approach first predicts the center location of the objects. Then, the proposed approach moves all input point clouds in order to place the predicted location at the origin. As a final ouput, the proposed approach predicts the 3D bounding box coordinates. Note that two epBRM modules don't share the parameters.

4.5 Data augmentation method for 3D LiDAR point clouds

In this section, we describe our novel data augmentation method for 3D LiDAR point clouds based on mixup of two set of point clouds from training samples. The proposed approach aims at improving the generalization of the box regression network by increasing the number of training samples as well as diversity among them which can help improving the quality of bounding box regression.

The Figure 4.5 briefly describe the concept of mixup approach for 3D LiDAR point clouds. Figure 4.6 and Figure 4.7 describe the overall process for creating a new dataset which contains ground truth label and realistic pattern of 3D LiDAR point clouds.

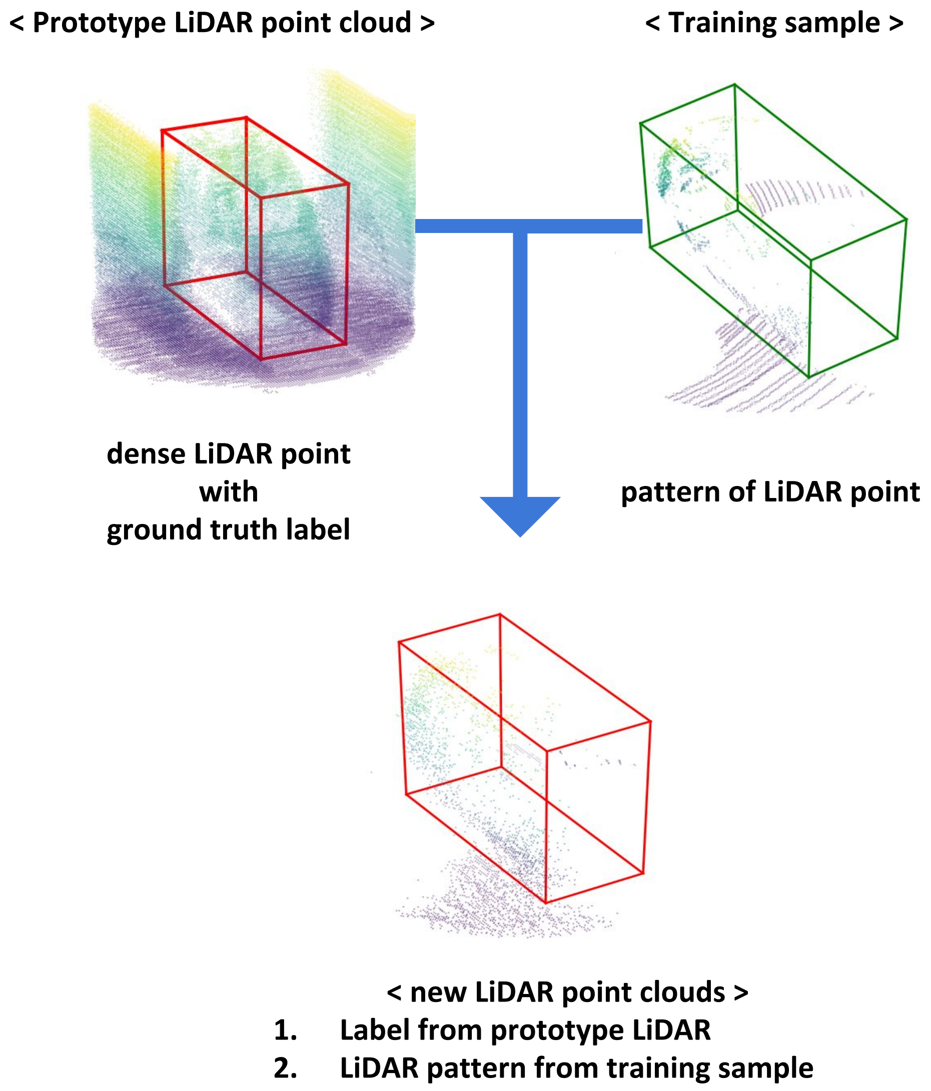
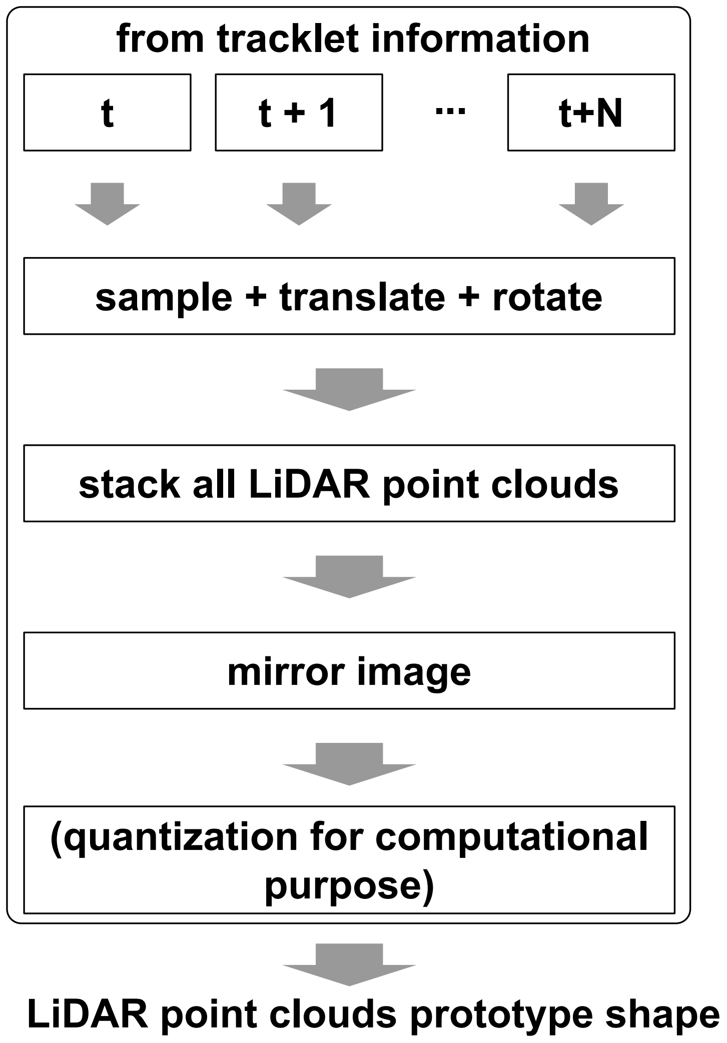
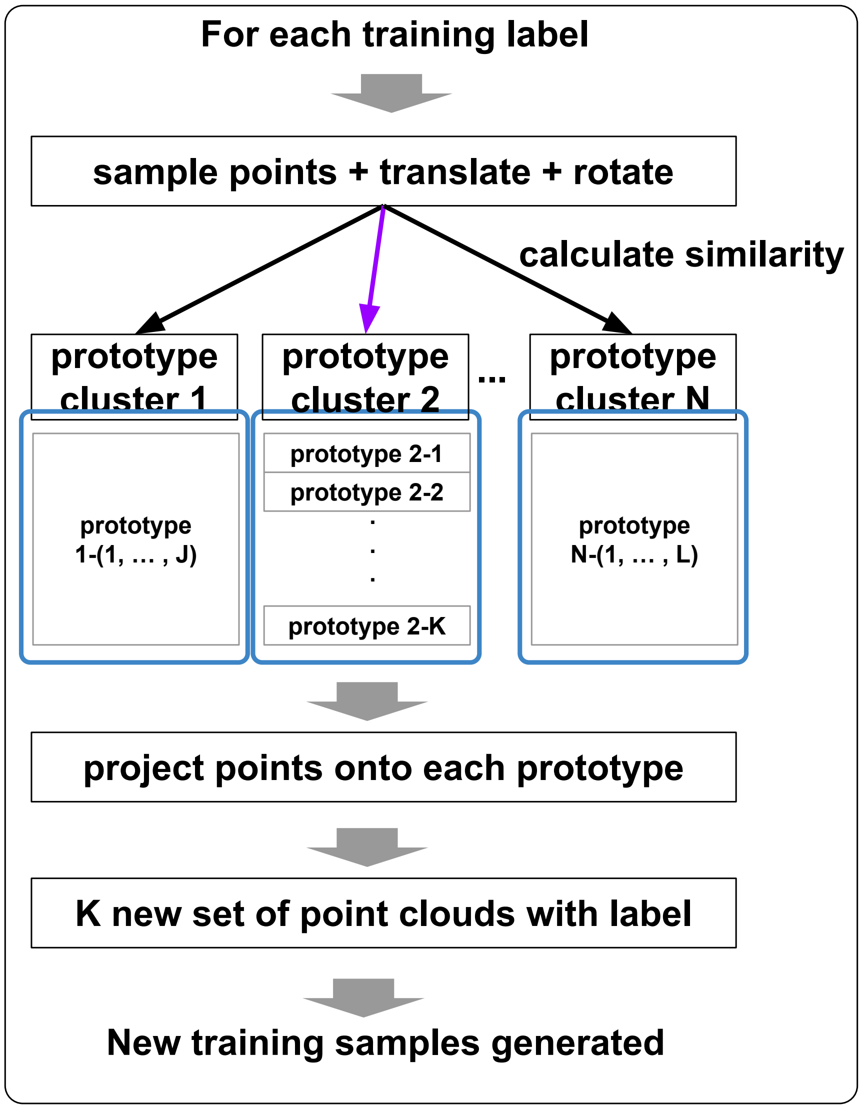

< Prototype LiDAR point cloud >
< Training sample >
dense LiDAR point with ground truth label
pattern of LiDAR point
< new LiDAR point clouds >
1. Label from prototype LiDAR
2. LiDAR pattern from training sample

Figure 4.5: Conceptual explanation on how to create a new training sample from a prototype 3D LiDAR point clouds and training sample. The proposed approach is basically a data-driven 3D LiDAR point clouds simulator which consumes an actual pattern of LiDAR point clouds observed from training sample.


from tracklet information
t
t + 1
···
t+N
sample + translate + rotate
stack all LiDAR point clouds
mirror image
(quantization for computational purpose)
LiDAR point clouds prototype shape

(a) LiDAR prototype generation process

Figure 4.6: Process to generate a dense 3D LiDAR point clouds from the tracklet information which is generally provided from the dataset. We call the dense 3D LiDAR point clouds which is an output of this process as Prototype



(a) LiDAR sample generation process using prototype shape

Figure 4.7: Steps to generate new training samples with labels and realistic pattern of LiDAR point clouds. This mixup based approach uses LiDAR point clouds prototype generated by Figure 4.6 and each training sample.

Create a library of prototype 3D LiDAR point clouds from tracklet

In this section, we describe how to create a library of prototype 3D LiDAR point clouds. We define a prototype 3D LiDAR point clouds as a dense set of point clouds whose ground truth label is known. Once we generate a library set which contains multiple prototype 3D LiDAR point clouds, the proposed approach uses the library and their label to create new training samples. The process of creating a new training sample using a library of prototype is discussed in Section 4.5.

A density of prototype 3D LiDAR point clouds should be dense. However, a density of 3D LiDAR point clouds is generally very sparse and it requires special equipment to get dense 3D LiDAR point clouds. Instead of using additional equipment to get dense 3D LiDAR point clouds, our proposed approach exploits **tracklet** information which contains ground truth labels which track each instance over the temporally continuous frames.

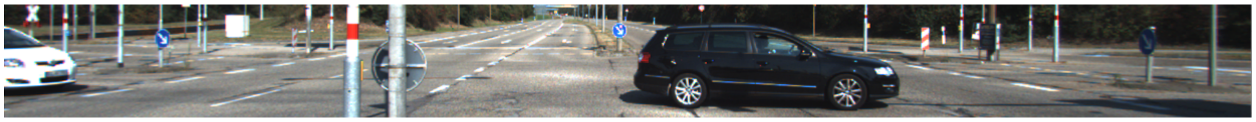
The biggest advantage of using tracklet information for creating a library of prototype 3D LiDAR point clouds is that we can get these information very easily from the public dataset such as KITTI[15], Waymo dataset from Waymo[55], nuScenese dataset from APTIV[2] and ArgoVerse dataset from ArgoAI[6] provide tracklet information which is instance-level ground truth labels tracking the objects throughout the continuous frames.

Figure 4.11 shows 5 frames which are temporally continuous. The ground truth labels for each vehicle are given as a form of tracklet information. Figure 4.9 visualizes the Bird's Eye View image of 3D LiDAR point clouds as well as ground truth boxes which track the black vehicle shown in Figure 4.11.

(a) dataset captured at 0.0s

(b) dataset captured at 0.1s

(c) dataset captured at 0.3s


(d) dataset captured at 0.4s

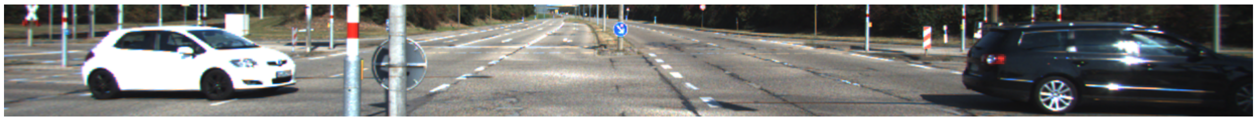
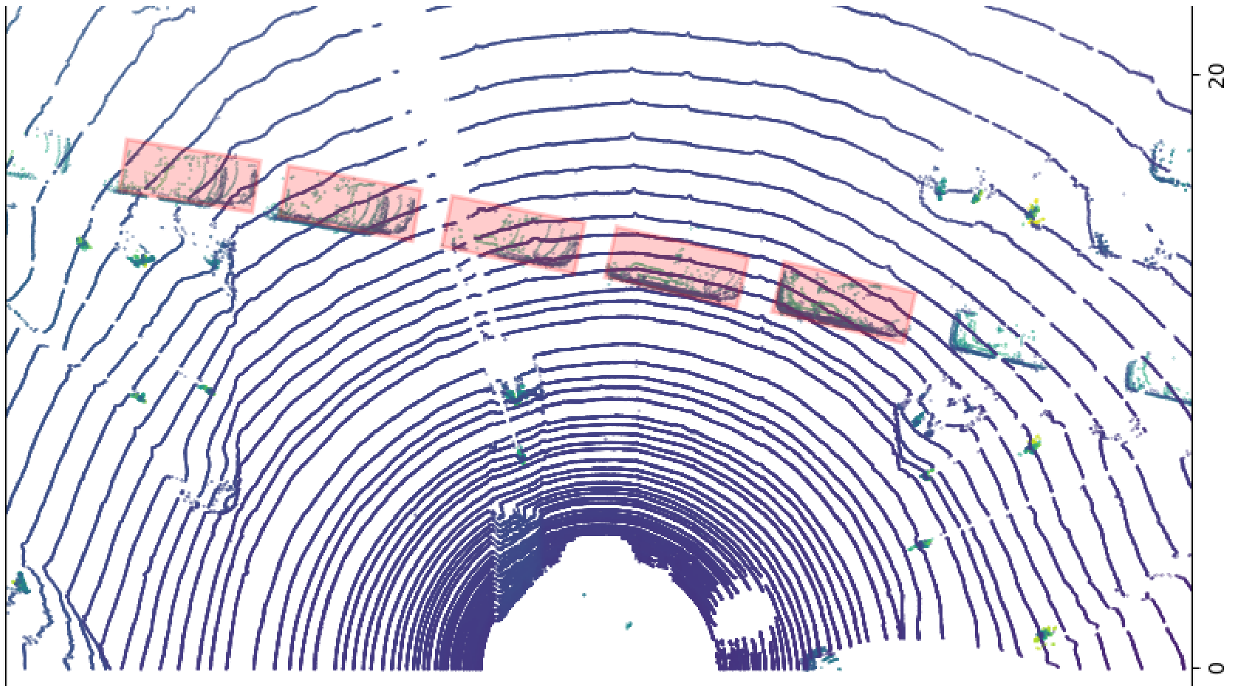
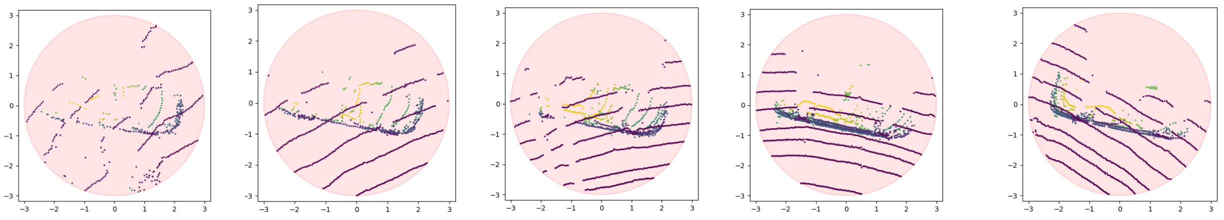

(e) dataset captured at 0.5s

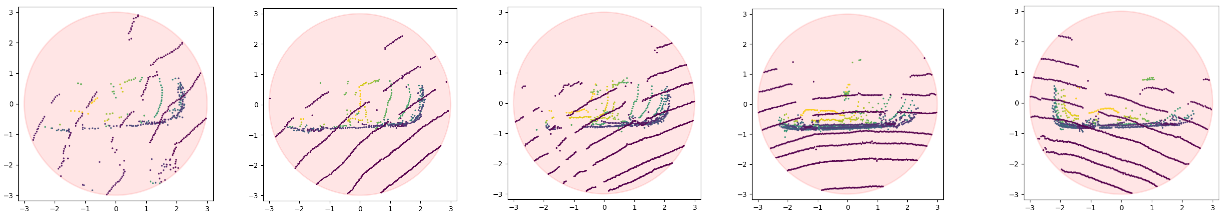
Figure 4.8: This figure shows the temporally continuous frames. A tracklet of the black vehicle in this figure provides the information about the location and heading angle relative to the ego-vehicle as well as dimension of the object. Figure 4.9 shows the Bird's Eye View image of LiDAR point clouds with labels of black vehicle.



(a) Bird's Eye View image of 5 sequential frames.



(b) LiDAR point clouds sampled using ground truth labels.



(c) LiDAR point clouds after aligning to the same yaw angle

Figure 4.9: Bird's Eye View image showing the sampling process of 3D LiDAR point clouds using tracklet information of black vehilcle shown in Figure 4.11. Tracklet provides the heading angle of the object in multiple time frames. The proposed approach uses those information to align all LiDAR point clouds into the same direction so that we can concatenate them together.

A. Sampling 3D LiDAR point clouds around ground truth location

Tracklet is a set of the ground truth labels which captures the movement of each instance. Depending on the datasets, it may contains information not only about location, heading angle, dimension of the object but also the occlusion level, velocity, and etc.

In this step, we use location information extracted at each time stamp to sample point clouds that belongs to the objects. Unlike to camera image, the 3D LiDAR point clouds is an un-ordered set of spatial information. Therefore, we can easily build overlapped point clouds by stacking point clouds which are sampled from all different time stamps.

As in [49] and [50], the proposed approach samples point clouds in a cylindrical sampling region centered at the ground truth location of each time stamp.

Since all point clouds are sampled from different time stamp, the relative locations to the ego vehicle are all different. As a result, we can see different pattern of 3D LiDAR point clouds at each time stamp as shown in Figure 4.9.

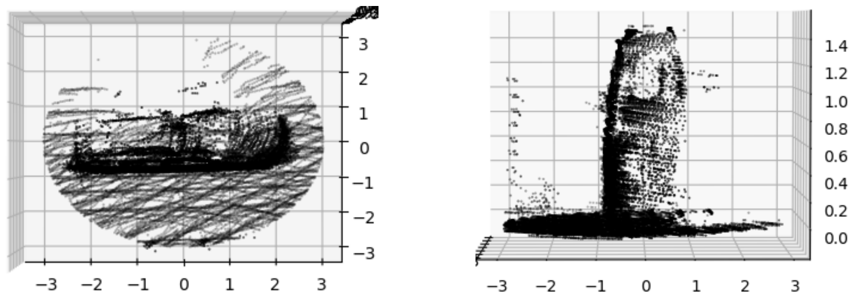
B. Aligning all point clouds into the same configuration

The relative locations of an object to the ego-vehicle are different at each timestamp. To create dense point clouds which can show the clear shape of the object, we have to align each set of point clouds at a fixed location and fixed orientation. As shown at the third row of Figure 4.9, we translate the point clouds to the origin and rotate all points around z axis so all point clouds are aligned each other.

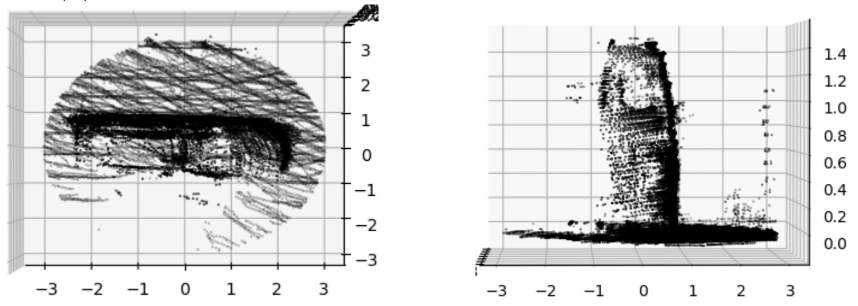
Without aligning all point clouds into the same heading angle, we cannot get clear pattern on 3D LiDAR point clouds even though we stack them together.

Note that 3D LiDAR point clouds shown in left most figure can contribute to build front-right part of vehicle while 3D LiDAR point clouds shown in right most figure can provide information to construct back-right part of vehicle. Sampling point clouds from multiple time stamp can provide more complete shape information about the objects.

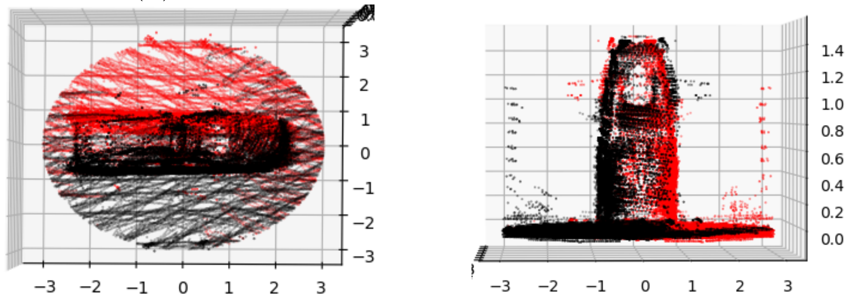
As a result, while 3D LiDAR point clouds sampled at single time stamp can provide only a fractional of the shape of an object, we can create a new 3D LiDAR point clouds by reconstruction from the multi-directional views.



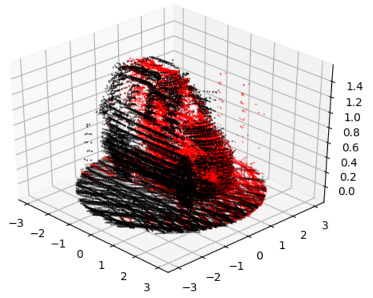
(a) LiDAR point clouds stacked over multiple time stamps.



(b) LiDAR point clouds after mirroring process

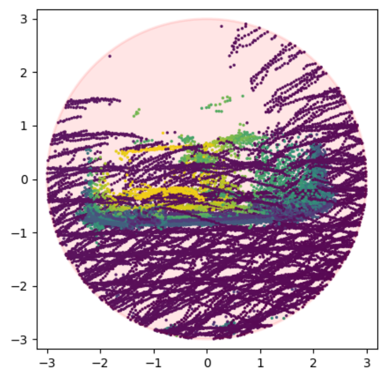


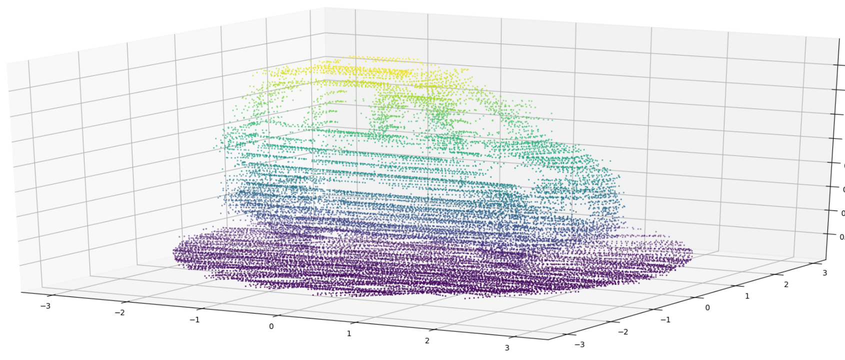
(c) LiDAR point clouds (a) stacked with mirrored point clouds (b)



(d) 3D view of 3D LiDAR point clouds after mirroring process

Figure 4.10: Mirroring process of 3D LiDAR point clouds to build more dense prototypes. This process uses the mirrored property of the objects.


(a) BEV of concatenated point clouds


(b) 3D view of concatenated point clouds

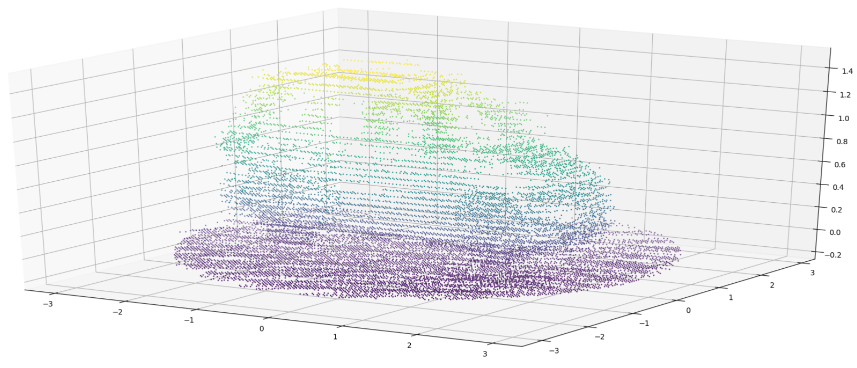

(c) After quantization

Figure 4.11: Quantization process to control the density of point clouds

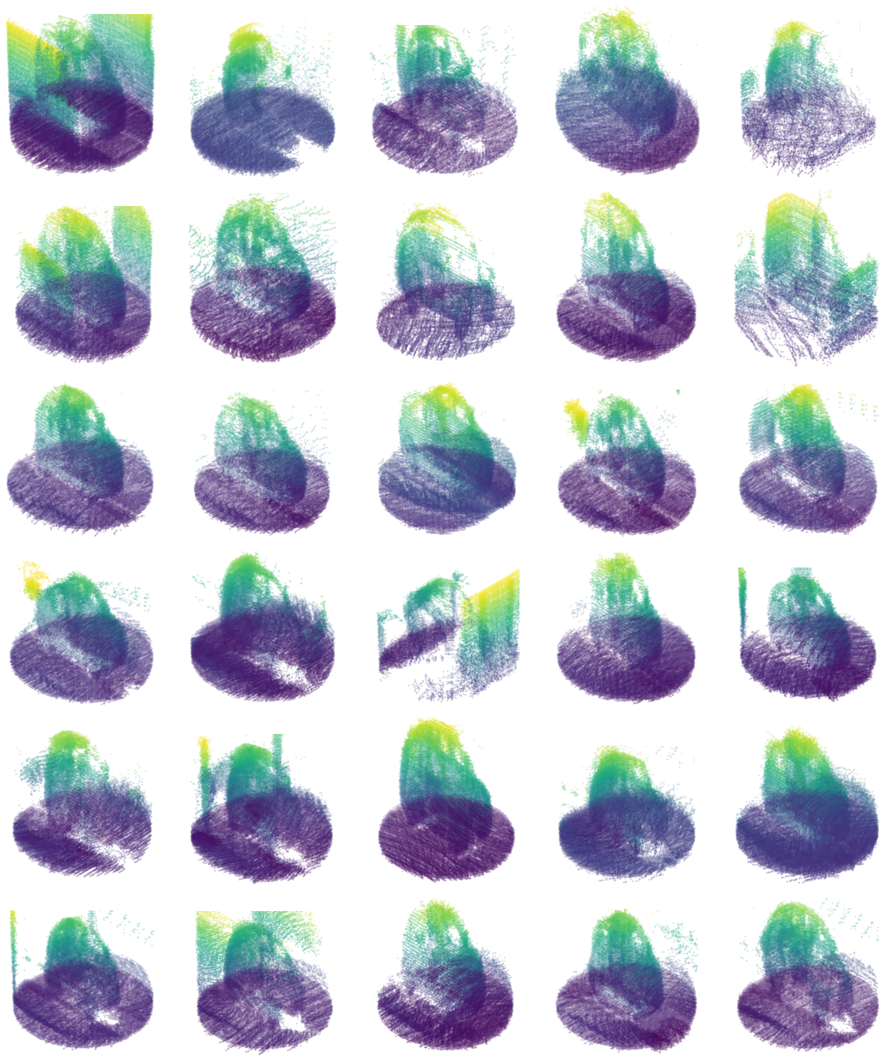


Figure 4.12: Subset of prototype 3D LiDAR point clouds. The density of 3D LiDAR point clouds of each prototype is much dense compare to the density of point clouds sampled from single frame. Note that ground truth label of each prototype is known from the tracklet information.

C. Stacking all aligned point clouds

In this step, the proposed approach actually stacks all point clouds from multiple time stamp so that it can create very dense point clouds of the objects.

D. Mirroring process

The proposed approach uses mirrored property of the objects to create even more complete prototype 3D LiDAR point clouds as shown in Figure 4.10. Figure 4.10 (b) shows the mirrored 3D LiDAR point clouds of Figure 4.10 (a). Figure 4.10 (c) and Figure 4.10 (d) show the LiDAR point clouds after mirroring process.

In most case, only the half part of the vehicle is visible from the 3D LiDAR point clouds because of the relative location between the object and the ego-vehicle. However, based on the fact that most vehicles have mirrored shape around the axis, the mirroring process helps to increase the density of point clouds which results in giving us a more clear shape of 3D LiDAR point clouds than before.

E. Quantization applied to avoid too many points in small area

After processing 3D LiDAR point clouds as described above, the total number of point clouds which consists the prototype can be excessive. The excessive number of point clouds requires heavy computation for the latter step and high volume of memory to store the information. Therefore, opposite to the previous steps which aims at increasing the density of 3D LiDAR point clouds, this process removes the redundant point clouds by quantization.

While having a number of point clouds can provide dense and more complete shape of prototypes, we apply quantization process on a set of point clouds to avoid excessive points within the tiny space and control the density of point clouds in a proper level. This step removes all points except for one within $0.05m$ x $0.05m$ x $0.05m$ space. Before applying quantization, the proposed approach randomly permutes all point clouds.

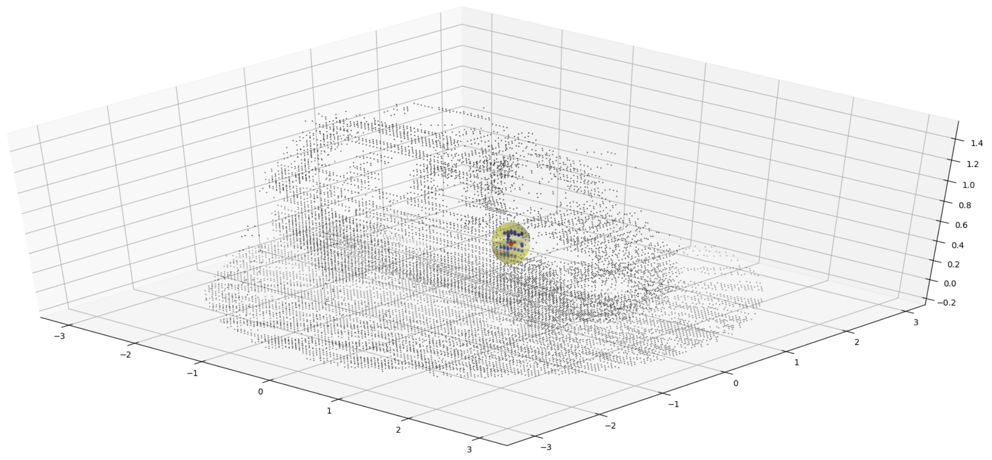
Figure 4.12 shows the subset of prototype 3D LiDAR point clouds created by the processes described above.

Create a new training sample using a prototype and existing training samples.

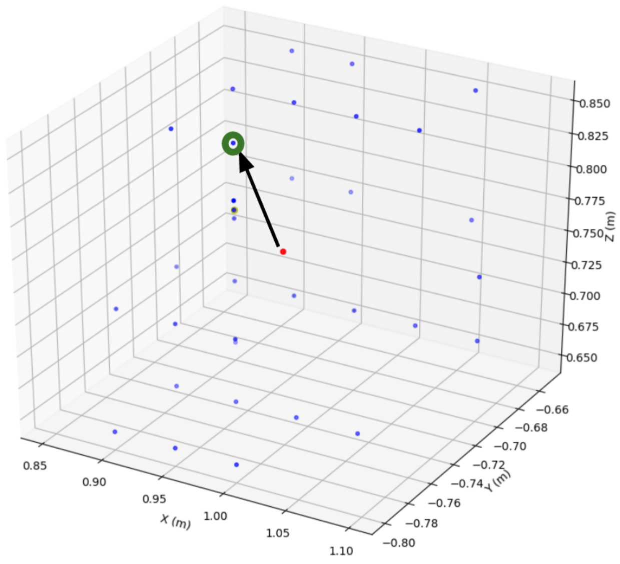
Figure 4.12 shows the subset of prototype 3D LiDAR point clouds created by the processes described above. In this section, we describe how to generate new training samples which will be used for training an epBRM_V2.

Figure 4.5 visualizes the process of generating new training samples using a prototype 3D LiDAR point clouds and training samples. There are two source of information required to create new training samples: a) ground truth label, b) realistic pattern of LiDAR point clouds reflected from the vehicle.

Figure 4.7 briefly explains how the proposed approach creates a new training sample with a ground truth label and realistic pattern of LiDAR point clouds.



(a) Boundary of sampling region centered at point from training sample



(b) Activation of point of prototype. Red: point from training sample, Blue: points from prototype which are located within the boundary from red point, green: one of blue points which is randomly sampled to be activated.

Figure 4.13: Activation of each point in prototype. The proposed approach calculates the distance between each of the point from training sample and the point from prototype. For each point of training sample, it randomly samples one point from the set of points within the boundary. The boundary is given as spherical shape.

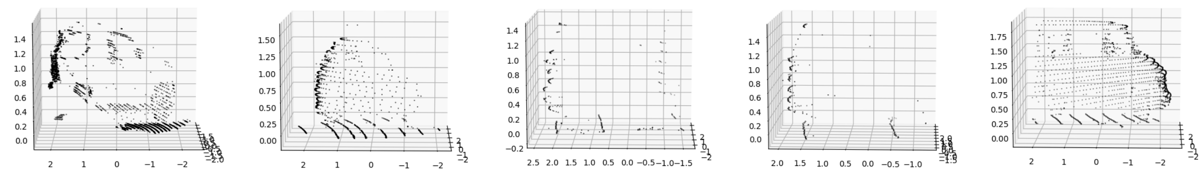
A. Data-driven LiDAR simulator based on activation of point clouds

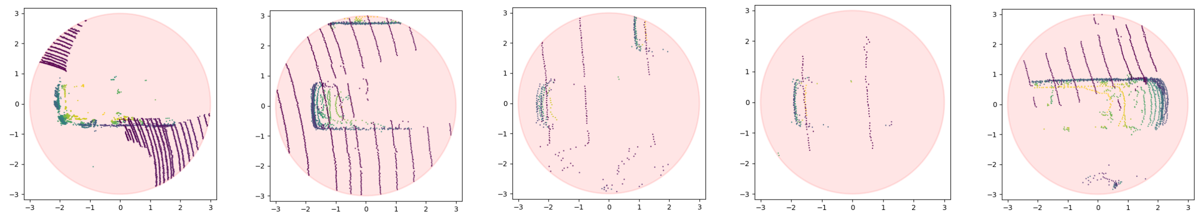
[35] uses the physical characteristic of the LiDAR equipment for building a raycasting engine and deep learning based approach for raydrop simulator. In this work, the proposed approach uses much simpler method which simulates the pattern of LiDAR beam reflection based on data-driven method.

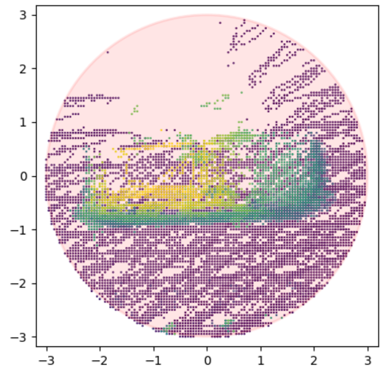
The proposed approach in this work is based on activation of each 3D LiDAR point of the prototype. The activation of each point of prototype is decided by whether there exists a point of training sample within the specific boundary.

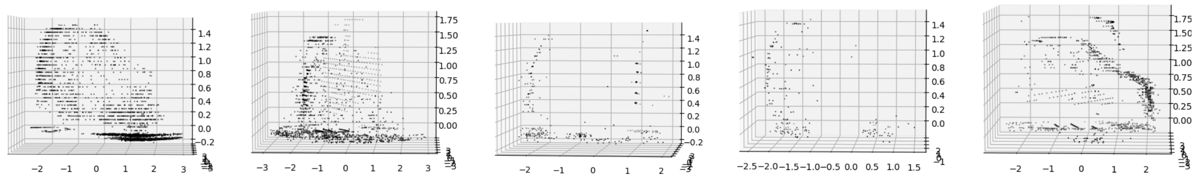
Figure 4.13a and Figure 4.13b visualize how the proposed approach activates each point in prototype 3D LiDAR point clouds.

Figure 4.14 shows how each training sample can be combined with prototype


(a) training samples (side view)


(b) training samples (bird's eye view)


(c) prototype 3D LiDAR point clouds (BEV view)


(d) activated points of prototype 3D LiDAR point clouds (side view)

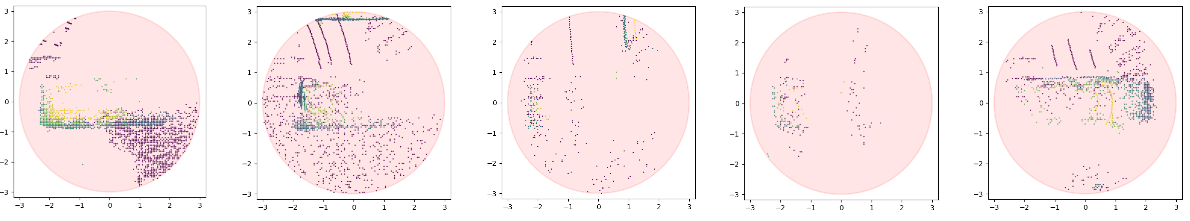

(e) activated points of prototype 3D LiDAR point clouds (bird's eye view)

Figure 4.14: Process of creating new training samples from the combination of training samples and prototype 3D LiDAR point clouds.

B. Clustering of prototype to avoid heavy computation

The proposed approach can generate new training samples which have novel shape and realistic pattern of 3D LiDAR point clouds. However, one limitation of this approach is that when the size of prototype and size of training sample are significantly different, the number of activation will be significantly smaller than the number of point clouds from training sample.

Assume that a prototype is generated from the large vehicle such as truck, and a training sample is categorized as small vehicle such as mini-sedan. Since the size of two objects are significantly different, no points of the prototype can be located within the boundary around each of point of the training sample. This leads to no activation of point of prototype, thus we get meaningless training sample from this combination.

Also, the number of combination available between prototype and training samples is numerous which imposes heavy memory requirement as well as computation resources. For example, from the KITTI dataset, we can generate 200 prototype LiDAR point clouds that has at least 10k points after quantization and each of them can be combined with each of training sample which is 20k. The total available number of combination is 200 * 20k = 4M.

To make this process more computationally efficient, the proposed approach generates cluster of prototype 3D LiDAR point clouds based on the dimension of each prototype, height, width, and length. The proposed approach applies K-Means clustering algorithm for width, length and height of the prototypes. Figure 4.15 visualizes the cluster of size of prototypes in 3D space.

The number of clusters, K, is a hyper-parameter we explore its effect by experiment. Each cluster may contain different number of prototype LiDAR point clouds. For example, a cluster with large or small objects may contain less number of prototype LiDAR instances compared to a cluster with normal size objects.

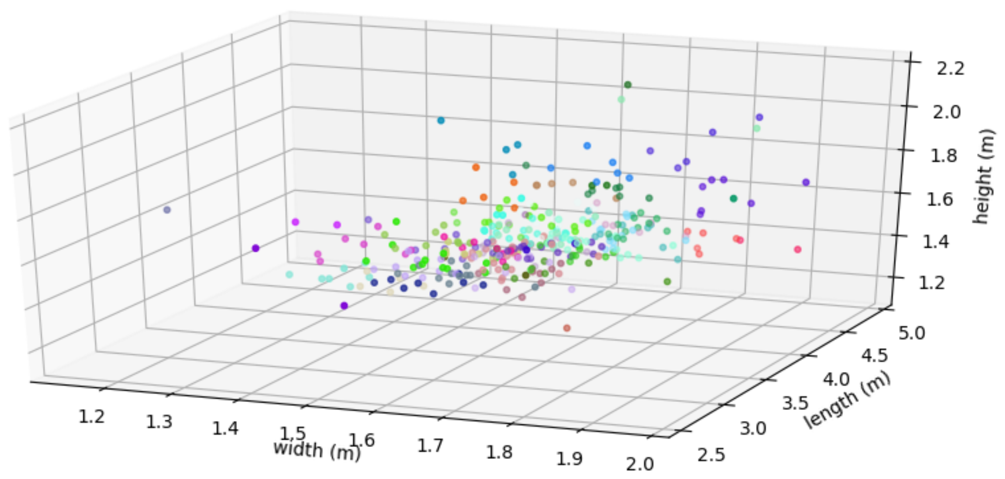


Figure 4.15: Visualization of clustering result. points with same color belong to same cluster. The proposed approach generates 50 clusters out of 250 prototypes.

4.6 Experiments

Dataset and metric

We evaluate the proposed method using KITTI dataset, the most frequently used 3D object detection benchmark. In this work, we focus on car class which has most training examples and symmetric shape around center axis. This property makes us easy to apply mirroring process for creating a library of prototype 3D LiDAR point clouds.

The detection results are evaluated according to three difficulty levels based on the occlusion/truncation/distance between the objects and the ego-vehicle. Among the three difficulty levels which includes easy, moderate and hard, we evaluate the proposed method on moderate level, which is a standard metric for evaluation.

For evaluating the localization module, we use 0.4m Euclidean distance based threshold in 3D space. For 0.4m distance threshold, the predicted location is true positive if there exists a ground truth object within 0.4m distance from the predicted location.

For evaluating the overall detection performance, we use 0.7 3D IoU threshold. If the 3D IoU between predicted 3D bounding box and the ground truth 3D bounding box is greater than 0.7, then the predicted 3D bounding box is true positive.

We evaluate the proposed method on training/validation set which are splitted from the training samples. We follow standard method to split a whole training samples into smaller training and validation set.[3, 24, 38]. In this work, training sample contains 3,717 frames and validation sample contains 3,769 frames. Frames in each split doesn't belong to same video clips.

Performance of a localization module without bounding box regression task

Problem statement

In this section, we mainly explore the localization performance as standalone module. The epBRM_V1 [50] analyzes its detection performance by using existing detection networks, such as AVOD-FPN[24], Frustum PointNet[38], and PointPillars[26] as its localization module. Since the PointPillars[26] shows the best performance among them, in this work, the proposed approach re-uses the structure of PointPillars except for removing the 3D bounding box related predictions from the last layer.

We explore the effect of the size of the pillars which controls the resolution of BEV image of 3D LiDAR point clouds. The size of the pillars is in trade-off relationship between computational cost and detection performance. If the size of pillars increases, it means that resolution of BEV image decreases, thus reduces the computational cost. However, coarse resolution generally leads to degraded detection performance.

As we remove the box regression task from the detection task and focus only on localization task, we assume that even coarse resolution of pillars is good enough to achieve high localization performance. The proposed approach compares the localization performance in different size of the pillar.

Reducing the computational cost is critical for most modern 3D object detection systems for self driving vehicles which aim to detect objects in far distance for safety purpose. If 3D object detection systems rely on high resolution BEV image, it becomes hard to meet the requirements for real-time pipeline.

Experiment setting and result

In this section, we describe the experiment setting for training a localization module. We mainly explain how to assign ground truth to anchors and proximity-based NMS operation.

For localization module, we follow the exactly same procedures for training a PointPillar network[26], except for how to assign ground truths to anchors. Since the proposed approach uses only location information, it cannot measure the similarity between pre-defined boxes on anchors and ground truth boxes using IoU metric.

Instead, the proposed method measures the similarity between anchor location and ground truth location using a distance between them. If the distance between the anchor location and the ground truth location is within 1 m, then the proposed approach assigns the ground truth to that anchor.

For NMS operation, we set NMS_{score}, the score threshold of NMS as 0.15 and NMS_{sim}, the similarity threshold of NMS as 0.5 m. This means that all predictions with confidence score less than 0.15 will be removed from the evaluation. And if the distance between two predicted location is less than 0.5 m, then one of those prediction which has lower confidence score than the other will be removed from the evaluation.

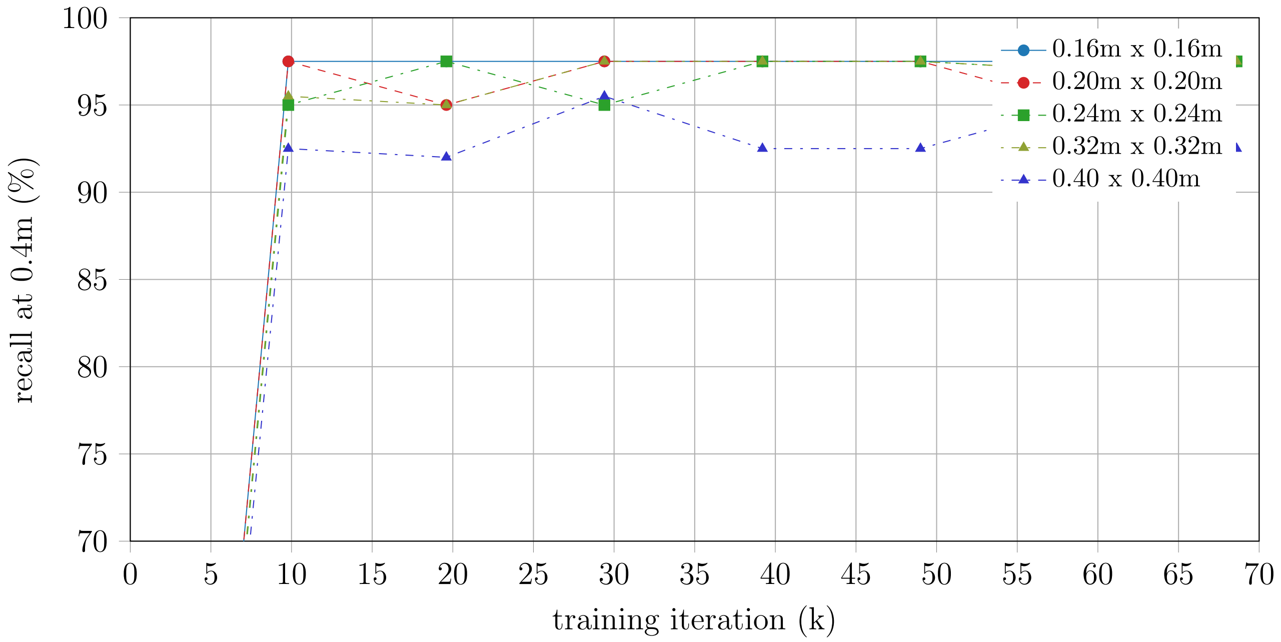
The localization network is evaluated at each 5 epoch. We use batch size of 2 and initial learning rate of 0.0002 with exponential decay. The localization performance of the proposed approach is evaluated using a recall and a mean average precision(: mAP) metric.

Figure 4.16 shows that the recall and mAP of localization performance with 0.4m distance threshold. We cannot see any performance degradation as we increase the size of pillar until 0.32m x 0.32m.

We can reduce the 4x computational cost by using 0.32m x 0.32m resolution of BEV image instead of 0.16m x 0.16m resolution of BEV image while keeping the same localization performance.

Proximity based NMS operation and precision-recall metric

In this section, we explore the effect of the NMS_{score} and the NMS_{sim} on the localization performance. The NMS_{score} is a parameter which decides whether to keep/discard the prediction based on the confidence score. The detection networks keep only the prediction which has confidence score higher than the NMS_{score}. Therefore, if a value of the NMS_{score} is low, it means that most predictions are kept for evaluation, thus we get a high recall and a low precision performance.



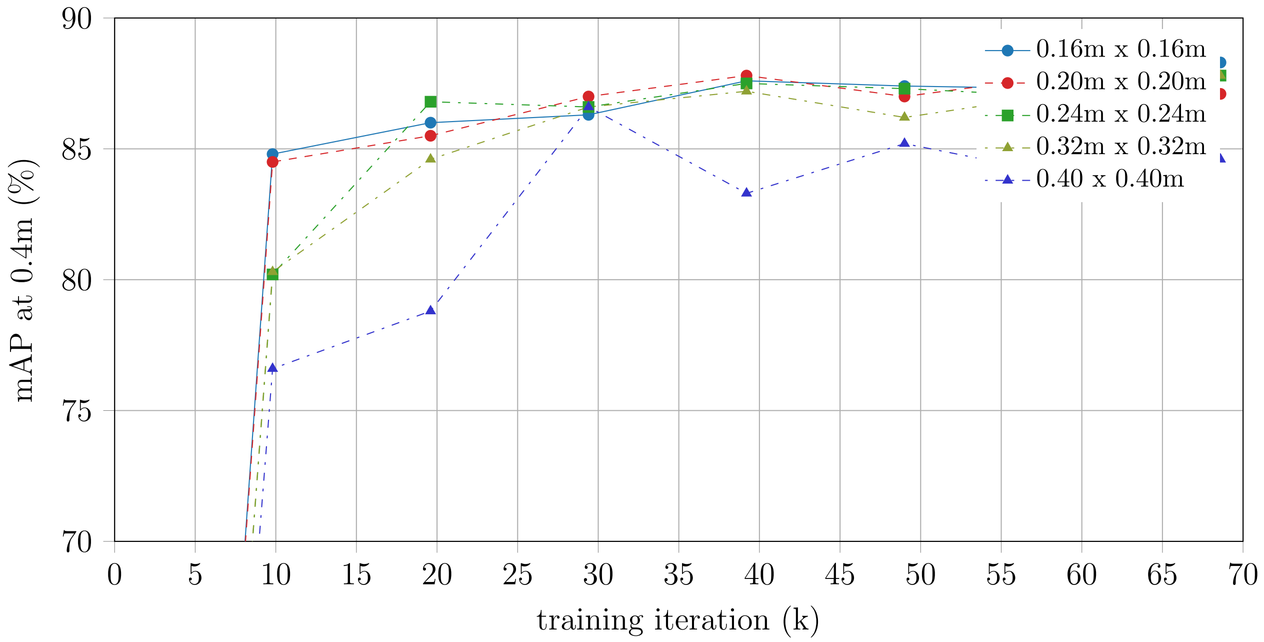


Figure 4.16: A comparison of the localization performance in different resolution of BEV image of 3D LiDAR point clouds.

The NMS_{sim} is used for measuring the similarity between two predictions and deciding whether two predictions actually predicts the duplicated or not. If the distance between two predicted location is lower than the NMS_{sim}, we regard those predictions are duplicated and keep only one of them which has higher confidence than the other. Therefore, if the value of the NMS_{sim} is low, most predictions are kept for evaluation, thus gets high recall and low precision performance.

We conduct extensive experiments with various combination of NMS_{score} and NMS_{sim} to evaluate the performance of the proposed localization module. Since there exists numerous number of combination available, we show the result which gives the best localization performance. In this experiment, 0.40m x 0.40m of pillar resolution is fixed. To evaluate the effect of the NMS_{sim}, we fix the value of the NMS_{score} as 0.15. To evaluate the effect of the NMS_{score}, we fix the value of the NMS_{sim} as 0.5m.

Figure 4.17 show that the NMS_{sim} of 0.5m and the NMS_{score} of 0.15 show the best performance both in recall and mAP of the localization network. Unlike epBRM_V1 which relies on the detection results of AVOD-FPN[24], Frustum PointNet[38], and PointPillars[26], we conduct extensive experiments to analyze various components which might affect the localization performance.

Performance of epBRM_V2

Problem statement

In epBRM_V1, we get the optimal value of target distance for each detection network as a result of experiments. Please refer Figure 3.7 about how we get the optimal value of target distance for each detection network and Table 3.5 for the detection performance of each network at their optimal target distance. The optimal target distance of PointPillars is 0.15m. This means that the epBRM_V1 is only responsible for the localization predictions which are actually within 0.15m from the ground truth objects.

While the epBRM_V1 successfully improves the overall detection performance, we aim further improvement by increasing the recall performance. To improve the recall performance using a box regression network, the box regression network should be able to refine predictions which located not only close to but also far from the ground truth objects. To precisely refine the predictions located far from the ground truth, we need to increase the value of target distance.

With a more sophisticated network structure of epBRM_V2 compared to epBRM_V1, we evaluate the proposed approach in various target distances.

Experiment setting and result

The proposed approach is evaluated in various target distance from 0.1m to 0.5m. The network structure of epBRM_V2 is shown in Figure 4.3 and Figure 4.4. The objective function for 3D bounding box regression task is stated by Equation (4.2), Equation (4.3),

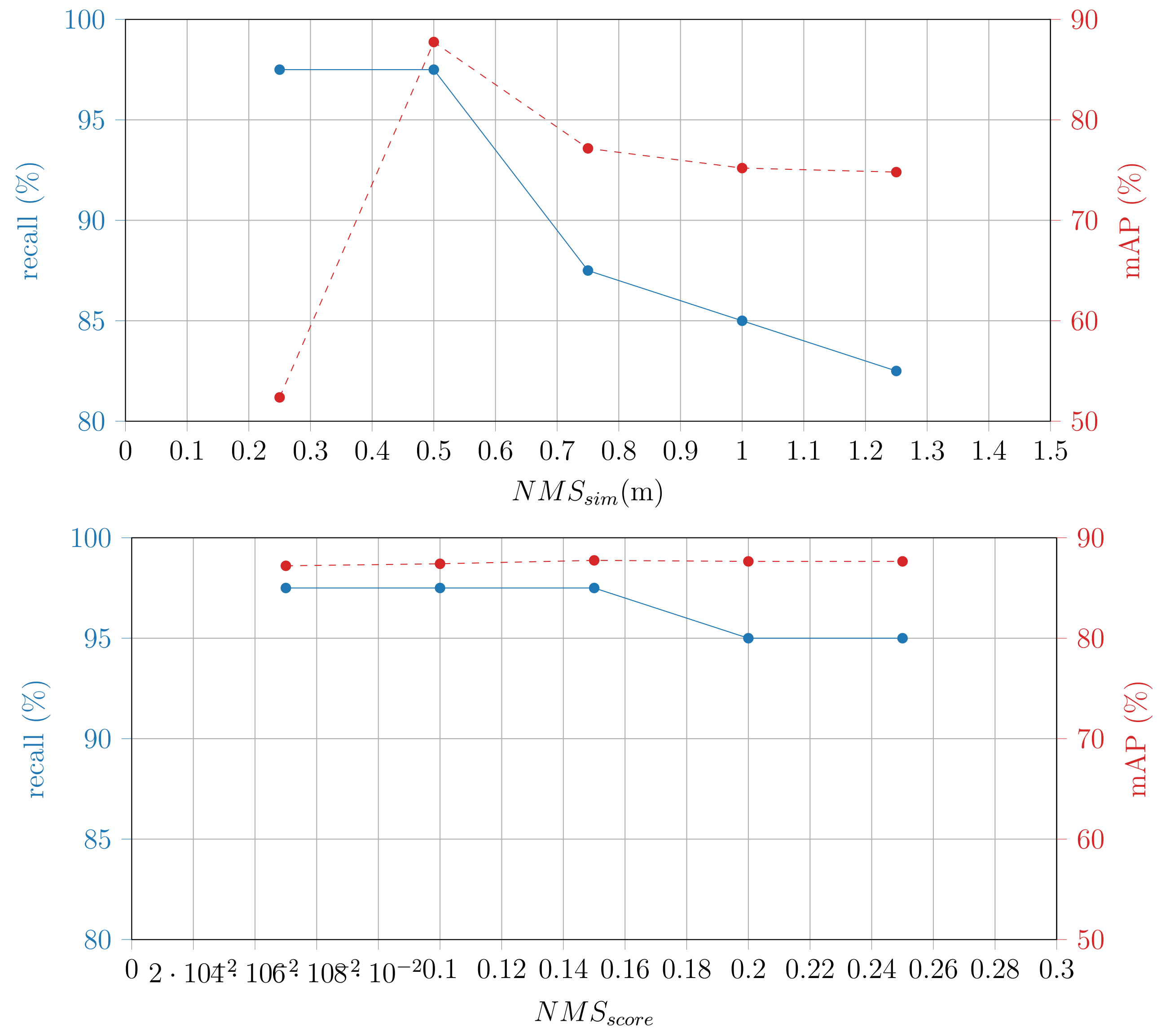


Figure 4.17: The effect of NMS_{score} and NMS_{sim} on the localization performance

and Equation (4.4). In this experiment, the proposed approach applies only the general data augmentation methods used in epBRM_V1, and doesn't apply the data augmentation method for 3D LiDAR point clouds based on mixup.

Figure 4.18 shows that the target distance of 0.4m shows the best performance both in recall and mAP. This result is surprising because the recall performance with target distance of 0.4m is 95%. Note that the recall performance of localization module with 0.4m distance metric is 97.5%. The epBRM_V2 successfully regresses the coordinates of 3D bounding boxes

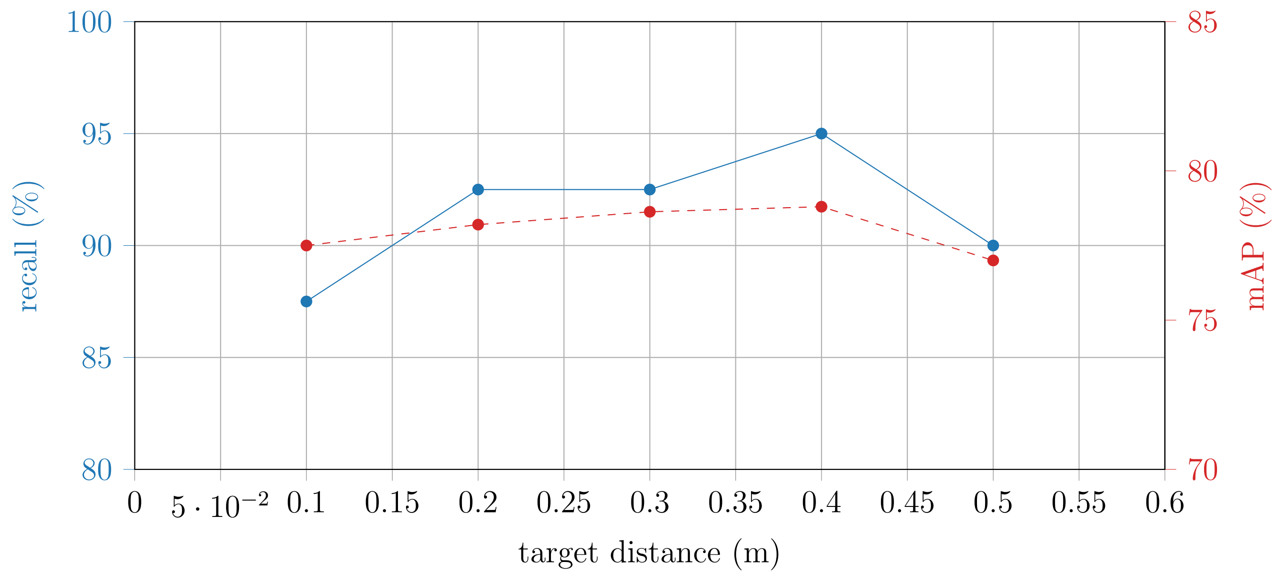


Figure 4.18: The effect of target distance on epBRM_V2

at roughly 95 / 97.5 = 97.4% of locations predicted from localization module.

Additionally, we apply our new data augmentation method for 3D LiDAR point clouds with the best experiment configuration so far. From the tracklet information provided from the KITTI dataset, we generate 350 set of prototype 3D LiDAR point clouds which have at least 5k points in each prototype. Then, we generate the 40 clusters of prototype based on their dimensions. Each training sample is assigned to one of the clusters and used for creating a new training sample. If the newly created sample contains at least 5 points activated inside a box, we add the new sample into the original training set. The total number of new training samples created at this step is 127k which is approximately 12x more than the number of original training samples.

This procedure is processed only once before the training and it takes approximately 2 hours in i7-6700k CPU with NVIDIA Titan X GPU with 12GB memory.

We compare the detection performance with and without mixup data augmentation method. To make fair comparison, other general data augmentation methods such as random scaling, random translation, random rotation, random horizontal flipping, occlusion, random jittering, random drop out of points are applied for both networks.

Figure 4.19 clearly shows that mixup data augmentation improves the overall detection performance compared to the naive data augmentation which are applied to both networks. The recall for both networks are measured as 95%.

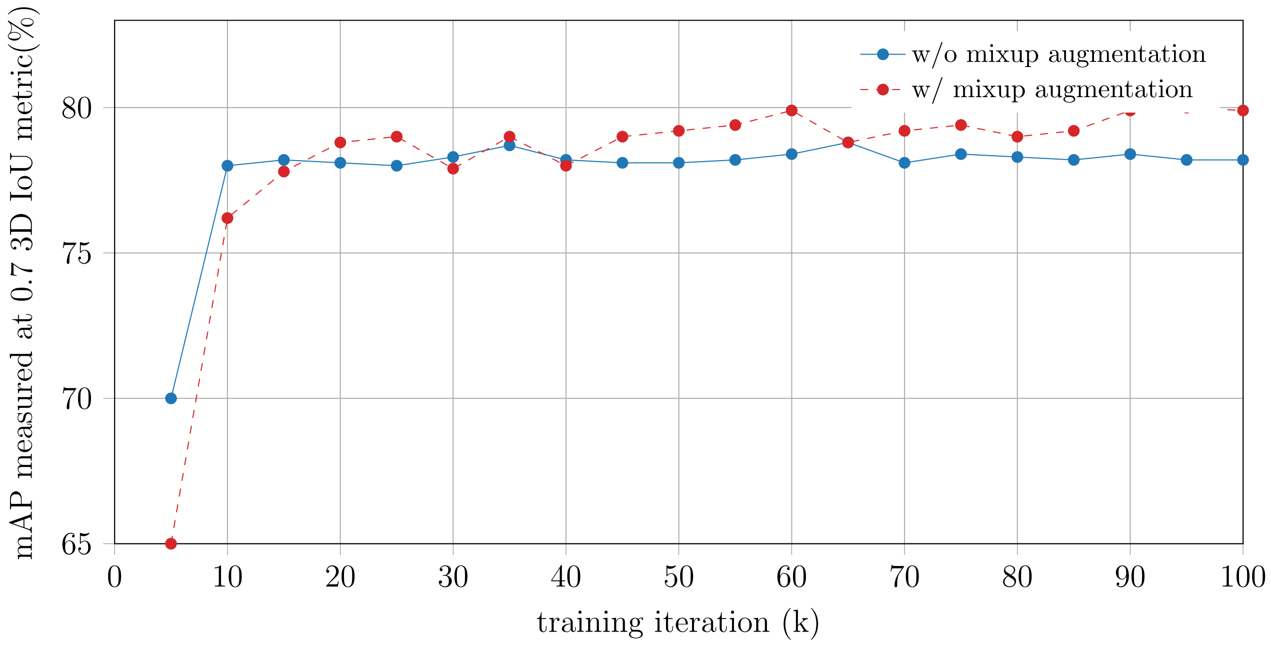


Figure 4.19: A comparison of the detection performance about our proposed data augmentation method for 3D LiDAR point clouds.

4.7 Discussion and future research direction

In this work, we focus on three topics: a) analysis on localization module and effect of NMS_{score} and NMS_{sim}, b) the optimal target distance when we use more sophisticated network structure for box regression task, c) effect of data augmentation method based on mixup of two set of 3D LiDAR point clouds.

First, throughout the extensive experiments, we explored various components that affect the localization performance and got the best parameters for the localization network. The analysis on localization performance showed that 97.5% of ground truth locations can be detected with 0.4m accuracy by using the localization module.

Second, by building a more sophisticated network structure than epBRM_V1, a representation power of epBRM_V2 was increased. Thus, the epBRM_V2 could also refine the objects whose initial location predictions are far from the ground truth. As more diverse predictions could be further refined precisely by epBRM_V2, the recall and mAP of overall detection performance was significantly improved.

Finally, we develop a novel data augmentation method based on mixup of 3D LiDAR point clouds. This augmentation method uses tracklet information which is prevalent in most public datasets for 3D object detection. The analysis on mixup data augmentation shows that it improves the overall detection performance by mAP of 0.5∼1.0%.

The proposed data augmentation method which uses the tracklet information is hard to be applied on deformable objects due to motion of the objects. For example, even a tracklet information of the pedestrian is given, pose of the pedestrian such as hand position and/or head position can be different at each time stamp. Therefore, it makes difficult to build a prototype 3D LiDAR point clouds which is a dense set of point clouds with a clear shape of the object. We think this implies the future research topic.

Chapter 5

Conclusions and Future Works

In this dissertation, we mainly concerned with various challenges in 3D object detection for self-driving vehicles. The topics we have discussed lie in the field of sensor-fusion based 3D object detection, efficient detection pipeline using 3D LiDAR point clouds, practical methods for training a 3D object detection network, design of high-performance detection network using deeply connected layers, and a novel data augmentation for 3D LiDAR point clouds.

In Chapter 2, we discussed a sensor-fusion based 3D object detection system. The proposed approach in Chapter 2, RoarNet, utilized both 2D camera image and 3D LiDAR point clouds to get accurate 3D object detections. Using 2D camera images, a geometrical agreement search and a spatial scattering method were proposed to narrow down the huge 3D search space into smaller one. Then, the proposed approach used PointNet as backbone network for further refinement of a 3D bounding box coordinates. The RoarNet showed one of the best performances in 3D object detection task in KITTI dataset, which is a standard benchmark for the self-driving vehicle detection.

In Chapter 3, we studied a practical method to improve the performance of 3D object detection system. The analysis on the detection result revealed that considerable portion of prediction results which are proximal to the ground truths are false-positive detections due to low quality in box regression task. The proposed approach in Chapter 3, epBRM_V1, was designed for improving the quality of 3D bounding box regression task using 3D LiDAR point clouds as input features. The proposed approach was built with sequence of small networks and was computationally lightweight. A spatial transformation mechanism was adopted into regression pipeline to recursively simplify 3D bounding box regression task. The epBRM_V1 took less than 1 hour of training time and only 12ms of additional latency to improve the performance of standard detection methods to the state-of-the-art 3D object detection methods.

In Chapter 4, we designed a high performance 3D object detection network based on in-depth analysis on the effect of localization network, a new architecture for box regression network, and a novel data augmentation method for 3D LiDAR point clouds processing. The proposed approach, epBRM_V2, re-designed a localization pipeline by using a proximity-based non-maximum suppression operation. We proposed a new metric for evaluating a localization module of 3D object detection pipeline. Then, a new network architecture for box regression module which was built with deeper and skip-connected layers improved the 3D object detection performance by large margin. Lastly, a new data augmentation method for 3D LiDAR point clouds processing was proposed for further improvement of the 3D object detection pipeline.

In the remainder of this chapter, we suggest several promising research topics in 3D object detection systems for self-driving vehicles. We mainly discuss 3D object detection systems for understanding the intention of objects, abnormality detection to guarantee safe self-driving systems, and continual learning method for 3D object detection systems. These topics reflect our perspective on 3D object detection system to build safe self-driving systems.

Understanding the intention of objects

In most standard self-driving systems, the 3D object detection results are mainly consumed by prediction module which is responsible for predicting the future motion of dynamic agents. Since each on-road agent continuously interacts with other agents, it is crucial to understand the intention of each object for predicting their future motion.

Multi-frames 3D object detection becomes an active research topic for forecasting a future trajectory. As a part of self-driving systems, a 3D object detection result can be used for predicting the future trajectory of on-road agents. Since each on-road agent tends to keep their motion at least for a short duration of time, detection results from several temporally continuous frames helps to predict what will happen in a few second. Recently, many researchers are exploring various methods to build an unified pipeline for both detection and prediction modules together as an end-to-end manner so that features representation can be learned jointly for detection and prediction.

Human pose estimation can be used for predicting the intention of pedestrians more accurately. While current 3D object detection systems aim at predicting the tight 3D bounding boxes around the objects, it is not enough to recognize the intention of objects. Human poses which are predicted from the detection module can be supplemental information to understand the intention of objects. To drive safely urban roads which are crowded with a number of pedestrians, we believe that a human pose estimation will play a crucial role for whole autonomy systems.

Abnormality detection

To guarantee a safety of self-driving vehicles, the autonomy systems must be able to react properly when they encounter abnormal circumstances. For 3D object detection systems, the abnormality may include a) encountering an object in a long-tailed distribution, b) occasional sensor failure.

An object in a long-tailed distribution refers to the object which are rarely seen during the normal driving situation. A debris and/or pot holes are examples of object in a long-tailed distribution. Not only the static objects, but also dynamic objects such as pedestrians and animals walking on a freeway can be another example of objects in a long-tailed distribution. Since most deep learning based 3D object detection systems are trained with the dataset where the majority of distribution is collected from normal driving situation, it is challenging to train a detection network for rarely seen objects. However, considering that the most accidents are caused when a driver encounters abnormal situation, the 3D object detection systems must be able to deal with an rarely experienced situation.

Sensor failure means that when one of sensors in detection system fails to operate properly. General on-board camera vision systems for self-driving vehicles collect images from more than 5 or 6 camera sensors every 30ms. Each image from different camera sensor is processed independently. Therefore, failure in one camera sensor may cause inconsistency between several camera sensors. Additionally, inconsistency in detection results from different sensor types such as LiDAR, camera and RADAR can lead to severe detection failure. The 3D object detection systems should be able to deal with sensor-failure cases for safe self-driving system.

Continual learning for 3D object detection systems

The 3D object detection systems for self-driving vehicles can be incrementally upgraded or entirely replaced by other systems even after the original system is being used for deployment. Continual learning is the idea of training a network continuously in order to adapt the change of external world environment. The continual learning enables the incremental development of more sophisticated detection systems.

It is often required to add new features to the currently deployed detection models. The new features may include the improvement of detection performance, change of sensors, new training dataset, and/or addition of new attributes to be predicted by the detection models. In such cases, the detection model needs to be trained to adapt such external changes, which requires long training time and heavy computation resources if the network is trained from the scratch. The continual learning method can mitigate the time and computation required for adopting new features into detection models.

Bibliography

[1] David Berthelot et al. “Mixmatch: A holistic approach to semi-supervised learning”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 5049–5059.

[2] Holger Caesar et al. *nuScenes: A multimodal dataset for autonomous driving*. 2019. arXiv: 1903.11027 [cs.LG].

[3] Zhaowei Cai et al. “A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection”. In: *European Conference on Computer Vision*. 2016.

[4] Zhaowei Cai et al. “A unified multi-scale deep convolutional neural network for fast object detection”. In: *European conference on computer vision*. Springer. 2016, pp. 354–370.

[5] Florian Chabot et al. “Deep MANTA: A Coarse-to-Fine Many-Task Network for Joint 2D and 3D Vehicle Analysis from Monocular Image”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017.

[6] Ming-Fang Chang et al. “Argoverse: 3D Tracking and Forecasting With Rich Maps”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[7] Xiaozhi Chen et al. “Multi-View 3D Object Detection Network for Autonomous Driving”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017.

[8] Yilun Chen et al. “Fast point r-cnn”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 9775–9784.

[9] Ekin D Cubuk et al. “Autoaugment: Learning augmentation policies from data”. In: *arXiv preprint arXiv:1805.09501* (2018).

[10] Zhuo Deng and Longin Jan Latecki. “Amodal detection of 3d objects: Inferring 3d bounding boxes from 2d ones in rgb-depth images”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5762–5770.

[11] Martin Engelcke et al. “Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks”. In: *IEEE International Conference on Robotics and Automation*. 2017. arXiv: 1609.06666.

[12] Markus Enzweiler and Dariu M Gavrila. "A multilevel mixture-of-experts framework for pedestrian classification". In: *IEEE Transactions on Image Processing* 20.10 (2011), pp. 2967–2979.

[13] Di Feng et al. "Deep Multi-modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges". In: *arXiv preprint arXiv:1902.07830* (2019).

[14] Di Feng et al. "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges". In: *IEEE Transactions on Intelligent Transportation Systems* (2020).

[15] A Geiger et al. "Vision Meets Robotics: The KITTI Dataset". en. In: *The International Journal of Robotics Research* 32.11 (Sept. 2013), pp. 1231–1237. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364913491297.

[16] Ross Girshick. "Fast R-CNN". In: *IEEE International Conference on Computer Vision.* 2015. DOI: 10.1109/ICCV.2015.169.

[17] Alejandro González et al. "On-board object detection: Multicue, multimodal, and multiview random forest of local experts". In: *IEEE transactions on cybernetics* 47.11 (2017), pp. 3980–3990.

[18] Benjamin Graham. "Spatially-sparse convolutional neural networks". In: *arXiv preprint arXiv:1409.6070* (2014).

[19] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[20] Tong He et al. "Bag of tricks for image classification with convolutional neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2019, pp. 558–567.

[21] Hiroshi Inoue. "Data augmentation by pairing samples for images classification". In: *arXiv preprint arXiv:1801.02929* (2018).

[22] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. "Spatial transformer networks". In: *Advances in neural information processing systems.* 2015, pp. 2017–2025.

[23] Lu Jiang et al. "Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels". In: *International Conference on Machine Learning.* 2018, pp. 2304–2313.

[24] Jason Ku et al. "Joint 3D Proposal Generation and Object Detection from View Aggregation". In: *arXiv:1712.02294 [cs]* (Dec. 2017). arXiv: 1712.02294 [cs].

[25] Jean Lahoud and Bernard Ghanem. "2D-Driven 3D Object Detection in RGB-D Images". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV).* Oct. 2017.

[26] Alex H. Lang et al. "PointPillars: Fast Encoders for Object Detection from Point Clouds". In: *arXiv:1812.05784 [cs, stat]* (Dec. 2018). arXiv: 1812.05784 [cs, stat].

[27] Bo Li. “3d fully convolutional network for vehicle detection in point cloud”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 1513–1518.

[28] Bo Li, Tianlei Zhang, and Tian Xia. “Vehicle Detection from 3D Lidar Using Fully Convolutional Network”. In: *Robotics: Science and Systems*. 2016. arXiv: 1608.07916.

[29] Bo Li, Tianlei Zhang, and Tian Xia. “Vehicle detection from 3d lidar using fully convolutional network”. In: *arXiv preprint arXiv:1608.07916* (2016).

[30] Yangyan Li et al. “Pointcnn: Convolution on x-transformed points”. In: *Advances in neural information processing systems*. 2018, pp. 820–830.

[31] Ming Liang et al. “Deep Continuous Fusion for Multi-Sensor 3D Object Detection”. en. In: *European Conference on Computer Vision*. 2018, p. 16.

[32] Ming Liang et al. “Deep continuous fusion for multi-sensor 3d object detection”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 641–656.

[33] Tsung-Yi Lin et al. “Focal Loss for Dense Object Detection”. In: *CoRR* abs/1708.02002 (2017). arXiv: 1708.02002. URL: http://arxiv.org/abs/1708.02002.

[34] Wenjie Luo, Bin Yang, and Raquel Urtasun. “Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting With a Single Convolutional Net”. en. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018.

[35] Sivabalan Manivasagam et al. “LiDARsim: Realistic LiDAR Simulation by Leveraging the Real World”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 11167–11176.

[36] Arsalan Mousavian et al. “3D Bounding Box Estimation Using Deep Learning and Geometry”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017.

[37] Jiquan Ngiam et al. *StarNet: Targeted Computation for Object Detection in Point Clouds*. 2019. arXiv: 1908.11069 [cs.CV].

[38] Charles R. Qi et al. “Frustum PointNets for 3D Object Detection from RGB-D Data”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2018.

[39] Charles R. Qi et al. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017. arXiv: 1612.00593.

[40] Charles R. Qi et al. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. In: *Neural Information Processing Systems*. 2017.

[41] Joseph Redmon and Ali Farhadi. “YOLO9000: better, faster, stronger”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2017.

[42] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2016, pp. 779–788.

[43] Jimmy Ren et al. "Accurate single stage detector using recurrent rolling convolution". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 5420–5428.

[44] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Neural Information Processing Systems*. 2015, pp. 1–10. DOI: 10.1016/j.nima.2015.05.028.

[45] Zhile Ren and Erik B Sudderth. "3d object detection with latent support surfaces". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 937–946.

[46] Zhile Ren and Erik B Sudderth. "Three-dimensional object detection and layout prediction using clouds of oriented gradients". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1525–1533.

[47] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. "PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud". In: *arXiv preprint arXiv:1812.04244* (2018).

[48] Shaoshuai Shi et al. "From Points to Parts: 3D Object Detection from Point Cloud with Part-aware and Part-aggregation Network". In: *arXiv preprint arXiv:1907.03670* (2019).

[49] Kiwoo Shin, Youngwook Paul Kwon, and Masayoshi Tomizuka. "Roarnet: A robust 3d object detection based on region approximation refinement". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE. 2019, pp. 2510–2515.

[50] Kiwoo Shin and Masayoshi Tomizuka. *Improving a Quality of 3D Object Detection by end-point box regression module*. IEEE, 2020.

[51] Martin Simon et al. "Complex-YOLO: Real-Time 3D Object Detection on Point Clouds". In: *European Conference on Computer Vision*. Mar. 2018. arXiv: 1803.06199.

[52] Andrea Simonelli et al. "Disentangling monocular 3d object detection". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2019, pp. 1991–1999.

[53] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. "Sun rgb-d: A rgb-d scene understanding benchmark suite". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 567–576.

[54] Shuran Song and Jianxiong Xiao. "Deep sliding shapes for amodal 3d object detection in rgb-d images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 808–816.

[55] Pei Sun et al. *Scalability in Perception for Autonomous Driving: Waymo Open Dataset*. 2019. arXiv: 1912.04838 [cs.CV].

[56] Daiki Tanaka et al. "Joint optimization framework for learning with noisy labels". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2018, pp. 5552–5560.

[57] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. "Between-class learning for image classification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2018, pp. 5486–5494.

[58] Vikas Verma et al. "Interpolation consistency training for semi-supervised learning". In: *arXiv preprint arXiv:1903.03825* (2019).

[59] Vikas Verma et al. "Manifold mixup: Better representations by interpolating hidden states". In: *International Conference on Machine Learning.* 2019, pp. 6438–6447.

[60] Sourabh Vora et al. "PointPainting: Sequential Fusion for 3D Object Detection". In: *arXiv preprint arXiv:1911.10150* (2019).

[61] Zhixin Wang and Kui Jia. "Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection". In: *arXiv preprint arXiv:1903.01864* (2019).

[62] Yu Xiang et al. "Subcategory-aware convolutional neural networks for object proposals and detection". In: *2017 IEEE winter conference on applications of computer vision (WACV).* IEEE. 2017, pp. 924–933.

[63] Shuran Song Jianxiong Xiao. "Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images". In: ().

[64] Qizhe Xie et al. "Unsupervised data augmentation for consistency training". In: *arXiv preprint arXiv:1904.12848* (2019).

[65] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. "Pointfusion: Deep sensor fusion for 3d bounding box estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2018, pp. 244–253.

[66] Bin Yang, Ming Liang, and Raquel Urtasun. "Hdnet: Exploiting hd maps for 3d object detection". In: *Conference on Robot Learning.* 2018, pp. 146–155.

[67] Bin Yang, Wenjie Luo, and Raquel Urtasun. "PIXOR: Real-Time 3D Object Detection From Point Clouds". en. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition.* 2018, p. 9.

[68] Zetong Yang et al. "Ipod: Intensive point-based object detector for point cloud". In: *arXiv preprint arXiv:1812.05276* (2018).

[69] Zetong Yang et al. "Std: Sparse-to-dense 3d object detector for point cloud". In: *Proceedings of the IEEE International Conference on Computer Vision.* 2019, pp. 1951–1960.

[70] Sangdoo Yun et al. "Cutmix: Regularization strategy to train strong classifiers with localizable features". In: *Proceedings of the IEEE International Conference on Computer Vision.* 2019, pp. 6023–6032.

[71] Dominic Zeng Wang and Ingmar Posner. “Voting for Voting in Online Point Cloud Object Detection”. en. In: *Robotics: Science and Systems XI*. Robotics: Science and Systems Foundation, July 2015. ISBN: 978-0-9923747-1-6. DOI: 10.15607/RSS.2015.XI.035.

[72] Hongyi Zhang et al. *mixup: Beyond Empirical Risk Minimization*. 2018.

[73] Yin Zhou and Oncel Tuzel. “VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection”. In: arXiv: 1711.06396. Nov. 2017.