공학박사학위논문

실시간 자율주행 인지 시스템을 위한 신경 네트워크와 군집화 기반 미학습 물체 감지기 통합

Integration of Clustering-based Unlearned Object Detection and Deep Neural Network for Real-time Perception in Autonomous Driving System

2020년 8월

서울대학교 대학원

기계항공공학부

유 정 겸

Abstract

Integration of Clustering-based Unlearned Object Detection and Deep Neural Network for Real-time Perception in Autonomous Driving System

Jungkyum Yu

School of Mechanical and Aerospace Engineering

The Graduate School

Seoul National University

In recent few years, the interest in automotive researches on autonomous driving system has been grown up due to advances in sensing technologies and computer science. In the development of autonomous driving system, knowledge about the subject vehicle's surroundings is the most essential function for safe and reliable driving. When it comes to making decisions and planning driving scenarios, to know the location and movements of surrounding objects and to distinguish whether an object is a car or pedestrian give valuable information to the autonomous driving system. In the autonomous driving system, various sensors are used to understand the surrounding environment. Since LiDAR gives the distance information of surround objects, it has been the one of the most commonly used sensors in the development of perception system.

Despite achievement of the deep neural network research field, its application and research trends on 3D object detection using LiDAR point cloud tend to pursue higher accuracy without considering a practical application. A deep neural-network-based perception module heavily depends on the training dataset, but it is impossible to cover all the possibilities and corner cases. To apply the perception module in actual driving, it needs to detect unknown objects and unlearned objects, which may face on the road. To cope with these problems, in this dissertation, a perception module using LiDAR point cloud is proposed, and its performance is validated via real vehicle test. The whole framework is composed of three stages : stage-1 for the ground estimation playing as a mask for point filtering which are considered as non-ground and stage-2 for feature extraction and object detection, and stage-3 for object tracking. In the first stage, to cope with the methodological limit of supervised learning that only finds learned object, we divide a point cloud into equally spaced 3D voxels the point cloud and extract non-ground points and cluster the points to detect unknown objects. In the second stage, the voxelization is utilized to learn the characteristics of point clouds organized in vertical columns. The trained network can distinguish the object through the extracted features from point clouds. In non-maximum suppression process, we sort the predictions according to IoU between prediction and polygon to select a prediction close to the actual heading angle of the object. The last stage presents a 3D multiple object tracking solution. Through Kalman filter, the learned and unlearned object's next movement is predicted and this prediction updated by measurement detection.

Through this process, the proposed object detector complements the detector based on supervised learning by detecting the unlearned object as an unknown object through non-ground point extraction. Recent researches on object detection for autonomous driving have been actively conducted, but recent works tend to focus more on the recognition of the objects at every single frame and developing accurate system. To obtain a real-time performance, this paper focuses on more practical aspects by propose a performance index considering detection priority and detection continuity. The performance of the proposed algorithm has been investigated via real-time vehicle test.

Keywords: Autonomous Driving (AD), LiDAR Sensor, Artificial Neural Network, Object Detection, Multi Object Tracking (MOT)

Student Number: 2013-20687

List of Figures

Contents

Chapter 1 Introduction

1.1. Background and Motivation

Autonomous driving has been an aspiring research topic for many years because it is one of the core technologies that might redefine our society and life. Once autonomous driving is realized in our real life, the infrastructure of public transportation and our life environment will change. Furthermore, autonomous driving is expected to reduce traffic accidents caused by driver errors, as well as to save driver's time. The achievement in researches about autonomous driving has been facilitated mostly by improvements and lowered cost of computer hardware capable of running the complex algorithms needed. Autonomous driving also needs various sensors and a large amount of data calculations. Unlike the early days of autonomous driving research, where sensor manufacturers were less common, the appearance of many new sensor manufacturers and the lowered price of sensors make it more possible to use various sensors to develop an autonomous vehicle. A large amount of sensor data is being utilized to localize and ultimately guide the vehicle through its environment and this requires fast processing and considerable computational power.

Many automotive companies and related industries have been struggling to develop an autonomous vehicle or self-driving related technologies. Some of the most advanced self-driving vehicles in existence today are in the fourth stage according to The Society of Automotive Engineering (SAE). The fourth stage is defined as “Fully Automated Driving: the vehicle drives independently most of the time but the driver must remain able to drive.”

This means that they are not perfect and they are fully autonomous but just under certain conditions like on the highway.

To reach level 5 autonomy, perception and related technologies play the most important roles in not only the safety of autonomous vehicles but in their ability to account for unexpected variables while driving - a key milestone for autonomous vehicles to achieve.

Perception of autonomous drive systems is the most essential function for reliable driving because it is the task of identifying the surrounding environment and understanding information that is related to driving safety. Various methods have been attempted to develop a perception module for autonomous driving in various environments, and recently, methods using Artificial Neural Networks (ANNs) have been actively attempted.

Artificial Neural Networks (ANNs) are a type of computer model that simulates the behavior of biological neural networks such as the human brain. Over the past few years, the researches on Artificial Neural Network have been infused by drastic improvements and lowered the cost of computer hardware. This results in a notable success in object detection and classification on RGB images. [Liu’16, Girshick’15, Ren’15, Redmon’16, Redmon’17]

This has been proved that the applications of the neural network on a regression problem mostly shows higher performance than the past researches, which sets a specific model. Inspired by the positive results of 2D ANN implementations, research groups and companies have looked towards applying the more successful variants of neural networks to 3D data such as RGB-D images, CAD models, and 3D point cloud scans, and also to the autonomous driving system. The main obstacle in the application of neural networks on autonomous driving is handling 3D data. Because previous researches and almost existing networks are appropriately designed to 2D image data, so it is required to take different approaches and method of handle the data and to construct model structure properly to 3D data. Several methods such as projecting 3D data to 2D [Chen'17, Ku'18, Li'16] and voxelization the 3D data into occupancy grids stored as binary matrices [Chen'17, Ku'18, Simon'18, Yang'18] have been tried to solve the problems and they have been shown to have relatively high success in comparison to the existing methods.

However, most studies are not approaching from a practical point of view by concentrating to receive high Average Precision (AP) of their object detector for KITTI dataset. Although the actual autonomous driving is a continuous situation in which each frame is connected, the network is learned the dataset without any association between continuous scenes. So even if it finds the learned object in this scene, it may miss it in the next scene. In order to respond to such situations, object detection, as well as object tracking, must be performed together.

Also, we may encounter many objects that the detector has never learned. For example, many buses are encountered on Korean roads while there are no bus annotations in the KITTI dataset. It is difficult to learn all kinds of objects that can be encountered on the road, and it is necessary to cope with the methodological limitations of supervised learning that can find only the learning objects. Therefore, this dissertation focuses on developing a perception module that tracks both of unlearned objects detection from object clustering and the results of neural network-based object detectors.

1.2. Overview and Previous Researches

Even though the application of Artificial Neural Network on a 3D point cloud for the autonomous driving systems is still a fresh research topic, a number of researches and thesis work have been published over the last few years. In this chapter, the characteristics and the difficult aspects to processing of 3D point cloud data will be discussed, as well as the research trends related to object detection using 3D point cloud. The point cloud is data with 3D spatial information, and there have been various methods exist depending on how to process this data. In addition, these researches are divided into a one-stage method or two-stage method according to the network structure.

Properties of point cloud. As mentioned earlier, autonomous driving system usually uses multiple types of sensors to ensure reliability. The most commonly used sensors are radio detection and ranging (RADAR), light detection and ranging (LiDAR), and ultrasonic sensors. As shown in the Figure 1.1, among those sensors, LiDAR can directly provide a precise 3D information of surroundings. There have been many researches trying to reconstruct 3D environment information or to estimate the depth information based on 2D images. Recent works related to these topics have been achieved significant improvements with the development of deep learning based methods. Despite these achievements, the results of these studies are still not always precise or reliable. On the aspect of this, LiDAR is useful because it provides 3D environment information through direct physical sensing. Most companies leading the autonomous driving market and researches rely on LiDAR to perceive the surroundings and build a reliable autonomous vehicle [Meyer'19].

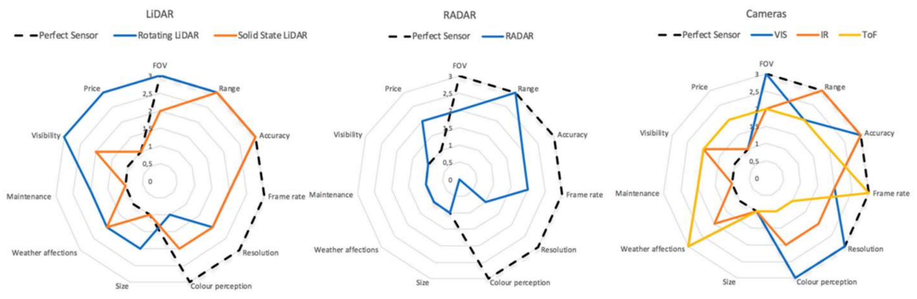


Figure 1.1. Comparison of the features of the different sensors used in environment perception systems [Rosique'19]

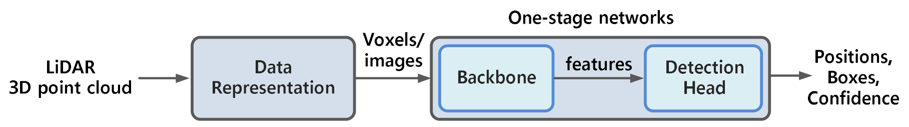
3D point cloud is represented as a set, which ignores any order of 3D points. Let $S = \{(\mathbf{p}_i, \mathbf{a}_i)\}_{i=1}^{N}$ be a set of 3D point cloud having N points. The set is consists of N elements and i th element $\mathbf{p}_i$ represents the 3D coordinate of the i th point, which can be defined as $\mathbf{p}_i = [x_i, y_i, z_i] \in \mathbb{R}^3$. In addition, $\mathbf{a}_i$ represents other attributes i th point, and it is intensity $\mathbf{a}_i = r_i \in \mathbb{R}$ in general cases. Thus, in general, the i t th point of 3D point cloud has a form of feature $\mathbf{x}_i = (\mathbf{p}_i, \mathbf{a}_i) \in \mathbb{R}^4$.

3D object detection from 2D images. There are several approaches to detect 3D bounding box from 2D images. Utilized the geometric relation between 3D and 2D bounding box to estimate the 3D object position and orientation. Chen et al. leveraged an energy function as a presentation of the 3D geometric information of objects to score the predefined 3D boxes. However, although the performance of object detection on 2D images is already proved and the image has rich information, these works can only generate coarse 3D detection results due to the lack of depth information.

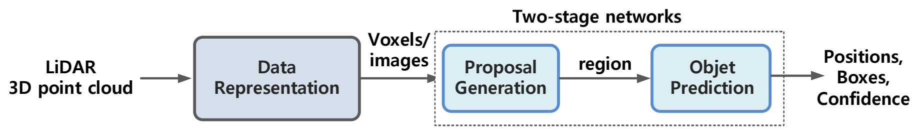
3D object detection from point clouds. Recently, most of 3D detectors have been adopted LiDAR to get depth information. However, since the LiDAR point cloud has a form of 3D spatial information unlike 2D image information, it is difficult to directly apply the method used in the vision area. Some studies [Chen'17, Ku'18, Simon'18, Yang'18] have adopted the voxelization method to process this three-dimension spatial information into the form of a tensor that is easy to apply to the convolution network. Because object detection with point clouds is an essentially three-dimensional problem, it is intuitive to deploy a 3D convolutional network for object detection. Despite the intuitive structure, the 3D convolutional method is relatively more computational than 2D convolution. Engelcke et al. [Engelcke'17] require 0.5s for inference on a single point cloud. To overcome the computational time problem, recent researches adopted the method to project the 3D point cloud onto the ground plane [Chen'17, Ku'18, Yang'18, Liang'18, Yang'18, Lang'19], which is called a bird's eye view (BEV). To form a pseudo-image that can be processed by the 2D Convolution network method, the bird's eye view projection utilizes the voxelization to allocate the point clouds into vertical columns encoded as fixed-size. Some researches including MV3D [Chen'17], PIXOR [Yang'18] and Complex YOLO [Simon'18], AVOD [Ku'18] have leveraged the bird's eye view method and accomplished notable results. MV3D and AVOD fuse the point cloud features with 2D image features and these detectors adopted two-stage detection pipelines.

Meanwhile, Qi et al. [Qi'17] proposed a simple architecture, PointNet, which can learn the feature characteristics directly from point cloud without voxelization. VoxelNet [Zhou'18] is one of the first methods accomplished notable performance by deploying PointNets in their object detection architecture. VoxelNet applied a simple version of PointNet to voxels, then they are processed by a sequential 3D convolution layers followed by a 2D CNN backbone and a detection head. Despite end-to-end learning and high performance, VoxelNet has low detection speed requiring 225ms (4.4 Hz) for a single point cloud scene in inference by using 3D CNN. This detection speed is not fast enough to infer in real-time because the conventional LiDAR data update is 0.1ms.

Paradigms of 3D object detection based on point cloud. There are usually two frameworks of 3D object detection: one-stage detection and two-stage detection; These frameworks are shown in Figure 1.2.



(a) One-stage detection framework



(b) Two-stage detection framework

Figure 1.2. The network frameworks of the one-stage detection and the two-stage detection. The one-stage detection directly estimates object's position and bounding boxes. The two-stage detection first proposes coarse regions where object is supposed to be included and then estimates the object's information.

The one-stage detection directly estimates object's position and bounding boxes. The two-stage detection first proposes coarse regions where object is supposed to be included and then estimates the object's position and boundary boxes. The detector based on the one-stage framework satisfies following form.

$$\{\mathbf{o}_i\}_{i=1}^{O} = h(S) \tag{1.1}$$

Where $\mathbf{o}_i = [\mathbf{y}_i, \mathbf{b}_i, \mathbf{c}_i]$ is the i th object in the scene, with $\mathbf{y}_i$ the object's label, such as car, cyclist and pedestrian, and the positions and $\mathbf{b}_i$ the bounding box information, and $\mathbf{c}_i$ the confidence. A neural network based on one-stage method consist of a backbone, which extracts deep spatial features, and a detection head, which estimates outputs. One-stage method has relatively simpler architecture than two-stage method, so one-stage detection tends to be faster and enjoys a high recall, while the two-stage method tends to achieve high recall. For this reason, some researches [Zhou'18, Yan,'18, Lang'19, Yang,'18, Simon,'18] are following the one-stage method for achieve simpler and faster model. Contrary to the one-stage method, the two-stage approach implements the detection in two stages, which can be defined as follows:

$$\begin{aligned} \{r_i\}_{i=1}^{R} &= h_1(S) \\ \{\mathbf{o}_i\}_{i=1}^{O} &= h_2\left(S, \{r_i\}_{i=1}^{R}\right) \end{aligned} \tag{1.2}$$

Where r_i is a th proposed region where the object is supposed to be included in the 3D space. As shown above equations, two-stage detection process is consists of two steps, which are region proposal stage and object detection stage. At the detection stage, the positions and bounding boxes are estimated based on the proposals from earlier stage. A two-stage method, which are Patches [Lehner,'19], PointRCNN [Shi'19] and FrsutumNet PointNet [Qi'18], tend to accomplish higher detection accuracy. Frustum PointNet detects object on an image plane and generates a frustum from it into 3D space, then it uses PointNet segment and classifies the point cloud in the frustum. Despite their structural advantages to accomplish high accuracy, they have low detection speed because they split the detection process into two steps as region proposal and regression.

3D Multi-Object Tracking. Multi-object tracking (MOT) is an essential function for many applications not only autonomous driving. Due to the drastic advance in object detection performance on 2D image, they can also achieve the much progress on 2D MOT. Although the accuracy of object tracking on the 2D image has been improved, its application on 3D motion prediction and tracking has not been conducted sufficiently and some works require more complex architecture and computational cost. Several methods have been tried to tracking objects in an environment and these methods can be divided into two categories: model-based tracking and model-free tracking

In the model-based tracking method, the most widely used approaches are using Kalman filters or a variant methods with an appropriate pre-defined model. When the task is to track a two or four-wheeled vehicle, a standard bicycle model is used to predict object next state. However, model-based method needs an appropriate pre-defined model, and if the assumption about the model or its parameters are different from real values, the prediction based on the method will result in errors. In addition, some object's behaviors are very difficult to associate with a specific model.

On the Contrary, model-free approaches need no specific model to predict the object's next motion. The proposed model by [Tipaldi'16] uses random sample consensus (RANSAC) to estimate motion models for both the sensor and the dynamic objects. Bahraini et al. [Bahraini'18] proposes the Multilevel Random Sample Consensus (ML-RANSAC) algorithm that enhances the speed of

RANSAC. The RANSAC algorithm comprises two repeated steps. The first step is the generation of hypotheses. A randomly minimal sample subset is selected from the input data to form a set of hypotheses. The second step is hypothesis validation, which verifies if the data is consistent with the estimated model, which was obtained from the first step. The hypotheses that lie outside of the confidence interval of the estimated model will be removed.

While model-free object tracking approaches usually serve faster tracking results compare to the model-based approaches, it serves little accurate tracking result in heading angle since the model-free approach does not consider dynamic and kinematic constraints of objects.

Xinshuo et al. [Xinshuo'19] proposed a simple object tracking method separable from the object detector. The proposed tracking method takes object detection results and associate it with the prediction result predicted by Kalman filter. This method utilizes Hungarian algorithm to associate the current objects and previous objects. The newly appeared and disappeared objects are controlled by birth and death memory. This paper proposes a network having high detection performance and this network's detection performance between frames is supported by the object tracking module. Furthermore, we secure the real-time performance of this network by rising its detection frequency.

1.3. Thesis Objectives

This dissertation focuses on developing an object detection and tracking algorithm for various objects (learned and unlearned) that may encounter while driving.

The perception module corresponds to the eyes of autonomous vehicles. The autonomous vehicle should recognize the surrounding environment and obstacles by various sensors, and the fault of the perception module directly leads to an autonomous vehicle accident. Despite there have been many types of research using neural network in recent years due to drastic enhance of the neural network, they only focus only to single frame detection without object tracking and they have been less researched the perception algorithm in a practical aspect. Most of the object detectors developed based on the neural network are using a supervised learning method. Although they show relatively high performance, there is still a methodological limitation that they only detect learned objects. In actual driving, there are cases that the detector encounters an object it did not learn or that the detector misses an object even though it learned. It is very important to fail to recognize or miss these objects because they lead to accidents. Therefore, it is necessary to detect even if the object has not to be learned or detected.

To complement the methodological limitations of the current supervised learning-based detector and secure practically applicable perception ability, it is necessary to find the unknown object (the objects that detector never learned), and it is needed to track the discovered objects and the unknown objects to ensure the detection continuity.

In the remainder of this thesis, we will provide an object detection using neural network method, point clustering method, and multi-object tracker developed in practical aspect and the experimental results which show the effectiveness of the proposed perception algorithm. The effectiveness of the proposed automated driving perception algorithm is evaluated via vehicle tests with 32 channel LiDAR. In addition, we propose a performance index considering detection priority and detection continuity. The test result has been evaluated via the performance index we proposed in this paper.

1.4. Thesis Outline

This dissertation is structured in the following manner. An overall architecture of the proposed perception module using a LiDAR point cloud is described in Chapter 2. In Chapter 3, the object detector based on a neural network method is introduced. The object detector utilizes the point cloud raw data and non-ground mask, which is generated from the non-ground point clustering process introduced in Chapter 4. In Chapter 4, the non-ground point clustering process is described. By using the result from the process, we can find the obstacles or objects that vehicles should avoid. In Chapter 5, the object tracking method is introduced. This predicts the object's motion based on Kalman filter, then associates the predictions from previous detections and current detection by using Hungarian algorithm. In Chapter 6, the performance of the proposed object detector is compared with other networks.

In Chapter 7, Average precision (AP), which is broadly used in object detection research area, is introduced and modified performance index which considering detection priority and continuity is proposed.

Chapter 8 shows the experiment results for the evaluation of the performance of the proposed perception algorithm. Then future works and conclusion that describes the summary and contribution of the proposed perception algorithm are presented in Chapter 7.

Chapter 2 Overview of a Perception in Automated Driving

From a considerable amount of recent literature, automated driving technology has the potential to reduce the environmental impact of driving, reduce traffic jams, and increase the safety of motor vehicle travel. However, in advancing self-driving technology, the ability to recognize the surrounding environment and obstacles is an essential task to be achieved. Because perception module corresponds to the eyes of autonomous vehicles, to secure reliability of automated vehicle technology requires highly accurate perception algorithm to distinguish the nearby object to avoid the crush and achieve right path planning, control, and decision in complex driving conditions. Perception in autonomous driving has made great progress through going down of sensors cost, improving computing technology, and remarkable achievements in the neural network field. Recently, through the achievement of neural networks in the field of vision, many studies have been conducted on object detection using Lidar point clouds. Indeed, some recent studies have been trained with the KITTI dataset and have shown remarkable achievements.

However, these researches have an unpractical aspect to actual driving, and as aforementioned, some issues are considered. First, a methodological limitation of supervised learning. Most of the recent researches related to object detectors based on neural networks has been developed based on supervised learning techniques. The supervised learning based detector can only find the kind of object the detector has learned. In the case of KITTI dataset, there is no annotation of 'bus', so the object detector trained by KITTI dataset cannot find a bus in actual driving. Secondly, most of the recent researches only focuses only on single frame detection without object tracking and they have been less researched the perception algorithm in time continuous aspect. This frame-by-frame detection has a problem related to the detector performance evaluation, and the current object detection performance evaluation based on AP, which do not consider detection priority and continuity, cannot show reasonable performance. The perception architecture of the algorithm proposed in this dissertation to solve the aforementioned problems is outlined in Figure 2.1.

In the remainder of this paper, we will provide an overview of the overall architecture of the proposed perception algorithm using point cloud and the experimental results which show the effectiveness of the proposed algorithm.

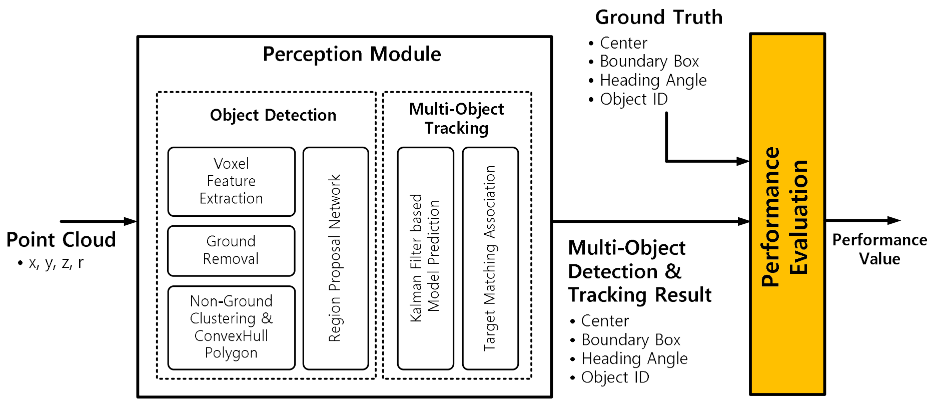


Figure 2.1. Overview of the proposed perception algorithm. The proposed algorithm consists of two parts: object detection and multi-object tracking.

Chapter 3 Object Detector

Autonomous driving costs a significant computational load because it is a complex total system, which its sub-modules such as localization, planning, and control operate complementarily each other. Besides, most LiDAR products basically update point cloud data at speeds of 0.1Hz or higher, so it is said that the detector does not guarantee real-time performance if the update frequency of detection result is slower than point cloud update frequency.

Therefore, the speed of the perception module is directly related to whether the algorithm is real-time applicable. The LiDAR product used in the KITTI dataset is Velodyne’s HDL-64E and this LiDAR is mounted on the roof of the test vehicle to cover 360 degrees around the test vehicle. Even though all the point cloud around the test vehicle are served, the KITTI dataset offers ground truth's annotation only limited to objects within the front camera’s field of view. This means that the inference speed on KITTI benchmark, which many pieces of research are showing, was recorded using less than half of the total point cloud. Naturally, the calculation burden of point clouds will increase as the number of point clouds increases.

For this reason, it is reasonable to expect that the actual inference rate of the networks on KITTI benchmark will be reduced if they are applied in a real-time environment, which is the case that all the point cloud are used or the perception algorithm operates with other self-driving modules like planning, localization, control.

Most state-of the-the-art object detection methods can be categorized into either (1) a two-stage framework or (2) a one-stage framework. Essentially, the two-stage methods are divided into two steps. It generates a set of sparse candidate bounding box proposals and calculates an objectness score for the content of each box in the first step and then conduct further regression and classification in the second step. In contrast, a one-stage method [Redmon,'15, Liu,'16, Redmon,'16, Lin,'17, Redmon,'18, Iandola,'16] performs object location and bound box regression and classification directly by dense sampling from each feature map.

Some recent works showing good performance on KITTI benchmark adopted this two-stage method [Qi'18, Wang'19, Shi'19, Chen'19, Yang'19, Shi'19]. They pre-trained a segmentation network that learns point-wise object scores to generate coarse object proposals. With these proposals, they regressed box size and heading angle, confidence. Because two-stage method divided the detection process completely into (1) providing the balanced proposals and (2) Regression stage which tends to improve the object location and heading angle, they usually show more accurate results than the one-stage method. However, the two-stage method requires a pre-trained model to generate proposals, so this method is not advantageous for end-to-end training, or even though end-to-end is possible, the separated process of two-stage method leads to low detection speed.

For these reasons, networks using the two-stage method have a low inference speed than one-stage method. In practice, networks adopted two-stage method in KITTI benchmark have inference rate about 40ms to 80ms. As mentioned before, the inference rate could be lower in actual real-time application because of other computational burden or LiDAR point cloud ranges. In this dissertation, one-stage method is adopted to secure the detection speed as efficiently as possible for practical aspects.

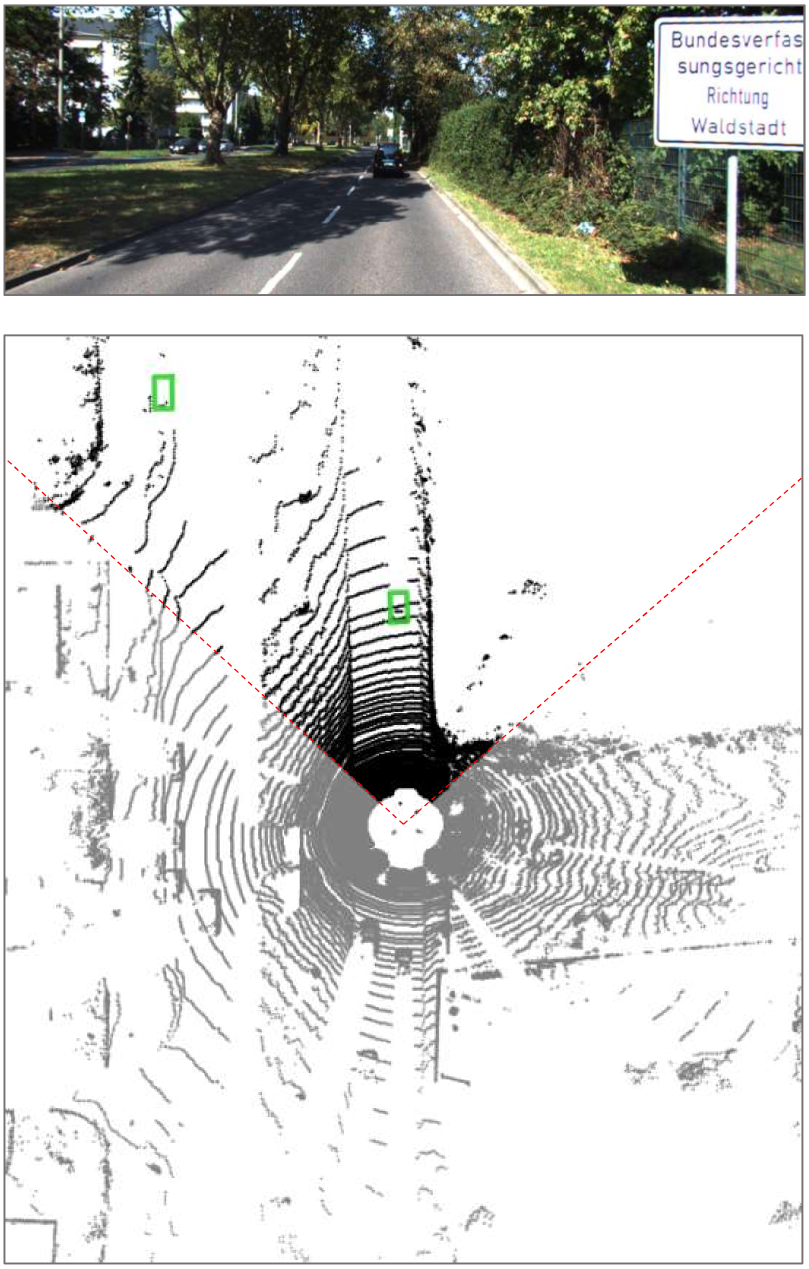


Figure 3.1. Camera view and point cloud of KITTI dataset. Green boxes are ground truth information. Annotation is limited to objects inside camera view

3.1. Voxelization & Feature Extraction

LiDAR point cloud data is 3D spatial information and has information of x, y, z, and intensity. Therefore, unlike a tensor type 2D image data, it is impossible to apply a convolutional network directly. In addition, since there is no order for 3D spatial data, the same result should be shown even when the points order is changed. PointNet [Qi'17] achieved notable result by learning the relationship between the points and classifing them as point-wise directly processing the LiDAR point cloud without converting it to another form. However, if the number of points in LiDAR is large, a lot of calculation is required to process in point units, so it can be seen through Voxelnet [Zhou'18] showed that it is effective to convert the point cloud data into a pseudo-image and apply a convolution network.

To apply a 2D convolutional network method, we first transform the point cloud into a pseudo-image form such as RGB image. As a first step, the voxelization step subdivide the point cloud x, y, z, r into equally spaced voxel in the x-y plane with one z-direction channel as Voxelnet. In this step, like [Lang'19] did, we adopt only one-channel along the z-axis to secure network efficiency. As the pseudo-image has only one-channel, it is easy to apply 2D convolution not 3D convolution, which results in speed efficiency. To obtain point canonical locations in each voxel, we subtracting the arithmetic mean (X,Y,Z) values and augmented the distance to the arithmetic mean of all points and offset from the voxel center position in the voxel. The augmented LiDAR point is 9 dimensional information: $x, y, z, x_c, y_c, z_c, x_p, y_p, z_p, r$. We set the number of the augmented information number as D = 9. According to many researches [Yan'18] we set the voxel size to (d_l = 0.16m, d_w=0.16m, d_h = 4m), d_l , d_l and d_l means width, height and height of voxel respectively. By applying voxelization, we can allocate the points cloud into voxels having much less number than total point numbers as shown in Figure 3.2.

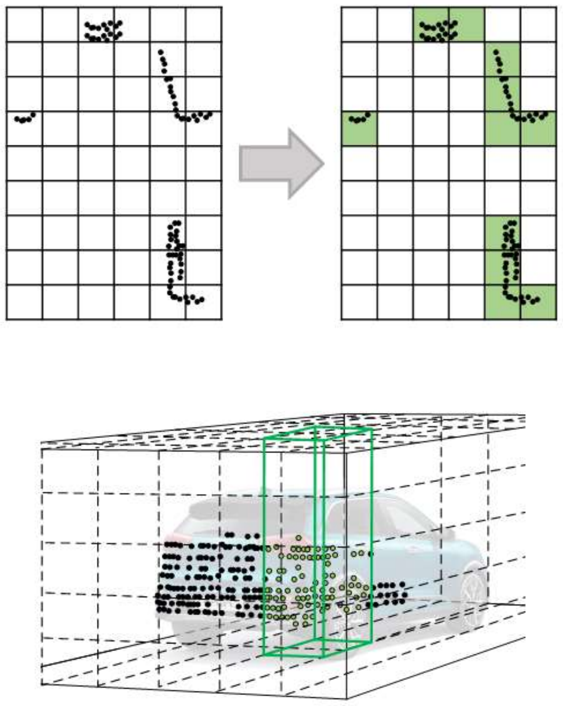


Figure 3.2. Voxelized map of point cloud

Most of the voxels will be empty due to sparsity of the point cloud, and there are only a few points even in the non-empty voxels. We impose a limit both on the number of non-empty voxels per sample (P) and on the number of points per voxel (N) to create a dense tensor of size (D, P, N). If too much number of the point cloud are allocated in a sample or voxel, the points are randomly sampled to fit this tensor. Conversely, zero paddings is applied if a sample or voxel has too little number of the point cloud to populate the tensor.

Many researches [Simon'18, Ali'18, Beltran'18] adopted max height, intensity, density(the number of the points in a voxel or grid) for the network input feature's channel. This selection is intuitive and simple to process. However, it is not easy to learn feature characteristics in the local area of the voxel unit.

Next, we apply a simplified PointNet to each point to extract the feature characteristics of the voxel. PointNet is consists of a linear layer followed by Batch Normalization [Ioffe'15] and ReLU to generate a tensor with the size of (C, P, N). After passes the simplified PointNet, a max operation is applied over the channels to generate a (C, P) sized output feature tensor. If the feature extraction process is ended, the extracted features present values of corresponding voxel channels. We can get pseudo-image after applying this process over the whole voxels. The feature extraction process is shown in Figure 3.3.

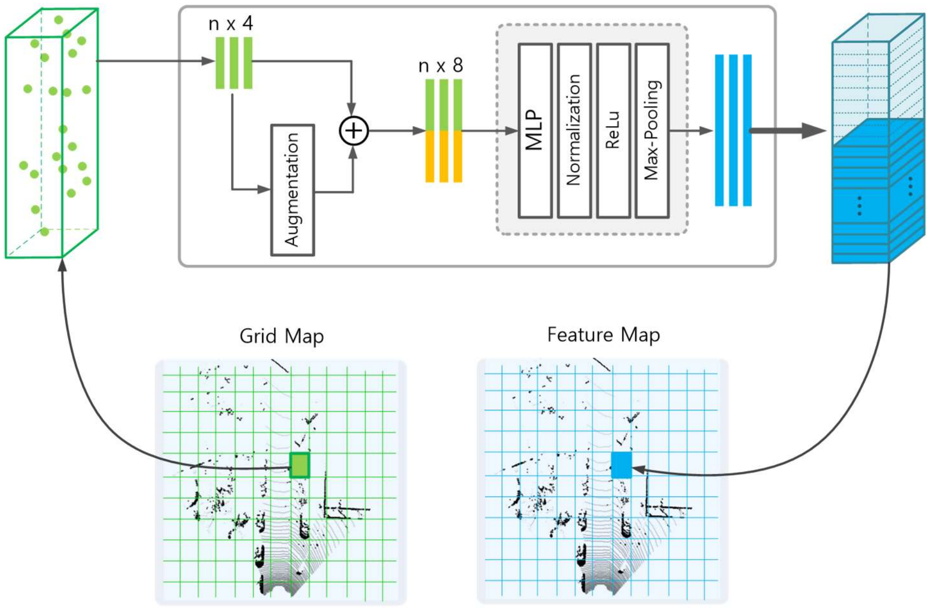


Figure 3.3. Feature Extraction Network

3.2. Backbone Network

As one of the basic components in object detectors, the region proposal network serves as an important module to decode the input feature maps and transform them into candidate boxes. The backbone structure is shown in Figure 3.4.

The backbone is consists of two sub-networks. The first sub-network has a form of top-down Feature Pyramid Network (FPN) design that produces features at increasingly small spatial resolution. The second sub-network takes features from the first network and performs upsampling. Upsampled features are concatenated at the end of the network.

The top-down network is a series of convolution layers followed by Batch Normalization and a ReLU. The feature passed a first top-down layer are combined with a non-ground mask created by the non-ground extraction process, then the combined feature passes a top-down network again. The final features from each block of first sub-network pass upsampling layer, then Batch Norm and a ReLU is applied to each upsampled feature. The final upsampled features are concatenated.

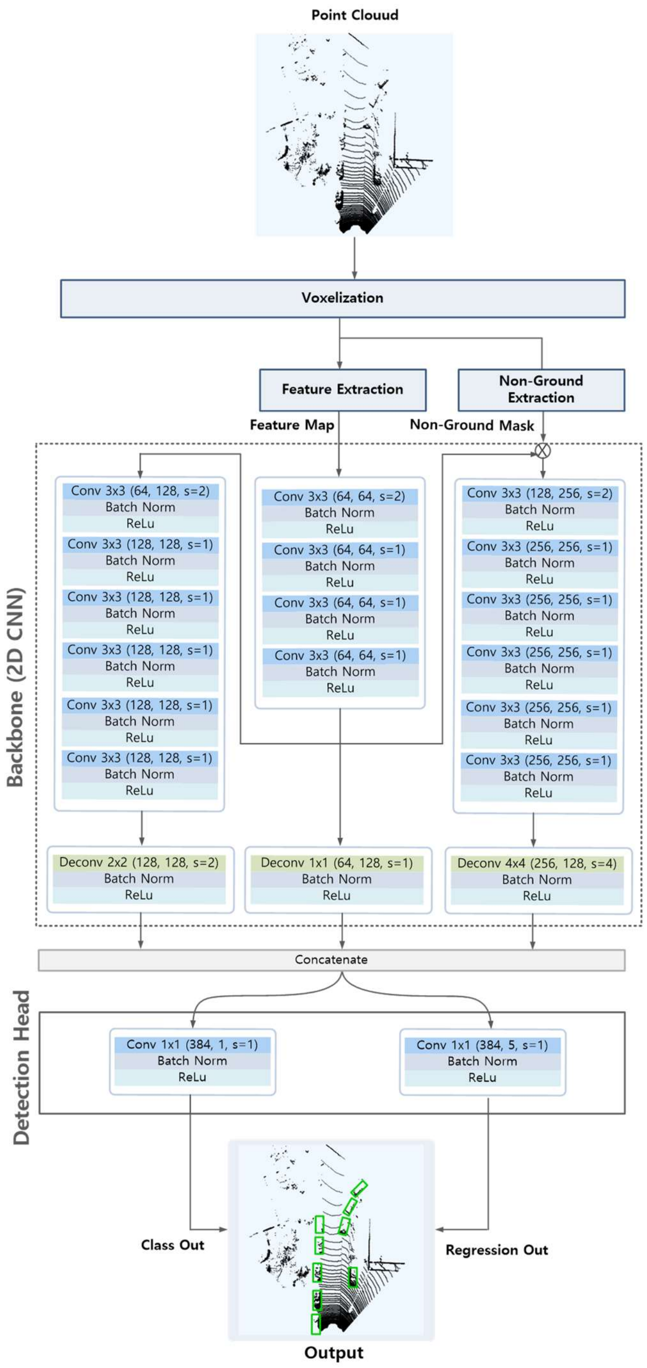


Figure 3.4. Backbone Network of object detector

3.3. Detection Head & Loss Function Design

Essentially, the trouble in object detection problem with LiDAR point cloud comes from the difference between image and point cloud's data form. RGB image data are 2D pixel-wise data and they are filled with meaningful data without emptiness. However, the point cloud is 3D physical data having distance and orientations. Even though we convert the point cloud point to voxel-wise data, there are only a few voxels filled with point cloud and the most voxels are empty and these voxels are out of our interests, and this imbalance between foreground voxels and background voxels results in degradation of detection performance while training.

LiDAR sparsity

On the other hand, most cells of voxelized will be empty due to sparsity of the point cloud, and the non-empty voxels will, in general, have few points in them. For example, despite the description in it's user manual that the LiDAR model used in KITTI, HDL-64E Velodyne LiDAR, generates 1,300,000 points per second in single return mode and 2,200,000 points per second in dual return mode, at $0.16^2\ m^2$ bins the point cloud from the LiDAR has 6k-9k non-empty voxels in the range typically used in KITTI for ~ 97% sparsity.

Class Imbalance Problem

Although the major driving force of progress in object detection has been the incorporation of deep neural networks, imbalance problems in object detection at several levels have also received significant attention because this imbalance can result in performance degradation.

The balanced distribution of the inputs is the key property affecting the performance. When the distribution imbalance is not addressed, an imbalance problem has adverse effects on the final detection performance. The most commonly known imbalance problem in object detection is the foreground-to-background imbalance which means the extreme inequality between the number of positive examples versus the number of negatives. In that imbalance case, if a given image, while there are typically a few positive examples, one can extract millions of negative examples. If not addressed, this imbalance greatly impairs detection accuracy.

Detection head

We found that this imbalance problem occurs in object detection using the LiDAR point cloud because of the aforementioned problems. To take a one-stage structure for efficient detection speed with mitigating the class imbalance problem, we adopt the focal loss as a loss function for class classification and the detection head of Single Shot Detector (SSD) because the structure's performance of SSD already verified in 2D object detection on the image. Similar to SSD, we set the anchor boxes and match the anchors to the ground truth using 2D Intersection over Union (IoU). The matching processed on the

Bird's Eye View (BEV), which means bounding box height and elevation were not used for matching. The height and elevation become additional regression targets.

3.4. Loss Function Design

Because we divide the range as equal-sized voxels, the objects to be detected are of an approximately fixed size. Thus, we use two fixed-size anchors determined based on the means of the size and center locations of all ground truths in the KITTI training set with a rotation of 0 and $\pi/2$. In the case of cars, we define an anchor with dimensions of (1.6 x 3.9 x 1.56m), centered at z = -1.0m. In a voxel, we can describe the relation between anchors and ground truth as shown in Figure 3.5. Similar to [Yan'18], we set the residuals between ground truth and anchors to be encoded with the following equations:

$$\begin{aligned}
&\Delta x = \frac{x^g - x^a}{d^a},\ \Delta y = \frac{y^g - y^a}{d^a} \\
&\Delta \omega = \log\frac{\omega^g}{\omega^a},\ \Delta l = \log\frac{l^g}{l^a} \\
&\Delta \theta = \sin\left(\theta^g - \theta^a\right)
\end{aligned} \tag{3.1}$$

Where x, y, z are the center coordinates and ω, l, h are the width, length, and height of the 3D bounding boxes respectively. θ is the heading angle, and

$d = \sqrt{l^2 + \omega^2}$ is the diagonal of the base of the anchor box. Subscripts ‘g’ and ‘a’ indicate the ground truth and anchor boxes respectively. The total localization loss is given by:

$$L_{loc} = \sum_{* \in (x,y,z,\omega,l,h,\theta)} SmoothL1(\Delta *) \tag{3.2}$$

According to SECOND [Yan’18], the angle loss form has two advantages. First, the adversarial problem occurs between orientations of 0 and π can be solved because 0 and π have the same orientation of the box, but they generate a large loss in the loss function. Secondly, it naturally models the IoU against the angle offset function. Because this angle function treats boxes with opposite directions as being the same, this approach can converge the angle loss regardless of direction problem.

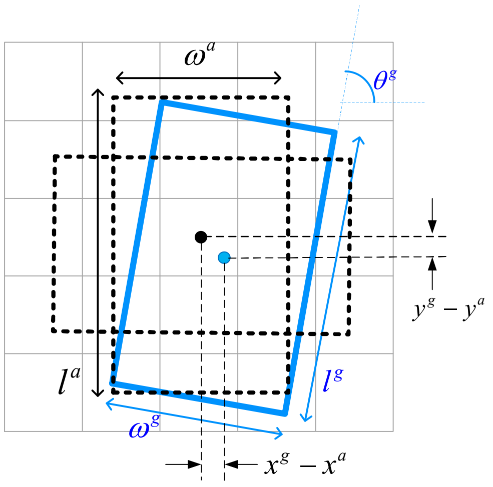


Figure 3.5. The relation between ground truth and anchors

As aforementioned, there is a class imbalance between non-object proposals and object proposals. For example, there are usually only a few voxels including points belong to ground truths even though our network usually generates about ~70k anchors within a KITTI dataset. So the most voxels in the input features are not related to the object. This lead to a considerable class imbalance between the foreground and background classes, which there are only a few foreground gird comparing to the background grid. To handle the imbalance problem, we use the focal loss for the object classification introduced by the authors of RetinaNet [Lin'17], thus the classification loss has the following form:

$$FL(p_t) = -\alpha\left(1 - p^a\right)^{\gamma}\log(p^a) \tag{3.3}$$

where p^a is the estimated probability of an anchor to be a positive sample and α and γ are the tuning parameters of the focal loss. We use the settings of α = 0.25 and γ= 2 according to original paper [Lin'17] in our training process. Therefore, by combining the losses discussed earlier, the total loss function is defined as follows:

$$L = \frac{1}{N_{pos}}\left(\beta_{loc} L_{loc} + \beta_{cls} L_{cls}\right) \tag{3.4}$$

Where is the classification loss and is the regression loss for location and object dimension. N_{pos} stands for a total number of positives, β_{loc} and β_{loc} are the constant coefficients of our loss function. To optimize the loss function we use Adam optimizer with an initial learning rate of 2×10^{-4} and decay the learning rate by a factor of 0.8 every 15 epochs, and we set total training epochs as 160.

3.5. Data Augmentation

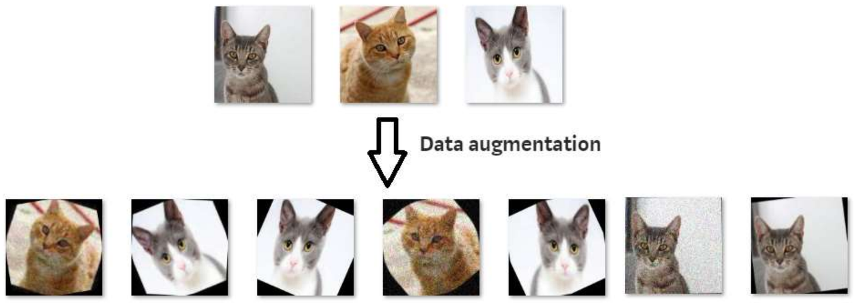


Figure 3.6. Data Augmentation Example

The neural networks based on supervised learning heavily depend on big data to avoid overfitting. Overfitting refers to the phenomenon when a network has high variance with respect to other datasets because it is intensively trained on specific data. Data Augmentation encompasses a set of techniques that apply a transformation on the size or quality of training datasets for better Deep Learning models to be built by using them. For this reason, data augmentation is a critical component of training deep neural network models. Data augmentation involves the process of creating new data by manipulating the original data by various methods as shown in Figure 3.6. The smaller the amount of data to train, the more effective this augmentation process can be for model training. In addition, overfitting also can be mitigated by data augmentation. In this dissertation, we apply basic augmentation such as global flip and rotation, translation on pseudo-image. Furthermore, we apply additional augmentation proposed by [Yan'18].

Global Translation and Global Rotation

We applied global translation and rotation to the whole point cloud and to all ground truth boxes. The translation is randomly applied from the uniform distribution [-30, 30] in x-direction and y-direction, and rotation augmentation is also randomly applied and the rotation angle range of $[-\pi/4, \pi/4]$ is used.

We tested our network via real vehicle test using 32 channel LiDAR, it will be introduced in later. The test LiDAR is mounted at the front of the test vehicle's bumper, and this position is different from the mount position of KITTI test vehicle. The difference in the sensor mounting position causes a change in the appearance of the scenes produced by the lasers emitted from the

LiDAR, and when the trained network is applied, the performance of the network is sensitive to the vertical position calibration.

To mitigate the sensitive performance change according to vertical calibration, vertical translation augmentation is applied to train the model as shown in Figure 3.7.

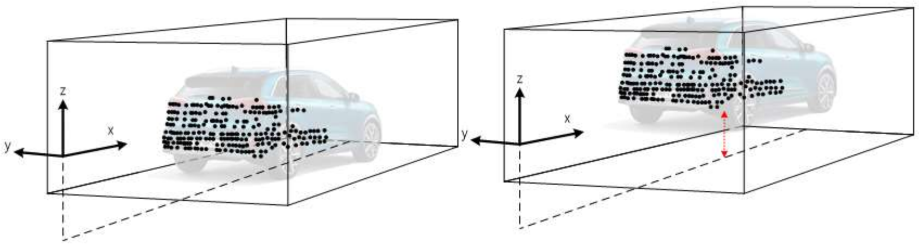


Figure 3.7. Vertical Translation augmentation

Global Flip & Scaling

The LiDAR product used in KITTI dataset is Velodyne's HDL-64E and this is mounted on the roof of the test vehicle to cover 360 degrees around the test vehicle. Even though all the point cloud around the test vehicle is served, the KITTI dataset offers ground truth's annotation only limited to objects within the front camera's field of view. In actual driving situations, however, the network may also need to detect vehicles in the rear. In order to train the model to cope with obstacles coming from the rear, the image created by LiDAR laser on the obstacle is needed to be similar to what of when the object is behind it. So we applied global vertical flip and horizontal flip to the whole point cloud and to all ground truth boxes.

Finally, we perform two kinds of global augmentations, which are scaling[Zhou'18, Yan'18] and noise addition that is jointly applied to the point cloud and ground truth information. When applying a noise augmentation to the point cloud, we randomly add or subtract the number of points from original point cloud data.

Ground Truths Scattering from the Database

As mentioned earlier, we encountered the imbalance problem during training. There are usually only a few ground truths even though our network usually generates about 70k anchors within a KITTI point cloud.

So the most voxels in the input features are not related to the object. This imbalance significantly limited the convergence speed and final performance of the network. As shown in Figure 3.8, the KITTI dataset used for training shows that there are less than 5 cars in most frames.

To solve this problem, we adopt a data augmentation approach similar to [Yan'18]. As shown in Figure 3.9, we construct a database containing the labels of all ground truths and their associated point cloud data from the training dataset. The associated point cloud data is filtered by extracting the points inside the 3D bounding boxes of the ground truths.

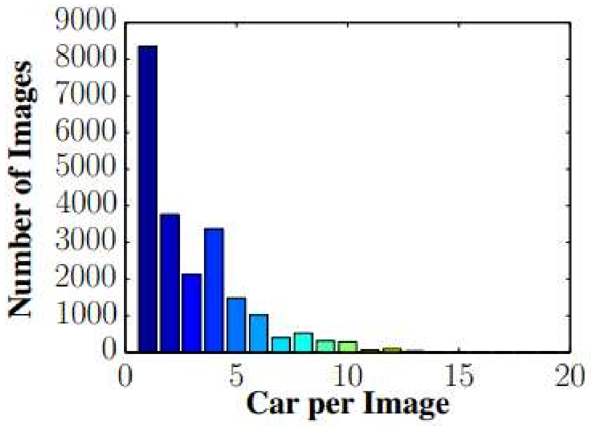


Figure 3.8. Number of car and image

Then, during training, as shown in Figure 3.9, we randomly selected several ground truths from this database and scattered them into the current training point cloud with translation and rotation augmentation. When the additional ground truth points from the database are scattered, we checked whether the points are overlapped with existing ground truth points to avoid an unnatural scene that the objects are overlapped. If there is overlapping, we re-scattered the points until the overlapping does not occur.

However, the method introduced in SECOND [Yan'18] did not care about the overlapping with the non-ground points, which are wall, flower bed, traffic sign, and tree, etc. In addition, when the additional points are scattered, they preserve their vertical positions. This results in an unnatural vertical position.

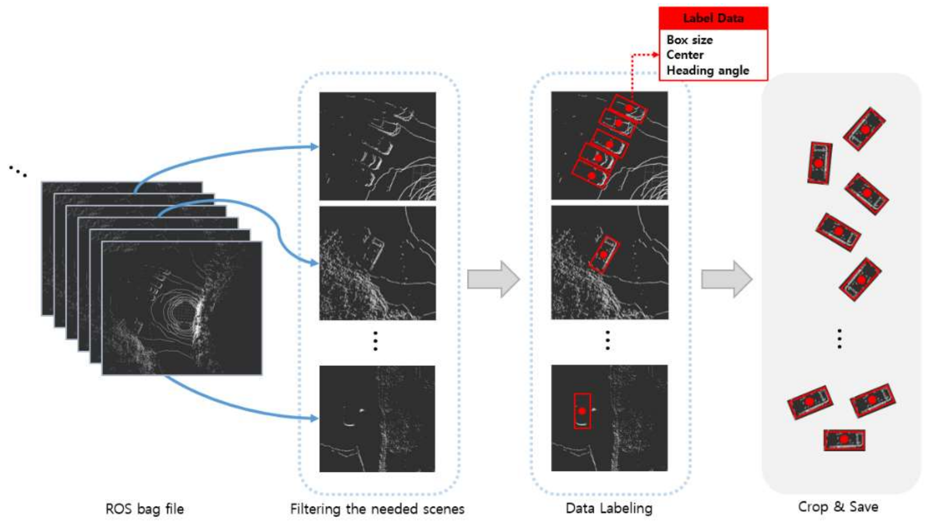


Figure 3.9. Construct Ground Truth Database

An unnatural scene means situations that cannot occur while real vehicle tests. For example, the scene that the objects are overlapped and the scene that cars are under the ground or placed off the ground. In addition, the situation that the closer object has sparse point cloud density than a distant object is can not occur physically. We wanted our model to learn more natural scenes because the unnatural scenes can degrade the final performance.

As shown in Figure 3.10, to avoid physically impossible outcomes, we take a step further from the proposed method in SECOND [Yan'18] by exploiting the aforementioned ground estimation. By estimating the ground points, we distinguish non-ground points from the point cloud. Thus, we additionally checked the scattering points overlapping with non-ground points, and move their vertical position to be placed on the ground points. Scattering positions are also constrained in a specific range for the relation between point number and its distance.

Using this augmentation approach, we could increase the number of ground truths per point cloud and simulate objects existing in different environments.

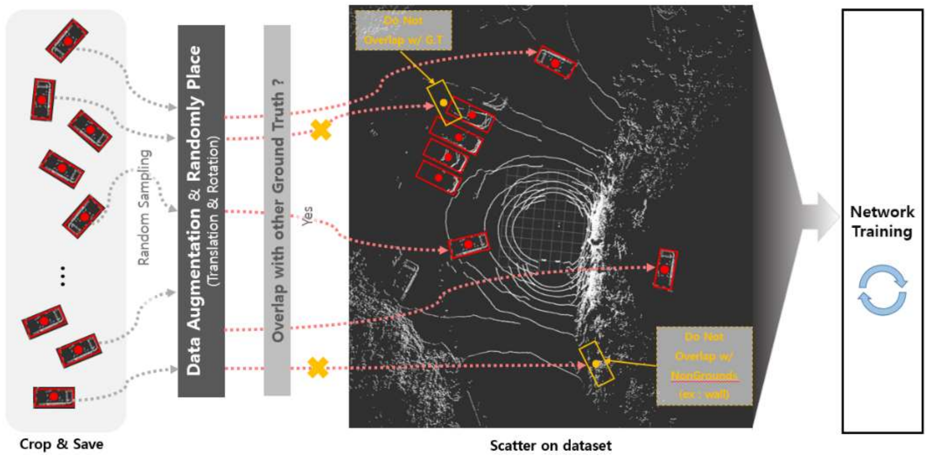


Figure 3.10. Random data scattering augmentation from database

3.6. Post Process

In detection, adjacent grids produce multiple prediction results for one object. In general, by applying non maximum suppression (NMS), prediction with the highest confidence is picked as a representative prediction, and the other predictions overlapping with the highest confident prediction are judged as predictions for the same object and they are erased. As an example, there are five predictions in Figure 3.11 and their confidence scores are shown in Figure 3.12 (a). From the definition of NMS, the number 3 prediction is selected as a result predictio. The confidence score is the probability that a prediction contains an object. However, the confidence score is just a value calculated by the object detector trained by dataset, not a strictly mathematically and physically derived value. Thus it cannot be convinced that the prediction with higher confidence leads to a more accurate results.

Therefore, there is a question that the existing NMS, which simply sorts predictions in order of confidence and determines the prediction with the highest confidence as the final prediction, is the most reasonable method. To compensate for these shortcomings of NMS, the polygon detection results covered in Chapter 4 are used.

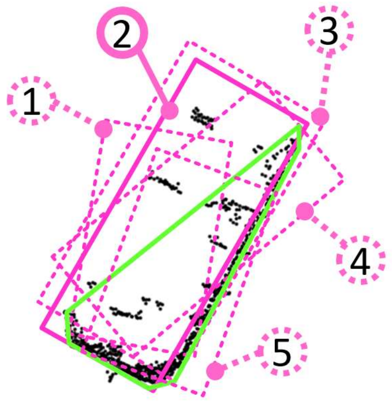
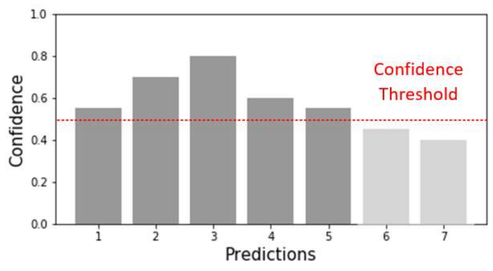


Figure 3.11. Polygon and predictions with confidence

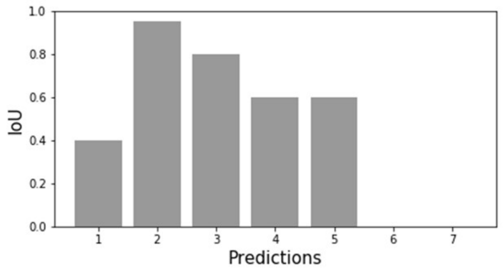
However, the number 2 prediction predicted fitted to the point cloud seems to be a more suitable prediction when viewed in intuitively. In this case, we use the polygon to predict the unknown object covered in Chapter 4 to induce the final prediction to be changed from 3 to 2. First, predictions are not sorted in the order of confidence score, but instead, they are sorted based on IoU with the polygon. At this time, the IoU with the polygon is defined as the ratio of the area of the polygon to the denominator and the overlapping area of the Polygon and the prediction to the denominator as defined in equation 3.5.

$$IoU = \frac{Area_{prediction} \bigcap Area_{polygon}}{Area_{polygon}} \leq 1 \tag{3.5}$$

The IoU results are shown in Figure 3.12 (b). In this case, the number 2 prediction has the highest IoU with the polygon. The IoU value with the polygon has a maximum of 1, and finally, predictions are sorted based on the multiplication of the confidence score and the IoU. The prediction with the highest value of the multiplication is picked as the result prediction. If the area of the polygon is not defined, the IoU is not considered and the final selection is made by sorting in confidence score order according to the existing NMS method.



(a) Predictions and their confidence scores



(b) Predictions and IoU with polygon

Figure 3.12. Confidence scores and IoU of predictions

Chapter 4 Non-Ground Point Clustering

In the classic rule-based method using the existing 2D LiDAR, erasing the ground was a basic task. This approach is based on the assumption that objects above a certain height above the ground are obstacles. This is a fairly appropriate assumption, and since it was a useful approach, ground removal was a very important process in performance. Following this manner, previous researches using 2D LiDAR have been used as a method of finding an object by fitting the features of an object that was supposed for the remaining points after deleting the points belonging to the ground. However, the neural network-based method is a method of learning the feature characteristics of an object from the position of point clouds and recognizing the object through their characteristics, so it is not necessary to remove a point belonging to the ground. This seems to be a very efficient and sophisticated task. Actually, object detection using the LiDAR point cloud has been shown relatively high performance and many researches have been conducted on various datasets.

Most of the detectors that show good performance in the benchmark of object detection are developed based on supervised learning. The detector based on supervised learning has a methodological limitation that it can only find the object learned in the learning process. Detectors that show good performance in most studies also show limited performance for specific objects. Since such performance considers only the detection results of every scene, detection continuity between scenes and scenes is not considered. Even though the detector has a very fast detection speed, it often fails to detect even a learned object due to the nature of the object detector.

In the actual driving process, objects that are not included in the dataset are often encountered. In this paper, we studied the method of detecting non-ground points and detecting them as unknown objects by clustering them in order to respond to these unlearned objects and objects that have been learned.

4.1. Previous Researches for Ground Removal

It is not an easy task to filter out points belonging to the ground in a 3D point cloud. There have been many attempts to solve this problem, and these can be broadly divided into three categories.

First, based on the specification of LiDAR and mount location, a method of finding based on the degree and height change of the locations where laser beams are formed. [Chu'17, Choi'13] However, this method is not robust to changes in the specifications and installation position of LiDAR, and is not efficient in terms of calculation since it extracts ground points for each beam individually. As mentioned earlier, this method is not appropriate because the real-time guarantee is very important in autonomous driving.

The second method is a ground plane estimation. [Asvadi'16, Zermas'17] In this method, planes representing ground are obtained through multiple iterations, and points coming within a certain distance are regarded as ground. However, in many cases, the road surface of the road we facing while driving is actually not neatly flat even though they look perfectly flat. The above method attempts to compensate for this by using multiple planes, but the number of planes must be fixed in advance, and multiple iterations are not efficient in terms of computation. Also, the method of filtering a point with a distance threshold from a plane does not completely perform ground removal.

Finally, ground point classification through segmentation using a neural network. [Velas'17, Dabbiru'20] This is the most attempted method in recent years by the universal application of neural networks.

Recently, even if not used in the ground removal method, this method is used in attempts to use confidence information about whether each point belongs to the ground to improve the detection performance of the network. [Shi'19, Yang'19] However, because the KITTI dataset used in this study does not provide annotations of ground points, the task of creating ground points annotations must be preceded in order to train such a network. This work is quite a time consuming and needs a hand-craft partially.

When a point belonging to the ground is mistaken for a non-ground point, it is recognized ground as an obstacle, so incorrect route planning and braking are executed, which can create incorrect behavior from the perspective of autonomous driving planning and control. For these reasons, we had to devise an effective rule-based method in this dissertation to secure both efficiency and performance.

4.2. Non-Ground Estimation using Voxelization

It is not easy to filter out the points belonging to the ground through the 3D point cloud. This requires a lot of complex computation, and these computations must work properly for various ground conditions. Estimating a specific ground plane or estimating whether it is a point belonging to the ground with height and angle differences between points does not produce an appropriate result for various situations.

In order to cope with these shortcomings and to produce appropriate results, we propose a simple method. In the object detector section, the point cloud is allocated in the 2D array through the voxelization divided into a certain size. Before the feature extraction, we calculate the height difference between the highest point as shown in Figure 4.1 and the lowest point for the point clouds in the voxel.

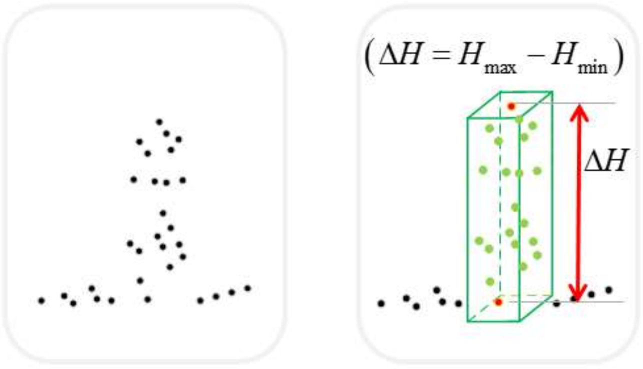


Figure 4.1. Vertical distance in a voxel for non-ground point extraction

If this height difference exceeds a certain value, it can be classified as an object. That is, the points belonging to this voxel are regarded as points belonging to a non-ground object or obstacle. Conversely, if the height difference is smaller than a certain value, the points belonging to this voxel are regarded as points belonging to the ground.

However, there is a case when the height difference is ambiguous, which is the case that the height is laid between two threshold values. [Wang'18] analyzed the variance of the point cloud of KITTI dataset, and found a tendency that the variance of the point cloud's reflectance on the ground is higher than that of other points. If the value of this variance is more than a certain point, the points in the voxel belong to the ground, and in the opposite case, it is considered to belong to the non-ground. By doing this, it is possible to distinguish non-ground points from ground points in a simple way, and it is computationally efficient because it uses voxelization used in the object detection process.

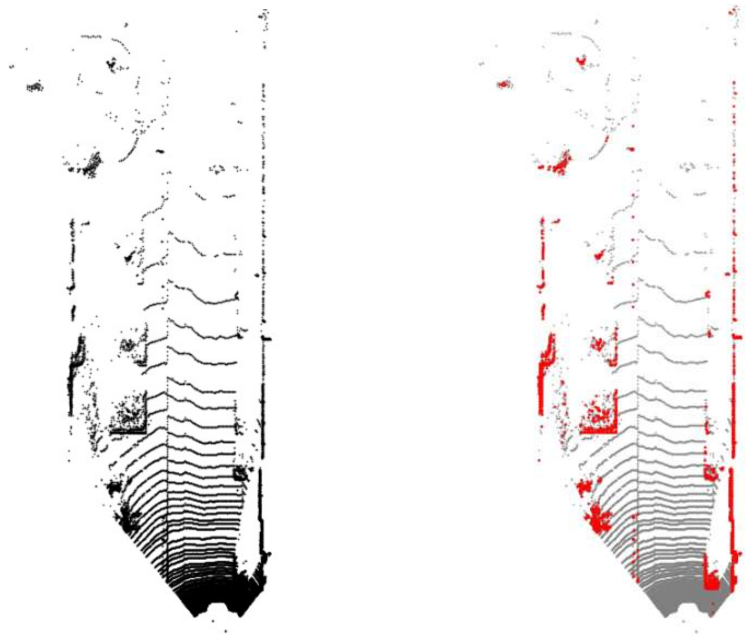


Figure 4.2. Non-ground point extraction result on KITTI dataset. Red points in the right figure are non-ground points, and grey points are grounds.

In addition, since this method produces results according to the relative distance of the point cloud along the z-axis, it is relatively robust for the specification and installation location of LiDAR. Figure 4.2 shows the result of non-ground point extraction is applied to KITTI dataset.

Table 1. Comparison of average processing time per frame for ground estimation

Method	Average processing time per frame (ms)
Fast ground segmentation [Chu,’17]	4.7
Sloped Terrain Segmentation [Cho,’14]	19.31
Loopy belief propagation based ground segmentation [Zhang,’15]	1000
Ground plane detection with RANSAC [Asvadi,’16]	15.4
Local convexity criterion [Moosmann,’09]	602
Proposed Method	8.3

Figure 4.3 and Figure 4.4 show the real vehicle test result. The LiDAR used in the vehicle test is 32 channels and installed in front of the vehicle. The driving course of the vehicle test is the road leading from Seoul National University Entrance Station to the main gate of Seoul National University. This road

consists of uphill and downhill roads. As a result of the qualitative evaluation, it was observed that the ground points and the non-ground points were properly classified even on the uphill and downhill roads. Table 1 shows the comparison of average processing time with other algorithms. As shown in the comparison result, Fast ground segmentation method is the fastest method followed by the proposed method. However, the proposed method is more efficient in this dissertation even though it has relatively low processing speed for the reason that the proposed network takes the voxelized feature as input. The main advantage of the proposed method is that it uses voxelization with high processing speed.

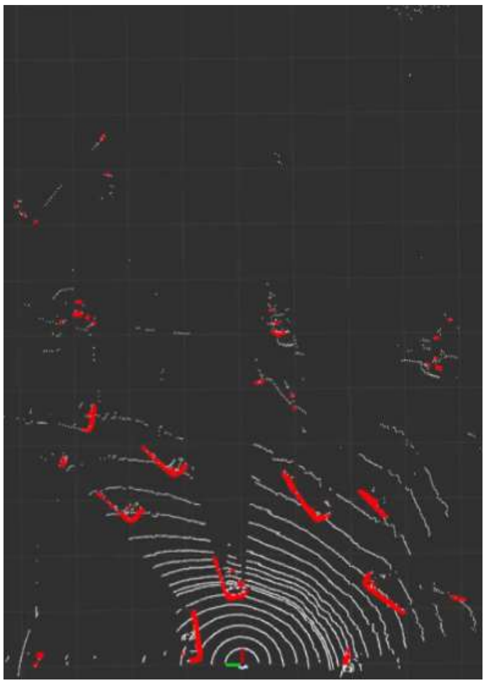


Figure 4.3. Bird's eye view of non-ground point extraction result on real vehicle test

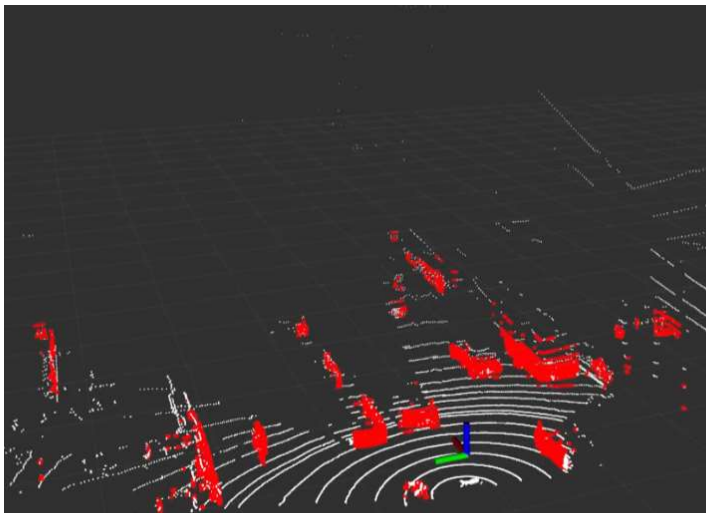


Figure 4.4. 3D view of non-ground point extraction result on real vehicle test

4.3. Non-ground Object Segmentation

In the methods of using the neural network, it is not necessary to distinguish which object each point belongs to, because they find objects directly through the point cloud raw data. Finding object detection without any pre-processing for point cloud input is the biggest advantage of neural networks. However, as mentioned above, such a supervised learning-based neural network has a disadvantage that it cannot detect objects except for the object the detector has learned.

Conversely, previous studies not using the neural network method used a model-free method, such as fitting objects for a certain size by segmenting the remaining points after first removing the ground, or matching cluster by considering dynamic information of the subject vehicle.

There are many clustering according to its manner, and hierarchical clustering, Centroid-based clustering, Distribution-based clustering, and Density-based clustering are the representative methods.

Hierarchical clustering is based on the idea that close points are more related than distant ones. It is divided into several types according to the way of calculating the distance, and in general, the complexity is $O(n^3)$ for agglomerative clustering and $O(2^{n-1})$ for divisive clustering, which makes them too slow for large datasets.

The most well-known algorithm of centroid-based clustering is k-means algorithms. It finds the k cluster centers and assigns the objects to the nearest cluster center, which the distances from the cluster are minimized. However the most -k-means algorithms require the number of clusters to be specified in advance, and this is considered as one of the biggest drawbacks because the cluster number can be varied in every time.

Density-based clustering separates high-density points into clusters, and objects in low-density areas are generally considered noise or boundaries. In density-based clustering, the most popular method is DBSCAN. (Density-based spatial clustering of applications with noise) It is based on connecting points within a certain distance range and it only includes points that satisfy a density criterion. This method has an advantage in that it is not necessary to determine the number of clusters in advance and can respond to various types of clusters. In this process, we cluster the non-ground points filtered in non-ground extraction process and create bounding polygon like bounding box detected by object detector. The total process of clustering and convex hull polygon with non-ground points are shown in Figure 4.5.

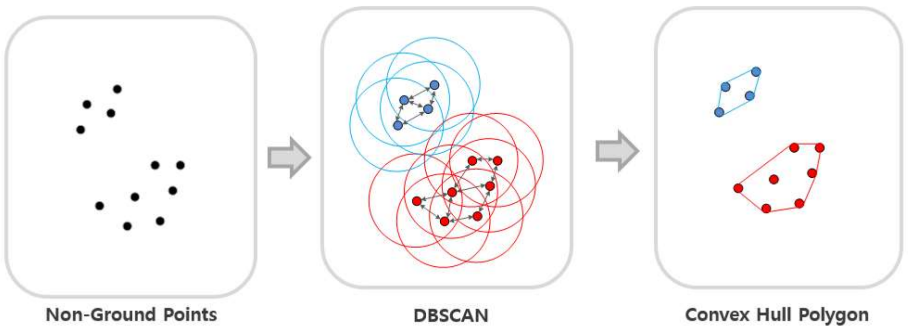


Figure 4.5. Point Clustering Sequence

4.3.1. Object Clustering

In this point clustering operation, specifying the number of clusters is not appropriate for situations in which various cluster numbers may occur while driving. In addition, in the case of the k-means algorithm, since an incorrect result is created for an arbitrary cluster type, a clustering technique showing an appropriate shape for an arbitrary cluster type is required. Also, an algorithm that can cluster points appropriately for any number of clusters is required.

DBSCAN (Density-based spatial clustering of applications with noise) can be the most appropriate algorithm for this task. DBSCAN does not require one to specify the number of clusters in the data a priori, as opposed to k-means clustering. In addition, it can find arbitrarily shaped clusters. It can even find a cluster completely surrounded by (but not connected to) a different cluster, and it is robust to outliers. This algorithm requires just two tuning parameters and is mostly insensitive to the ordering of the points in the database.

In general, it is ideal to apply DBSCAN to non-ground points, but when applying clustering to all non-ground points, too much of iteration is applied to points with only different z values for similar x and y positions. In Bird's eye view, these calculations along the z-axis have little significance for the clustering result. This calculation results in a computational load and affects the overall processing time.

When the total number of non-ground points is N and the number of voxels where non-ground points exist is M, generally N >> M is satisfied. Therefore, clustering based on the non-ground mask can reduce the calculation time without significant difference. This phenomenon is expected to reduce the total calculation time because the number of grids to be calculated decreases as the voxel size increases. About the scene shown in Figure 4.6, there are many differences in the number of points to be calculated between raw point cloud and voxelization. The total number of calculation leads to a large difference in calculation time.

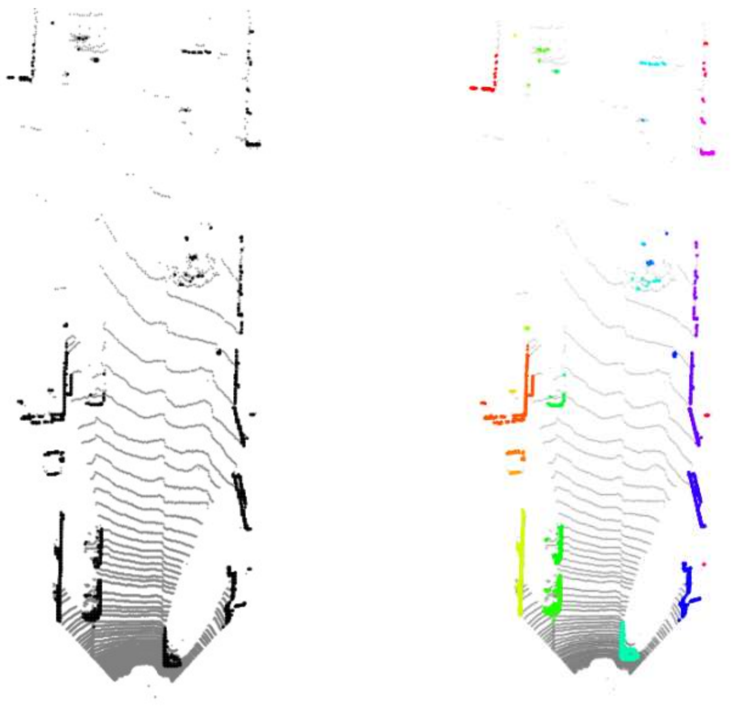
Table 2 shows the calculation time comparison according to the clustering method for the scene shown in Figure 4.6. When the voxelization is performed, the number of data to be calculated is reduced to more than one-tenth, and it is greatly reduced from the calculation time.

Therefore, as the voxel size increases, the efficiency of the calculation time increases. However, large voxel size may lead to inappropriate clustering results because the points belonging to different objects can be grouped together during clustering.

Therefore, it is important to cluster through the voxelization to maximize the speed without significant difference in clustering results, and to select voxel-size of the appropriate size empirically in the process

Table 2. Calculation time according to clustering methods and voxelsize

Method	No. of Data	Cal. Time [ms]	note
Raw Point Cloud Data	(11714, 4)	147.14	-
Voxleized Data	(1292, 2)	6.78	Voxelsize = 0.1
	(747, 2)	3.42	Voxelsize = 0.2
	(551, 2)	2.28	Voxelsize = 0.3
	(418, 2)	2.01	Voxelsize = 0.4



(a) Original point cloud data (b) After clustering

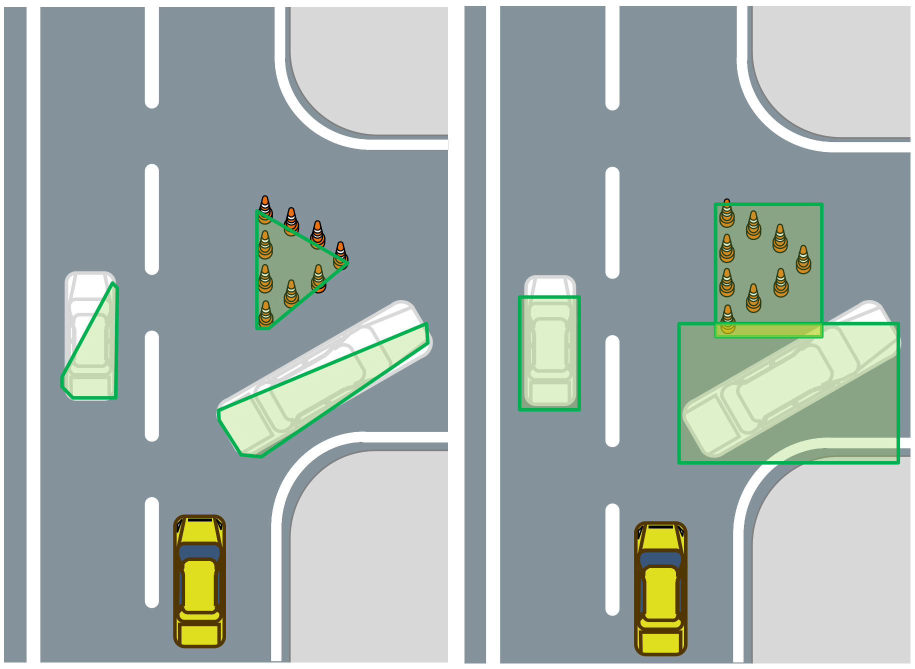
Figure 4.6. Clustering result on KITTI dataset

4.3.2. Bounding Polygon

In the case of object detection in this paper, detection is performed on a car, pedestrian, and cyclist. In the case of the object found by the object detector, the boundary of the object is defined through the bounding box, which includes the information such as object center position, width, length, the height of the box, and heading angle.

In the case of other obstacles, it is found as an unknown object through clustering. In this case, if a bounding box is defined with only the maximum and minimum x and y position of the cluster points, inefficient and unreasonable cases may occur as shown in Figure 4.7.

In order to solve this problem, this study defines the bounding polygon in the form of the polygon by finding the outer shell of an unknown object by using a convex-hull polygon.



(a) (b)

Figure 4.7. Two cases according to boundary of unknown object clustering

Chapter 5 . Object Tracking

Multi-object tracking (MOT) is an essential function for many applications not only autonomous driving. By applying the tracking function, the detection performance can be enhanced by maintaining detection continuously even if detection is missed. A lot of research on multi-object tracking has been conducted, and in the last 2 or 3 years, the car class on the KITTI MOT benchmark, the MOTA (multi-object tracking accuracy) has advanced from 57.03 [Yoon'18] to 84.24 [Sharma'18]. Despite the significant increase in MOTA score, the complexity and computational cost of the system also increased accordingly.

[Weng'19] proposed a simple but accurate 3D MOT system, which ensures a computational speed that can be utilized in real-time. We utilized the 3D MOT system architecture, and modify its inputs as object detection results and clustering for unknown objects, and its architecture is shown in Figure 5.1. The architecture of the tracking system is shown in Figure 5.1. Each tracking component has an associated Extended Kalman Filter that is used for prediction and estimation of the object state over time. Even if the objects go out of the detecting range and the detector lost the object, the tracking solution keeps track of detected objects until specified frames. Because the tracker predicts the movement of objects within the scene and to infer semantic information between frames, it can aid the object detection process.

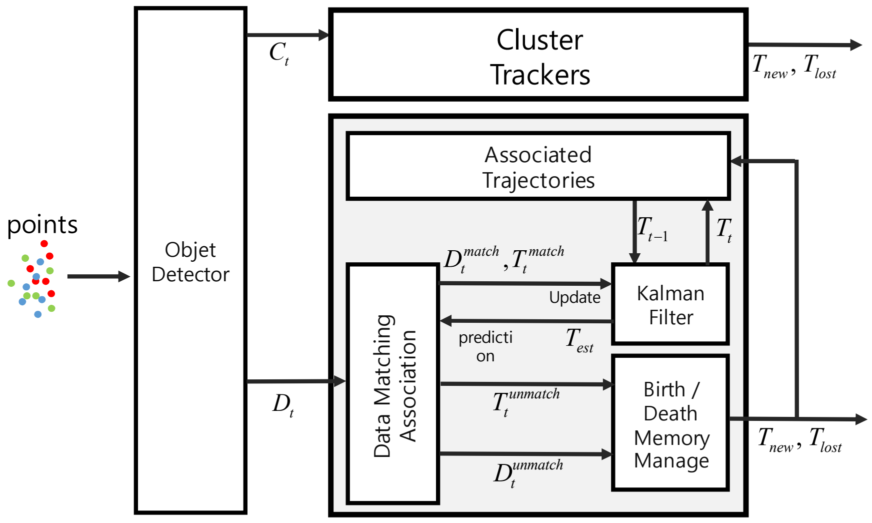


Figure 5.1. Multi Object Tracking Architecture

5.1. State Prediction and Update

In order to predict the state of the object in the next frame, a simple kinematic model using constant velocity is defined as shown in Figure 5.2.

As suggested in [Weng'19], even with a simple point model with a constant velocity assumption, there was no significant difference in model prediction results. But the model based on minimal kinematic is used for future works that to predict the possible dynamic behavior of objects using the ego vehicle's dynamic information. Because overlaps between object is physically impossible, we project the object behavior onto 2D image feature and define the state of the object as a 7-dimensional vector

$\mathbf{x}=[p_x, p_y, L, W, \psi, \dot{\psi}, V]^T \in \mathbb{R}^{(7\times1)}$, where p_x, p_y, L, W, ψ, $\dot{\psi}$, V means object x, y position, length and width of the bounding box, heading angle, and yaw rate, velocity respectively. The object's next position with respect to the current state can be defined in Equation (5.1) to (5.3).

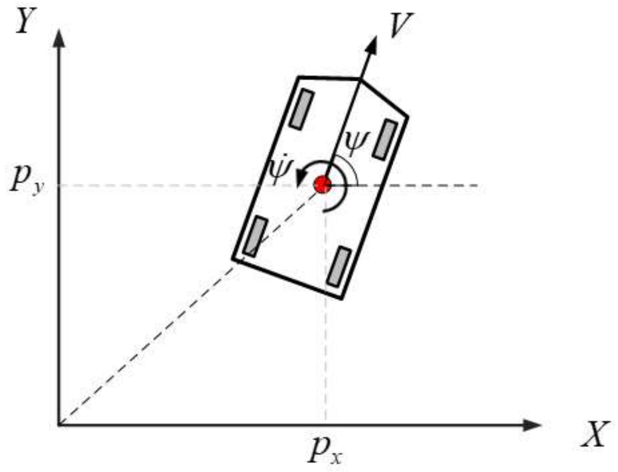


Figure 5.2. Vehicle model defined in object tracking

$$\mathbf{x}=[p_x, p_y, L, W, \psi, \dot{\psi}, V]^T \in \mathbb{R}^{(7\times1)} \tag{5.1}$$

$$\mathbf{z}=[p_x, p_y, L, W, \psi]^T \in \mathbb{R}^{(5\times1)} \tag{5.2}$$

$$\begin{aligned}\frac{d\mathbf{x}}{dt}&=\frac{d}{dt}[p_x, p_y, L, W, \psi, \dot{\psi}, V]^T \\ &=\begin{bmatrix} V\cos(\psi) & V\sin(\psi) & 0 & 0 & \dot{\psi} & 0\end{bmatrix}^T\end{aligned} \tag{5.3}$$

After state prediction, objects in the previous frame and the objects of the current state are compared and matched. This is described in detailed in the next section 5.2. At this stage, the objects are divided into matched and unmatched objects. Matched objects are considered to be the same object in the previous frame and the current frame, and unmatched objects are objects that were newly detected or existed in the previous frame but disappeared in the current frame. We update the state space to account for uncertainty for matched objects based on the measurements. Following the Bayes rule, the updated state is the weighted average between the prediction and the measurement. The weights are determined by the uncertainty of the prediction and measurement, and this is referred to Kalman filter [Kalman'1960] in detail.

5.2. Data Matching Association

The Multi-Object Tracking (MOT) solution creates and deletes object IDs constantly as objects enter and exit the scene. The memory director is responsible for the creation/deletion process based on the data association at the update step. At first, the detection results come from detection network and their corresponding Extended Kalman Filters are being calculated through their prediction step. Based on the predicted states of each EKF, an associated cost is being computed for each observation at the validation step a stored as a matrix. This matrix is based on the Intersection of Union (IoU) used to match the observations and the predictions. Since the detection speed is relatively very high than point cloud data update, the time gap between the two frames is very short. This means that the distance between data update is very short comparing to vehicle size. In this aspect, IoU is a suitable metric to use for the prediction-to-observation association.

To match the detections with prediction trajectories, we apply the Hungarian algorithm. The affinity matrix with a dimension of $n_t \times n_{t-1}$ is computed using the IoU between every pair of detection and prediction. Then the matching problems can be solved in polynomial time with the Hungarian algorithm. In addition, we reject the matching when the IoU is less than the threshold value. The outputs of the data association module are a set of detections matched with predictions along with the unmatched predictions and unmatched detections. As shown in Figure 5.3, there are sequential predictions along time T to time T+1. If IoU between the predictions of time T and T+1 is larger than threshold, we regard the predictions are for the same object.

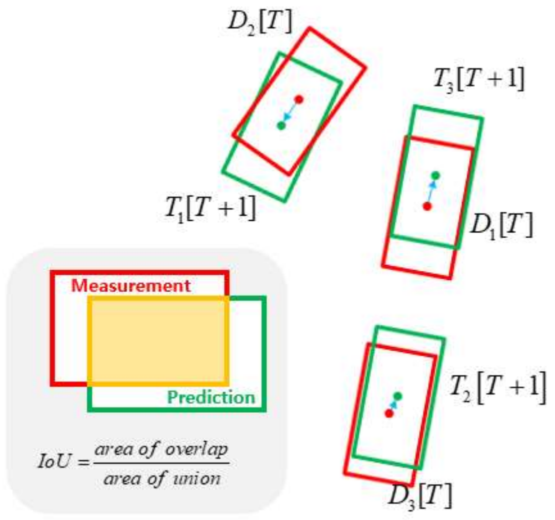


Figure 5.3. Prediction and detection matching with IoU

Chapter 6 Test result for KITTI dataset

We evaluate our proposed object detector by using the KITTI 3D/BEV object detection benchmark dataset, which contains 7,481 training samples and 7,518 testing samples. We divide the training samples into a training set with 3,712 samples and a validation set with 3,769 samples following the common manner. We trained the model with the divided training samples and compared it with other networks on validation samples. Since ground truth information provided by KITTI dataset is limited to those that exist inside the camera image view, it is necessary to evaluate only the objects that laid inside the camera image field of view. So we only used the LiDAR point clouds which can be projected onto the image for training. We conduct experiments on the car category, which is the most commonly used for network performance comparison. Average precision (AP) with an IoU threshold 0.7 was used as an evaluation metric to compare the results.

6.1. Quantitative Analysis

Network performance comparison proceeded for both Birds Eye View and 3D. In the case of birds-eye-view, the detection result is projected on the x-y plane by removing the z-axis information of the detection result. In the case of 3D detection, even if the x, y, and heading of detection are matched with ground truth, IoU between detection and ground truth can be lower than 0.7 because of the height difference. The 3D object detection performance is generally lower than birds-eye-view because IoU threshold 0.7 is more difficult in 3D detection considering height information additionally.

We compare the model to other approaches by using validation samples. The KITTI dataset is stratified into three difficulty levels: easy, moderate, and hard. These difficulties are divided based on the point cloud number on object and obscurity, object distance, and so on.

Table 3 shows a comparison result of our model and the different methods for BEV. As shown here, our detector achieved the highest AP score for hard. Although our method did not get the highest score for other difficulties, it got a high AP score for the remaining difficulties, which are easy, moderate. In addition, despite using only the Lidar point cloud, it scored a higher AP score than other networks such as the ones using fusion except for Frustum ConvNet and AVOD in Easy difficulty.

Table 4 shows a comparison result of our model and the different methods for 3D detection. Unlike BEV, our network did not achieve the best AP score in 3D detection. From this result, we can deduce that our network gives good detection for the object's center position, heading angle and length, and width of bounding box except for height.

As mentioned earlier, recent researches in the field of object detection for autonomous driving focus more on improving AP performance. However, the perception module works with other systems such as planning and control modules in actual autonomous driving. This results in a much larger computational load than when only the perception module is operated.

Even if a perception module has very good performance, it cannot be used in actual driving situations if it cannot guarantee real-time performance. For most LiDARs, the data update period can be set to 0.1 sec as default and can be adjusted more quickly to 0.05 sec, 0.025 sec. Depending on the setting. Therefore, it can be seen that the minimum real-time performance can be guaranteed only when the processing speed of the network is 10 Hz or higher.

In this aspect, the network proposed in this dissertation guarantees real-time better than other methods. We run our model in a desktop equipped with an Intel i7 CPU and a NVIDIA RTX2070 GPU. The overall pipeline consists of following steps: 1) read the LiDAR point cloud data file and extract the points inside the range of interest, 2) encode the point clouds into a voxelized map, 3) extract the non-ground voxels, 4) extract the feature characteristics and generate a feature map, 5) processing by the backbone and detection heads, 6) application of NMS on the CPU. As shown in the comparison results, our network shows good processing speed, although the performance of the network does not superior to other networks with high differences.

The comparison results indicate that some networks may perform slightly better than our network, but our approach shows better detection speed than other approaches. Complex YOLO has a processing speed of 50 frames per second, but in terms of AP score, it is much less than other networks. It can be seen that the network proposed in this study guarantees good detection speed with relatively high detection performance.

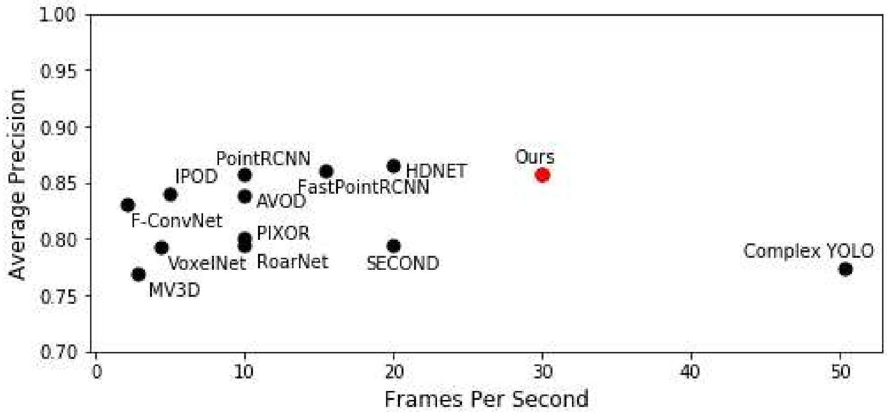


Figure 6.1. Performance comparison about car on BEV

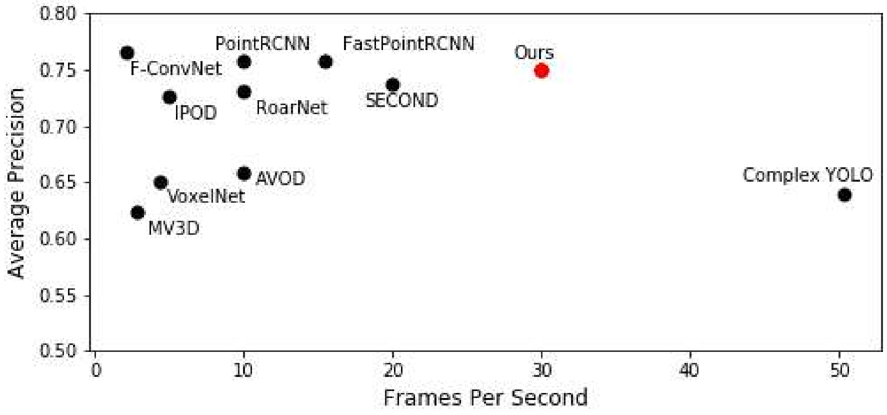


Figure 6.2. Performance comparison about car on 3D detection

Table 3. Results comparison on the KITTI test BEV detection benchmark

Method	Modality	Car (BEV)			Time
		Easy	Mod.	Hard	
MV3D [Chen,'17]	LiDAR + RGB	86.02	76.90	68.49	0.36
AVOD [Ku,'18]	LiDAR + RGB	88.53	83.79	77.90	0.1
Fast PointRCNN [Chen'19]	LiDAR	88.03	86.10	78.17	0.065
PIXOR [Yang,'18]	LiDAR	84.44	80.04	74.31	0.1
HDNET [Yang'18]	LiDAR + Map	89.14	86.57	78.32	0.05
RoarNet [Shin'18]	LiDAR + RGB	88.20	79.41	70.02	0.1
IPOD [Yang'18]	LiDAR + RGB	86.93	83.98	77.85	0.2
F-ConvNet [Wang'19]	LiDAR + RGB	89.69	83.08	74.56	0.47
VoxelNet [Zhou,'18]	LiDAR	89.35	79.26	77.39	0.5
PointRCNN [Shi,'19]	LiDAR	89.47	85.68	79.10	0.1
SECOND [Yan,'18]	LiDAR	88.07	79.37	77.95	0.05
Complex YOLO [Simon,'18]	LiDAR	85.89	77.40	77.33	0.02
MMF [Liang,'19]	LiDAR + RGB	89.49	87.47	79.10	0.08
Proposed	LiDAR	88.25	85.80	79.66	0.028

Table 4. Results comparison on the KITTI test 3D detection benchmark

Method	Modality	Car (3D)			Time
		Easy	Mod.	Hard	
MV3D [Chen,'17]	LiDAR + RGB	71.09	62.35	55.12	0.36
AVOD [Ku,'18]	LiDAR + RGB	73.59	65.78	58.38	0.1
Fast PointRCNN [Chen'19]	LiDAR	84.28	75.73	67.39	0.065
PIXOR [Yang'18]	LiDAR	-	-	-	0.1
HDNET [Yang'18]	LiDAR + Map	-	-	-	0.05
RoarNet [Shin'18]	LiDAR + RGB	83.71	73.04	59.16	0.1
IPOD [Yang'18]	LiDAR + RGB	79.75	72.57	66.33	0.2
F-ConvNet [Wang'19]	LiDAR + RGB	85.88	76.51	68.08	0.47
VoxelNet [Zhou,'18]	LiDAR	77.47	65.11	57.73	0.5
PointRCNN [Shi,'19]	LiDAR	85.94	75.76	68.32	0.1
SECOND [Yan,'18]	LiDAR	83.13	73.66	66.20	0.05
Complex YOLO [Simon,'18]	LiDAR	67.72	64.00	63.01	0.02
MMF [Liang,'19]	LiDAR + RGB	86.81	76.75	68.41	0.08
Proposed	LiDAR	78.85	74.92	68.10	0.028

Table 5. Results comparison on the KITTI test BEV detection benchmark

Method	Modality	Pedestrian (BEV)			FPS
		Easy	Mod.	Hard	
MV3D [Chen,'17]	LiDAR + RGB	-	-	-	0.36
AVOD [Ku,'18]	LiDAR + RGB	58.75	51.05	47.54	0.1
Fast PointRCNN [Chen'19]	LiDAR	-	-	-	0.065
PIXOR [Yang'18]	LiDAR	-	-	-	0.1
HDNET [Yang'18]	LiDAR + Map	-	-	-	0.05
RoarNet [Shin'18]	LiDAR + RGB	-	-	-	0.1
IPOD [Yang'18]	LiDAR + RGB	60.83	51.24	45.40	0.2
F-ConvNet [Wang'19]	LiDAR + RGB	58.90	50.48	46.72	0.47
VoxelNet [Zhou,'18]	LiDAR	46.13	40.74	38.11	0.5
PointRCNN [Shi,'19]	LiDAR	55.92	47.53	44.67	0.1
SECOND [Yan,'18]	LiDAR	55.10	46.27	44.76	0.05
Complex YOLO [Simon,'18]	LiDAR	46.08	45.90	44.20	0.02
MMF [Liang,'19]	LiDAR + RGB	-	-	-	0.08
Proposed	LiDAR	56.84	50.20	46.90	0.028

Table 6. Results comparison on the KITTI test 3D detection benchmark

Method	Modality	Pedestrian (3D)			FPS
		Easy	Mod.	Hard	
MV3D [Chen,'17]	LiDAR + RGB	-	-	-	0.36
AVOD [Ku,'18]	LiDAR + RGB	50.80	42.81	40.88	0.1
Fast PointRCNN [Chen'19]	LiDAR	-	-	-	0.065
PIXOR [Yang'18]	LiDAR	-	-	-	0.1
HDNET [Yang'18]	LiDAR + Map	-	-	-	0.05
RoarNet [Shin'18]	LiDAR + RGB	-	-	-	0.1
IPOD [Yang'18]	LiDAR + RGB	56.92	44.68	42.39	0.2
F-ConvNet [Wang'19]	LiDAR + RGB	52.37	45.61	41.49	0.47
VoxelNet [Zhou,'18]	LiDAR	39.48	33.69	31.5	0.5
PointRCNN [Shi,'19]	LiDAR	49.43	41.78	38.63	0.1
SECOND [Yan,'18]	LiDAR	51.07	42.56	37.29	0.05
Complex YOLO [Simon,'18]	LiDAR	56.66	49.01	45.66	0.02
MMF [Liang,'19]	LiDAR + RGB	-	-	-	0.08
Proposed	LiDAR	50.92	43.48	41.45	0.028

Table 7. Results comparison on the KITTI test BEV detection benchmark

Method	Modality	Cyclist (BEV)			FPS
		Easy	Mod.	Hard	
MV3D [Chen,'17]	LiDAR + RGB	-	-	-	0.36
AVOD [Ku,'18]	LiDAR + RGB	68.06	57.48	50.77	0.1
Fast PointRCNN [Chen'19]	LiDAR	-	-	-	0.065
PIXOR [Yang'18]	LiDAR	-	-	-	0.1
HDNET [Yang'18]	LiDAR + Map	-	-	-	0.05
RoarNet [Shin'18]	LiDAR + RGB	-	-	-	0.1
IPOD [Yang'18]	LiDAR + RGB	56.92	44.68	42.39	0.2
F-ConvNet [Wang'19]	LiDAR + RGB	82.59	68.62	60.62	0.47
VoxelNet [Zhou,'18]	LiDAR	66.70	54.76	50.55	0.5
PointRCNN [Shi,'19]	LiDAR	81.52	66.77	60.78	0.1
SECOND [Yan,'18]	LiDAR	73.67	56.04	48.78	0.05
Complex YOLO [Simon,'18]	LiDAR	66.70	54.76	50.55	0.02
MMF [Liang,'19]	LiDAR + RGB	-	-	-	0.08
Proposed	LiDAR	78.51	62.22	55.81	0.028

Table 8. Results comparison on the KITTI test 3D detection benchmark

Method	Modality	Cyclist (3D)			FPS
		Easy	Mod.	Hard	
MV3D [Chen,'17]	LiDAR + RGB	-	-	-	0.36
AVOD [Ku,'18]	LiDAR + RGB	64.00	52.18	46.61	0.1
Fast PointRCNN [Chen'19]	LiDAR	-	-	-	0.065
PIXOR [Yang'18]	LiDAR	-	-	-	0.1
HDNET [Yang'18]	LiDAR + Map	-	-	-	0.05
RoarNet [Shin'18]	LiDAR + RGB	-	-	-	0.1
IPOD [Yang'18]	LiDAR + RGB	71.40	53.46	48.34	0.2
F-ConvNet [Wang'19]	LiDAR + RGB	79.58	64.68	57.03	0.47
VoxelNet [Zhou,'18]	LiDAR	61.22	48.36	44.37	0.5
PointRCNN [Shi,'19]	LiDAR	73.93	59.60	53.59	0.1
SECOND [Yan,'18]	LiDAR	70.51	53.85	46.90	0.05
Complex YOLO [Simon,'18]	LiDAR	68.17	58.32	54.30	0.02
MMF [Liang,'19]	LiDAR + RGB	-	-	-	0.08
Proposed	LiDAR	74.95	59.01	52.85	0.028

6.2. Qualitative Analysis

The qualitative results are shown in Figure 6.3 and 6.4. Despite we trained our model only on LiDAR point clouds, we visualize the 3D predictions on BEV and image perspective. Figure 6.3 shows the vehicle prediction results. As can be seen in the figure, the prediction of the vehicle is particularly accurate compared to pedestrian and cyclist. In most cases, prediction about the object is accurate about the size and orientation.

In particular, the prediction of the vehicle is very accurate. Even when several vehicles are parked and several of them are occluded, it predicts well with looking at parts of their shapes. In addition, it finds the vehicle well even when it is relatively far away. This is because, unlike pedestrians and cyclists, the size of the car is larger, so the number of point clouds is higher than the pedestrian and cyclist even when it is far away. The detector also finds pedestrians and cyclists well. Especially when two pedestrians are close together, the detector predicts they as two pedestrian not one person or cyclist. However, if two close pedestrians are farther away, it can be predicted incorrectly, as shown in Figure 6.4.

As shown in Figure 6.4, however, there are some failures in some cases, which includes false negative and false positive. For false negative, the prediction may be missed if there are few points on the object, which is the case that the object is partially occluded or far away. Especially, this phenomenon is easy to occur about pedestrian and cyclist.

In Figure 6.4 (a), the detector predicts two pedestrian as a cyclist. Since the two pedestrians are close to each other and far away from the ego-vehicle, it can be seen of as the size of a cyclist from the viewpoint of the detector. As shown in (b) of Figure 6.4, the detector does not find a pedestrian that is far away, and it mis-predicts a pedestrian as cyclist. When the cyclist is far away, there is not enough point cloud, so it is hard to find the object or predict object's label correctly. As shown in (c) of Figure 6.4, if the pedestrian is close to the vehicle, the number of points belong to pedestrian are reduced because it is occluded. This makes detector to miss the pedestrian.

The object detector based on the neural network recognizes and judges objects through the distribution of point clouds. Because pedestrians or bicyclists have the number of points significantly less than that of the vehicle object, there is a high possibility that recognition may fail or be confused with other similar objects.

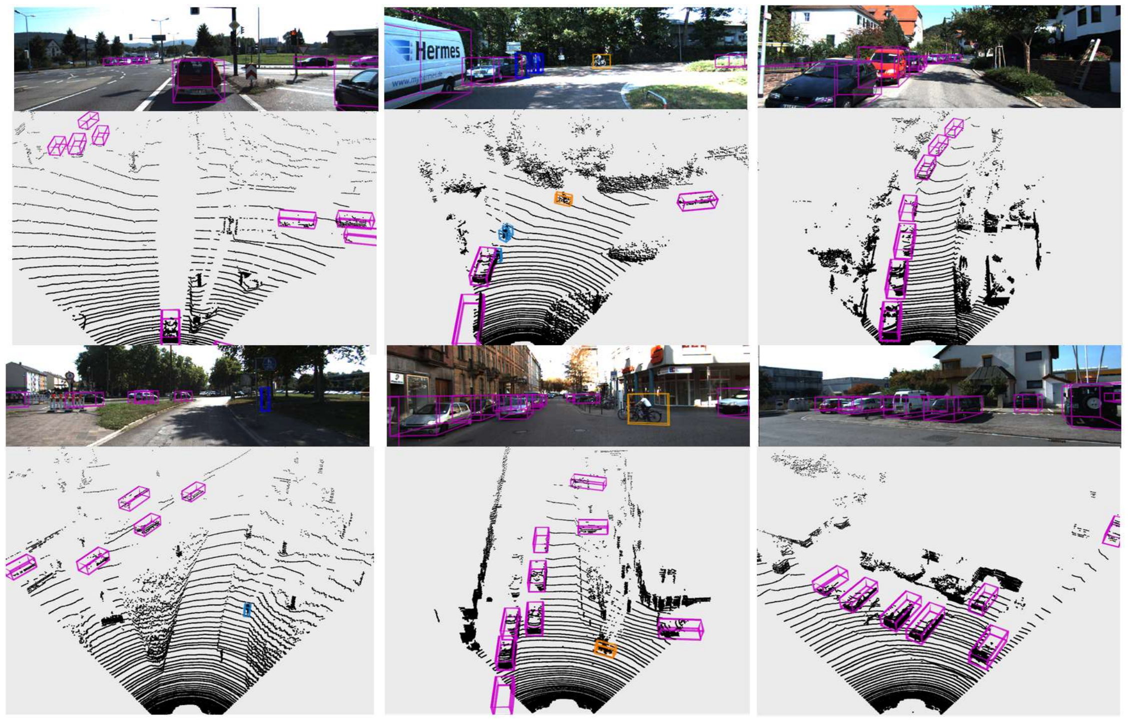


Figure 6.3. Qualitative analysis of KITTI results. We show the 3D bounding boxes projected onto the image (top), as well as 3D point cloud view (bottom). The predicted boxes for car are ‘purple’, for pedestrian are ‘blue’, for cyclist are ‘orange’.

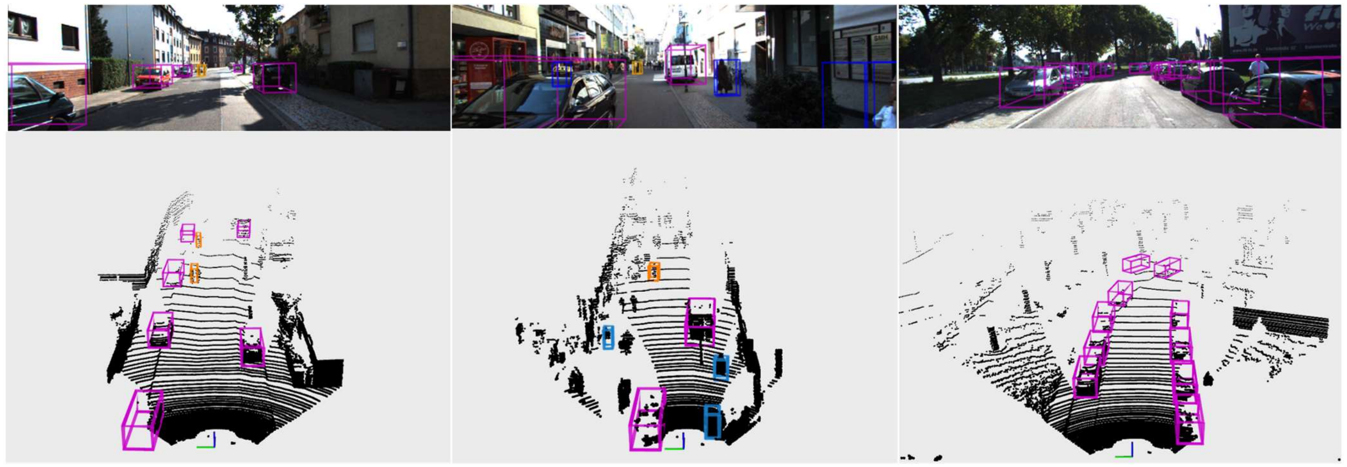


Figure 6.4. Failure cases on KITTI. The predicted bounding boxes are shown in 3D point cloud (top) and they are projected onto RGB image (bottom).

6.3. Additional Training

The KITTI dataset was obtained in the metropolitan area of Karlsruhe, Germany and the driving course is shown in Figure 6.5 [Geiger'13]. A driving environment consists of city, residential, campus, road, and others. KITTI dataset provides several labels, which are *car*, *van*, *truck*, *pedestrian*, *person* (sitting), *cyclist,* and *tram*. *car* and *pedestrian* occupy most of the dataset. The KITTI dataset has been used in many studies in the past few years and is now famous in the field of autonomous driving object detection. Many studies have validated the performance of their approaches by using this dataset. However, the KITTI dataset has limitations that it cannot cover all objects can show in other countries or region.

For example, there are some differences in the road conditions between Karlsruhe in Germany and Seoul in Korea. First of all, in the case of cyclists appearing in the KITTI dataset, they often appear on campus or with people, and there are no complex situations where the cyclist is moving between vehicles.

However, in Korea, where delivery and public transport environments are well established, it is very common to encounter buses or motorcyclists on the road. In the case of the bus, it is impossible to train *bus* because KITTI dataset does not offer data about *bus*. A method that trains *bus* by scaling the car was considered, but this method has limitations because the appearance of a general car and a bus is very different.

With the same reason, it is impossible to train motorcyclists. Since supervised learning relies heavily on the dataset, to train objects that do not exist in the dataset is a methodological problem. Although the KITTI dataset has been used in broad studies, there are some shortages to apply to the Korean road environment.

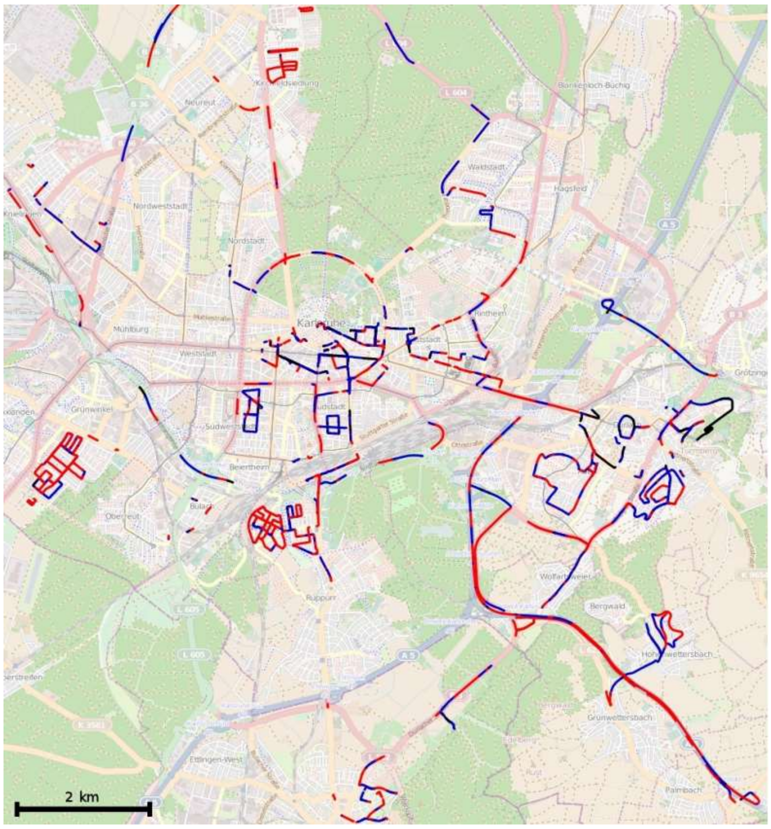


Figure 6.5. Recording zone of KITTI dataset [Geiger'13].

6.3.1. Additional data acquisition

The motorcyclist has a shape similar to that of cyclist. As shown in Figure 6.8, the network trained about the cyclist also detects the motorcyclist. However, it fails to detect the bus. Usually, Bus is larger and it has a shape closer to the cuboid box than a passenger car and van, so it can be recognized as a shape near the edge of the wall.

To solve the problem of bus detection, we need to train the bus. In order to acquire bus data, the KITTI dataset does not offer, driving data was collected through our test vehicle and additional data was obtained from it. Note, as shown in Figure 6.6, our data-collecting vehicle is equipped with a 32 channel LiDAR in front of the vehicle, which is different from the KITTI data acquisition vehicle that used 64 channel LiDAR mounted on the roof. While driving, we encountered many cars and buses, their point cloud data were stored in the form of a bag file of the robot operating system (ROS) in real-time.

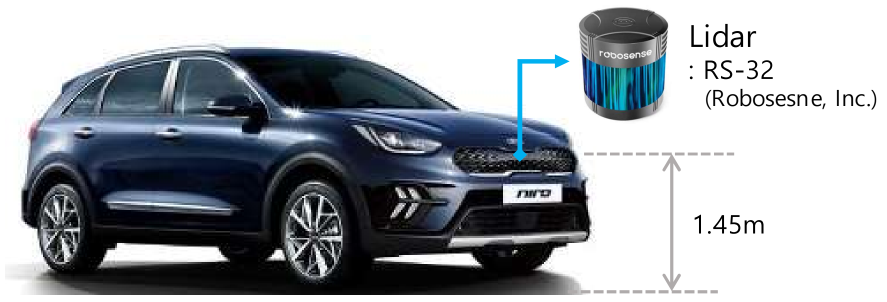


Figure 6.6. Test vehicle and sensor configuration

To make an annotation on objects that appear in every frame is handcraft jobs. Two approaches could be considered at the making annotation stage: 1) To annotate all existing objects including bus (car, pedestrian, cyclist), or 2) to annotate the only bus. The second method was chosen because labeling ground truth information for all objects in the frame, including the bus, takes a lot of time and labor. The ground truth extraction and build database augmentation mentioned earlier was used in this process.

We collect as many cases as possible such as partially occluded, observed in various angles, and create an annotation for the center, bounding box, and heading angle information on the bus. After annotation, we crop the point cloud related to the bus and move their center position to the origin and the heading angle to zero through translational and rotational transformation. Cropped and transformed point cloud is saved in the bus database, thus the database consists of bus data with a center of origin and zero heading angle. Figure 6.7 shows examples of saved *bus* data. At the training process, some buses randomly selected from the database and then scattered onto the scene with randomly applying translational and rotational transformation. In this way, the *bus* is added to the random scattering augmentation mentioned above and learned.

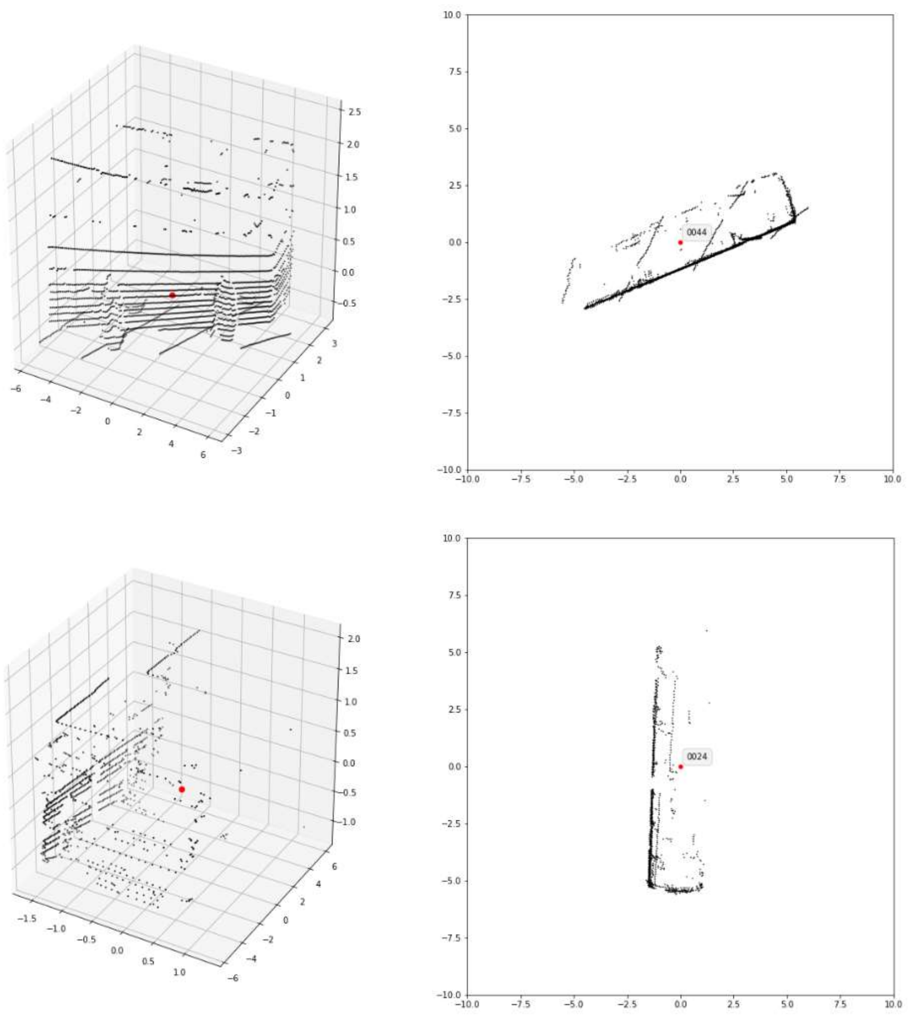
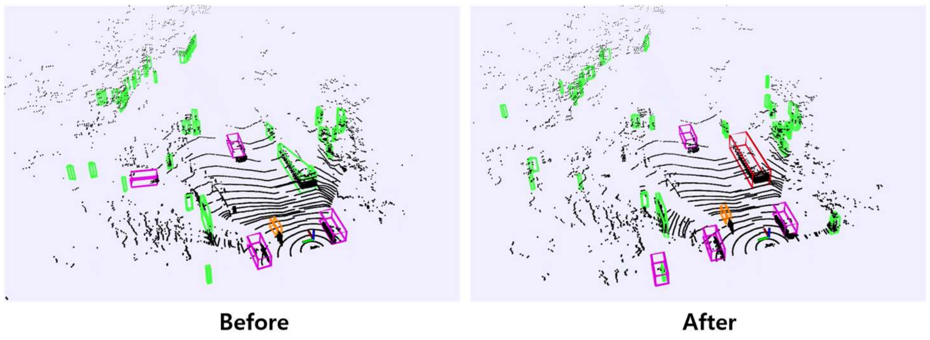


Figure 6.7. Bus data extraction example

6.3.2. Qualitative Analysis

Qualitative results are described in this chapter. Because the KITTI dataset does not serve bus data, we extracted the point cloud data about the bus from our driving data. We train the network by adding random scattering augmentation about the bus on training. As shown in Figure 6.8, two networks (a model trained including bus and a model only trained about the car) are compared on the same driving data. After the network is trained about the bus, it is able to find the bus it could not find before training the bus. The detection results on the various situations after training about the bus are shown in Figure 6.9. As shown in the figure, after learning about the bus, the network finds a parked bus or an approaching bus in the opposite lane.

However, the fluctuate prediction about the bus's size occurs sometimes. As shown in Figure 6.10 (a), the prediction of the size of the bus changed in the case that the bus is occluded at first and gradually appeared by approaching closer. At first, the size of the bus is predicted to be small when the bus is partially occluded, and if the bus is gradually revealed, the size prediction of the bus gradually changes to be large. By training the bus, it seems that the accuracy of the size prediction about cars decreased. This means that the variance of the prediction for the size of the passenger car was increased compared to the case where only the passenger car was trained. In addition, as shown in Figure 6.10 (b), there are some false positive cases where the wall or the object having a similar shape is mispredicted as a bus. Because, unlike a car or a van, the bus's height is very high, and it has a shape closer to the cuboid than a car when it viewed in the rear and side view.


(a) After training about the bus, the network detect bus on the right side.

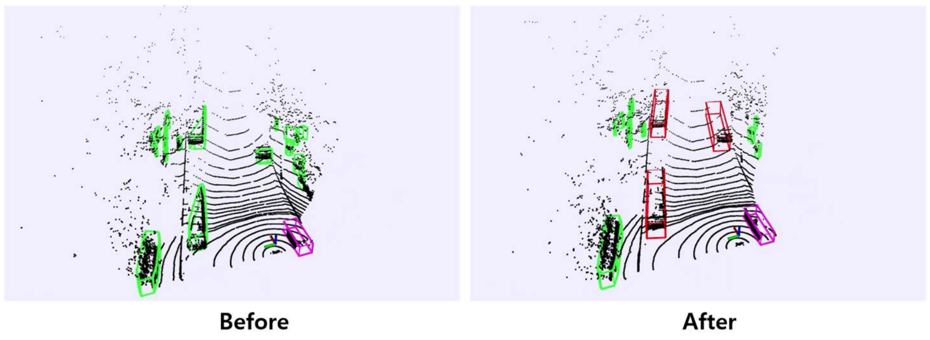

(b) After training about the bus, the network detects parked buses

Figure 6.8. The detection results after training about the bus. The detection results are shown as 3D bounding boxes on the point cloud view. The predicted boxes for car are 'pink', for bus are 'red', for cyclist are 'orange'

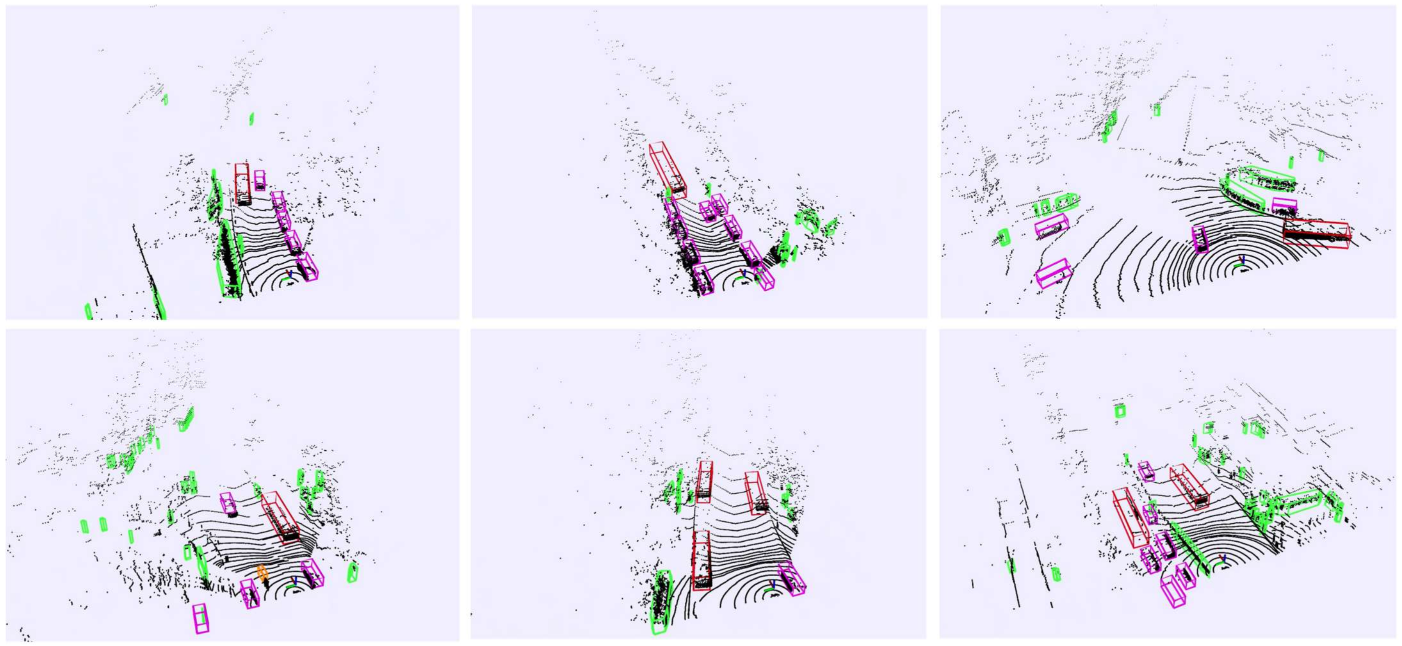
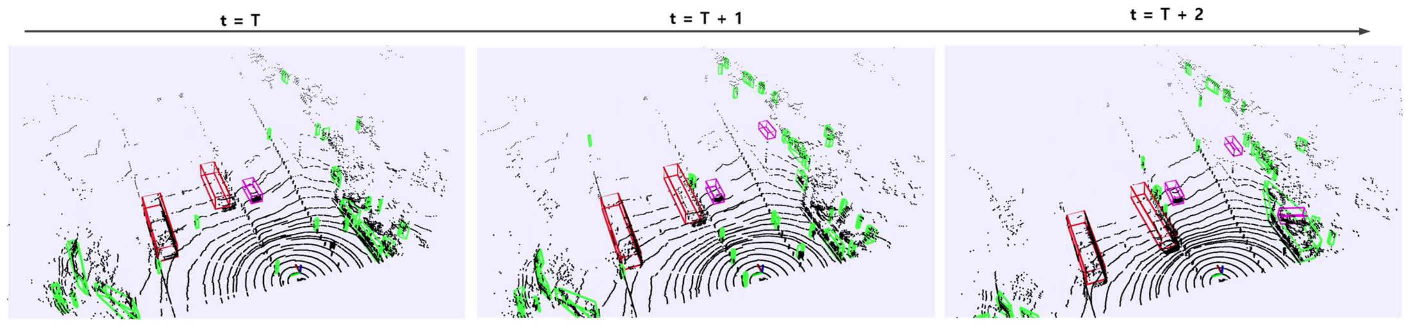
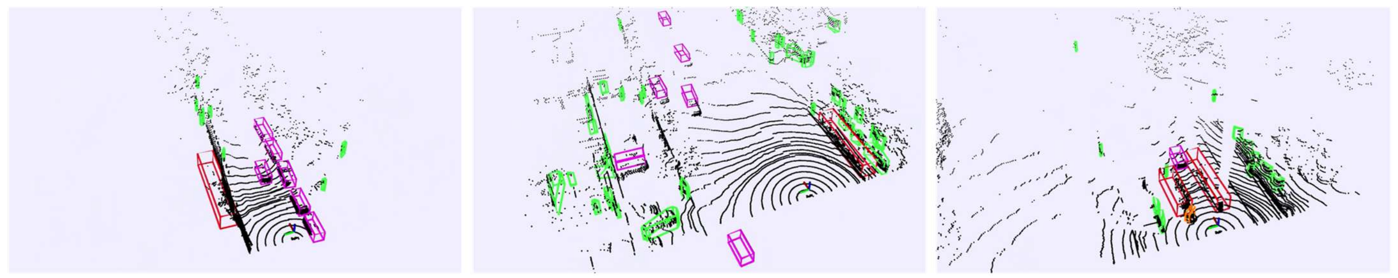


Figure 6.9. Qualitative analysis of additional training results on real-time test. We show the 3D bounding boxes on the point cloud view. The predicted boxes for car are ‘pink’, for bus are ‘red’, for cyclist are ‘orange’



(a) Size prediction change according to over time



(b) The wrong-prediction cases of non-bus objects as buses

Figure 6.10. Failure cases on real-time test with additional training. It shows predicted boxes for car (pink), bus (red), unknown object (green).

Chapter 7 Performance Evaluation

In this chapter, we introduce average precision (AP), which is the most widely used evaluation metric in the object detection field, and discuss its limitations to be used as an effective evaluation metric of practical object detection performance. Furthermore, we propose a new metric in measuring the practical performance of the perception module by considering detection priority and detection continuity.

7.1. Current Evaluation Metrics

Average precision (AP) has been a popular metric for measuring the accuracy used in the object detection field. Calculate recall and precision for each frame and sort the detection results for total frames in order of high confidence. *Precision* means how accurate are the predictions, and *recall* represents how good the detector finds all the positives.

$$\begin{aligned}\text{Precision} &= \frac{TP}{TP+FP} = \frac{TRUE \text{ detections}}{\text{Whole detections}} \\ \text{Recall} &= \frac{TP}{TP+FN} = \frac{\text{detected } TRUE}{\text{Total number of } TRUE}\end{aligned} \tag{5.4}$$

Mathematical definitions of *precision* and *recall* are shown in equation 5.4. As shown in Table 8, there are four cases in object detection results, which are *true positive*, *true negative*, *false positive* and *false negative*. Because *positive,* and *negative* means object we want to find, object we are not interested in respectively, *true positive* denotes the case we correctly find the object we want to find. Reversely, *true negative* is a case we do not find what we want to find. *False-negative* and *false positive* are can be defined in the same manner. In the case of true positive, the success of detection is determined by the Intersection of union (IoU). If IoU between object and prediction is larger than the threshold, the prediction is regarded as a success.

Table 9. Four cases of object detection

		Actual	
		Positive	Negative
Prediction	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

7.2. Limitations of Evaluation Metrics

The object classification and detection fields have been actively researched for the last few years, and the performance comparison of each network has been constantly performed. In order to compare performance between algorithms, there should be a reference dataset. PASCAL VOC and MS's COCO datasets [Lin,'14] have been used as a reference dataset in the performance evaluation of the aforementioned research area, and mAP (Mean Average Precision) has been broadly used as a representative performance index to present the network's performance. In the object classification and detection studies using the LiDAR point cloud, many researchers use KITTI benchmark as a standard dataset and mAP as a performance index to show the performance of the detection networks they proposed. It looks as a natural flow as the dimension of object detection increases from 2D to 3D. However, there is a question whether the AP is a suitable performance index in 3D object detection as it does in 2D object detection.

7.2.1. Detection Continuity

An autonomous driving vehicle should react to its surroundings in real-time, so it detects objects every single frame. Because actual driving is a sequence consists of frames, the detection results has relation previous frame's result. From this point of view, AP has a limitation that it focuses only on how many objects are detected for the entire dataset. First, there is no meaningful relationship between the previous scene and current scene when calculate the AP using KITTI dataset because the dataset scenes are randomly shuffled. Contrary to the way of calculation AP using KITTI dataset, every scene has a relation with its previous scene in actual real-time autonomous driving.

To ensure safe driving, it is necessary to keep detect objects continuously in the actual driving because detection fail may lead to traffic accident. Figure 7.1 shows the detection results of two detectors over time. *Detector A* detected more objects for both frames than *detector B*. However, the detection is not continuously performed and the object detected in time k is missed at time k+1. On the other hand, in the case of *detector B*, the number of detected objects is less than A, but it shows continuity for the detected objects. From the AP's point of view, *detector A* is considered a better detector than *detector B* because it succeeded in detecting more objects for the entire frame. However, from an actual driving perspective, it is important to maintain object detection around the vehicle.

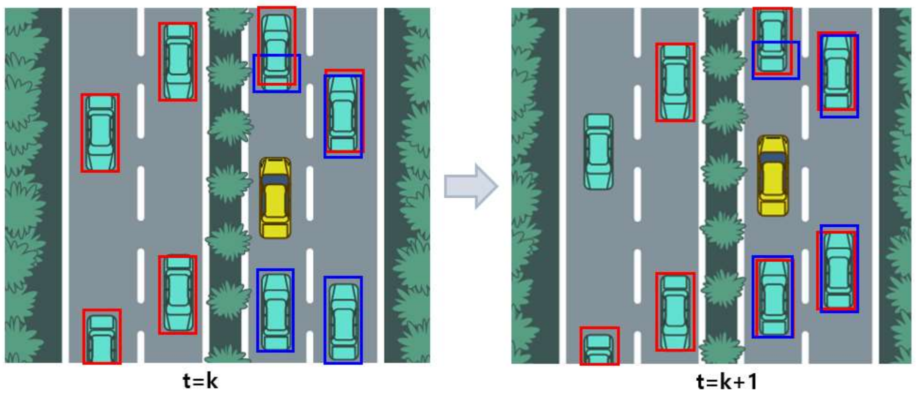


Figure 7.1. Object detection continuity

7.2.2. Detection Priority

Basically, *recall* and *precision* have a trade-off relation and they include all objects detection results in the current scene. However, from the point of view of autonomous driving control and planning, if the object is far from the subject vehicle or not within the drivable area of the subject vehicle, the detection result of the object becomes less important or does not matter at all.

In the respect of self-driving control and planning, the performance of the perception module is enough if it can detect all objects in the ROI even if it does not detect the objects outside the ROI. In other words, if all objects outside the ROI are detected but cannot detect a significant object in the ROI (for example, the vehicle in front of the subject vehicle in highway or the approaching vehicle from right-behind in case of right lane change), then the detector should be considered to perform worse than the reverse case.

In an aspect of mAP, it is a better detector to detect more objects regardless of the importance of the objects. This means that mAP has a blind point that it does not sufficiently consider the detector's performance in the practical point of view.

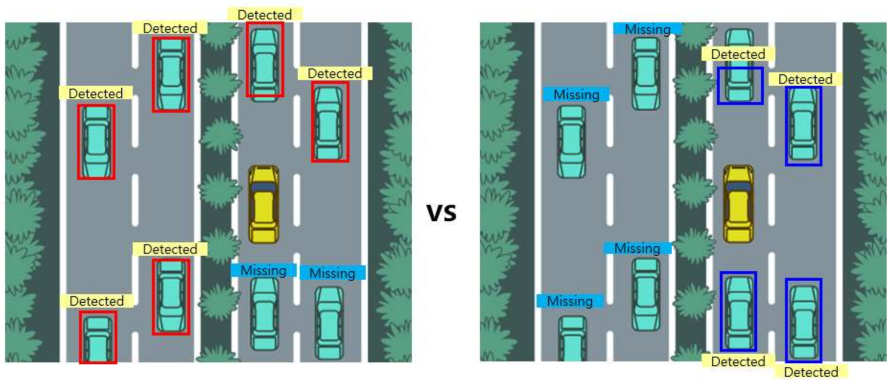


Figure 7.2. Detection priority of two cases

In actual driving, it is not a big problem even if you cannot find the object on the opposite lane. Rather, it is serious that I cannot find vehicles near the subject vehicle. It does not show how well the scene-to-scene detection continues. As such, the AP does not consider detection priority and detection continuity.

Figure 7.2 shows two object detectors. Detector A finds more objects than detector B, but it finds only half of the objects in the lane where ego vehicle exists. On the other hand, even though detector B finds no objects on opposite lanes, it finds all objects around the ego vehicle. In the actual driving situation, it is more important to find objects existing in the lanes around the ego vehicle than the other lane. The behaviors and existence of the objects around subject vehicles affect the ego vehicle's motion planning. For this reason, it is not important to simply find many objects in real driving, but it is more important to detect objects in and around the region of interest (ROI). Contrary to the calculation manner of AP on the object detection field, detection priority is considered in actual driving.

7.3. Criteria for Performance Index

In defining the performance index, criteria for determining whether or not a detector has found an object for an object should be established first. This can be defined through the Euclidean distance from the center position of the object to the detection position, but this has the disadvantage that the size and heading angle of the object cannot be considered. If only the distance is considered, even if the heading angle is detected in the opposite direction, the result is the same, so only the irrational result is obtained. Also, if the size of the object is small and large, the results are different even if they have the same Euclidean distance. Therefore, the size, heading angle, and center distance of the object should be considered as indicators for determining whether the detector has found the object. IoU is very suitable as an index of object detection correspondence that satisfies all three.

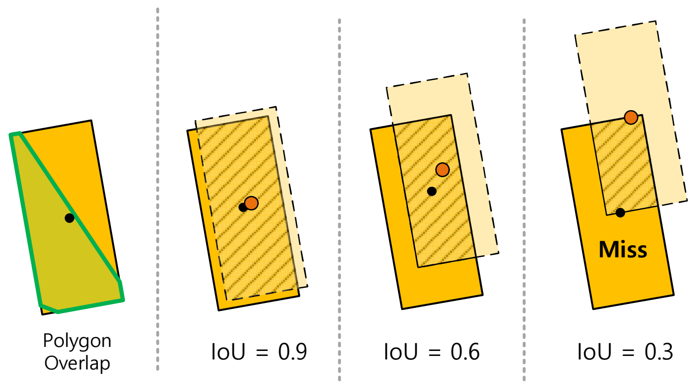


Figure 7.3. Object detection and failure

The KITTI benchmark is based on the IoU 0.7, but it can be regarded as a very strict standard for actual driving. In actual driving, a slight position error does not lead to a critical issue such as car accident. Nevertheless, the KITTI benchmark is set to a high standard of IoU 0.7, which is more strict because it considers height in the 3D detection. In fact, the criterion in the object detection field using RGB image is set to IoU 0.5 [Annotation, annotation, annotation, annotation, annotation]. Based on KITTI dataset , some researches shows the performance of their approaches by using IoU 0.5 as well as 0.7. Therefore, in a bird's eye view that does not consider height, the indicator IoU 0.5 is relatively reasonable, and it is assumed that detection failure occurs when IoU is lower than 0.5, and detection success when it is higher than 0.5. It is also regarded that object detection is successful even when it is found as an unknown object by Polygon.

In addition, when the detector detects the location where the object does not exist, it is a false positive and is deducted.

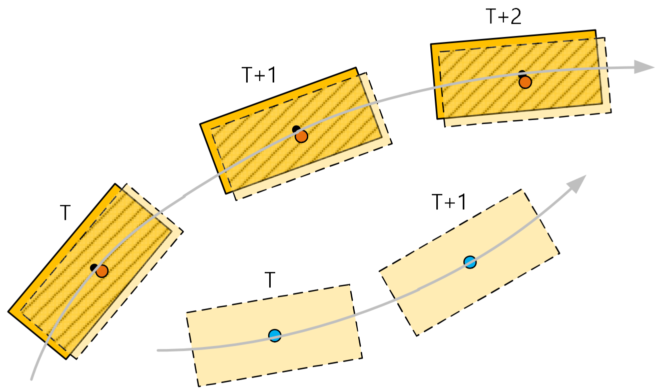


Figure 7.4. Example of false positive

In the process of object tracking, the detector continues tracking without missing the object. In this process, if the IoU exceeds 0.5, it is regarded as detection failure. Also, if an object is missed during the tracking process, it is considered a detection failure. However, if detection is detected as an unknown object through polygons while detection tracking is broken, it is considered that tracking is continuously performed.

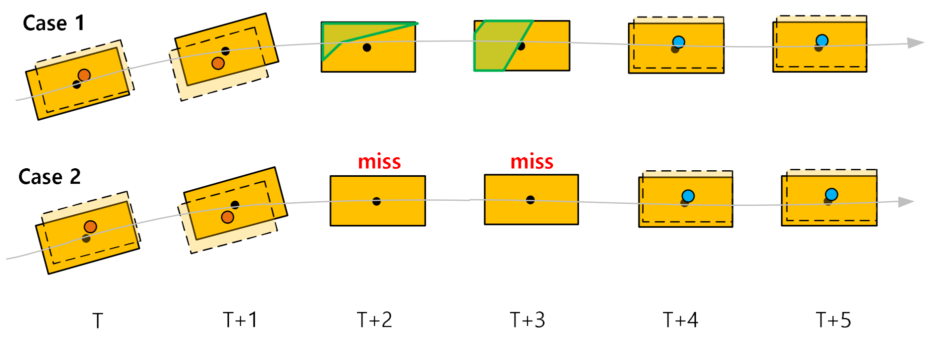


Figure 7.5. Continuous object detection

Also, it is necessary to define the detection priority. Missing an object at a short distance is a more critical miss than missing an object at a distance. Therefore, if a short-range object is missed in the process of calculating performance, more points should be deducted. That is, in the case of missing an object, the importance of missing the object is inversely proportional to the distance.

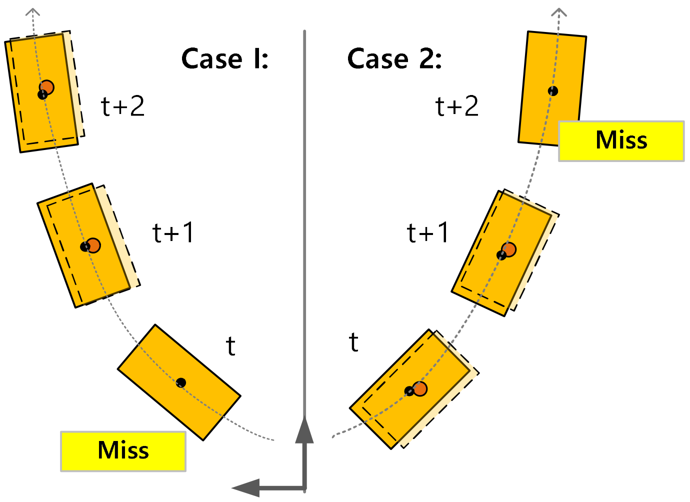


Figure 7.6. Detection priority

Finally, the closer the detection result is to the ground truth, the closer the result is to 1. The following equation is a performance index that satisfies the above criteria, and the vehicle test is verified through this performance index.

$$wMOTA = \frac{1}{N}\sum_{j=1}^{N}\left[1 - \frac{\sum_{i=1}^{N^j}\frac{1}{\omega_i^j}\left[m_i^j + fp_i^j + ids_i^j\right]}{\sum_{i=1}^{N^j}\frac{1}{\omega_i^j}}\right] \quad (5.5)$$

Chapter 8 Vehicle Tests based Performance Evaluation

In this chapter, we introduce average precision (AP), which is the most widely used evaluation index in the object detection field, and discuss its limitations to be used as a practical evaluation metric of practical object detection performance. Furthermore, we propose a new metric in measuring the

8.1. Configuration of Vehicle Tests

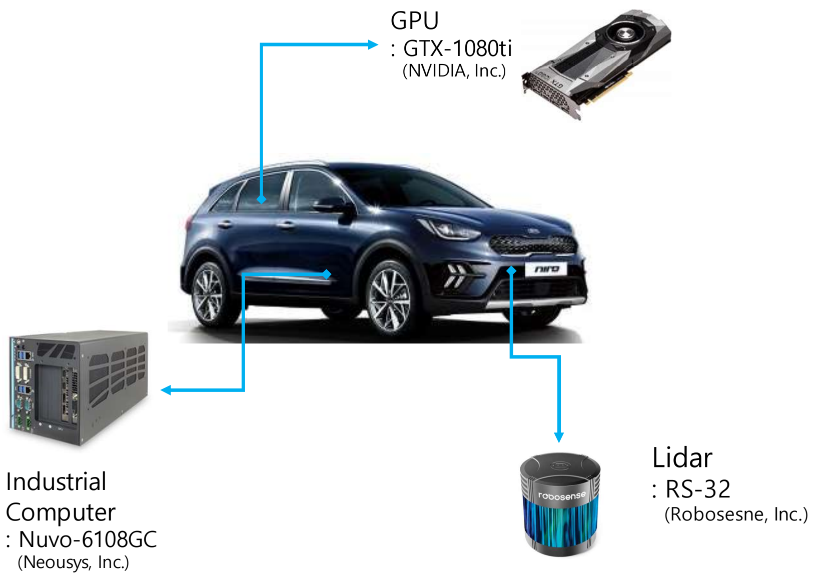
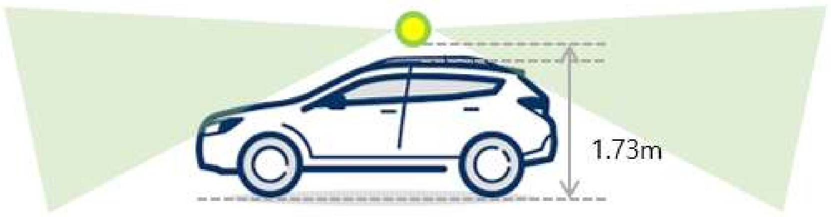
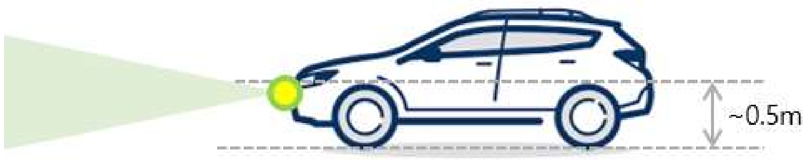


Figure 8.1. Test vehicle and Experimental setup

The above figure shows a test vehicle which is used in this study. The test vehicle is a B-segment SUV model of KIA motors (NIRO-hybrid). The vehicle tests have been conducted at the urban and inner circular road in Seoul National University, Korea. The network used in this dissertation was trained with point cloud data from the KITTI dataset, and the KITTI dataset provides point cloud gathered from Velodyne's HDL-64E products. 32 channel LiDAR was used in real vehicle test, and the product is RS-32 from Robosense. Since the actual driving test uses 32-channel LiDAR and off-line learning is done with 64-channel LiDAR, it is worth noting how much performance is achieved when using a LiDAR product with fewer channels than learning.



(a) KITTI data acquisition vehicle



(b) Test Vehicle used in this dissertation

Figure 8.2. LiDAR mount positions of KITT and test vehicle

For acceleration of the network with cuda, GTX-1080ti GPU from NVIDIA is used. This is the same product used in most of the studies ranked on the KITTI benchmark. NIRO-hybrid of KIA motors used in this experiment has different specifications with Passat of Volkswagen, the vehicle used in acquiring the KITTI dataset. Besides, in the case of the KITTI dataset, LiDAR is installed on the roof of the vehicle, while the 32-channel LiDAR is installed on the front bumper of the test vehicle. The specifications of the two vehicles and the sensors are shown in Table 9.

Table 10. Comparison of LiDAR sensors used in training and test

Specifications	Training	Test
Manufacturer	Velodyne	Robosense
Model	HDL-64E	RS-32
Channel	64	32
Field of view (Horizontal)	360 °	360 °
Field of view (Vertical)	26.9° (+2.0° to -24.9°)	40° (+15.0° to -25°)
Angular resolution (Horizontal)	0.09° to 0.18°	0.09° to 0.36°
Angular resolution (Vertical)	0.4°	Min 0.33°
Field of view update	10-20Hz	5-20Hz

Figure 8.3 shows the test driving course. The course is an urban driving environment starting from the intersection of Seoul National University Station. This leads to hills and downhill roads, leading to the main gate of Seoul National University. It is a course that leads to the back gate toward Nakseongdae Station after passing through the circular road on the campus of Seoul National University. Because passenger cars and city buses enter Seoul National University, we can face various buses and vehicles while driving through the inner ring road. The total driving distance is about 7.1km.

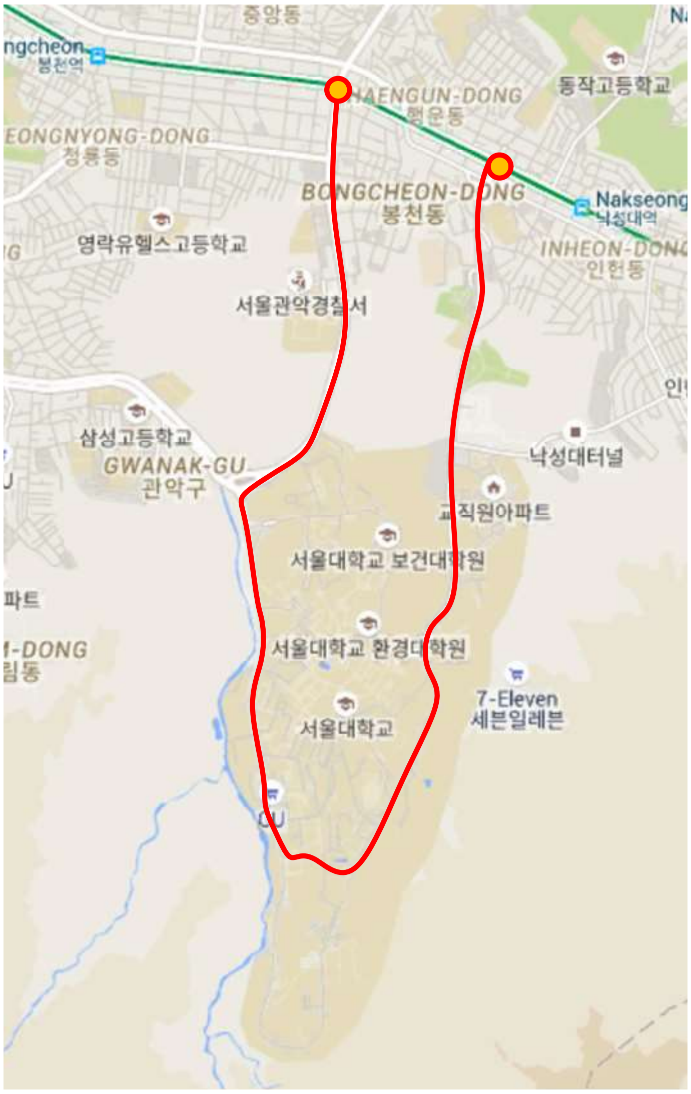


Figure 8.3. Driving course

8.2. Qualitative Analysis

The proposed perception model and reference object detector are compared through Robot operating system (ROS) bag file achieved by test driving. Figure 8.4 shows the effect of object tracking over time. As shown in the figure, the proposed method detects cars and motorcyclists and keeps track of them over time.

On the other hand, in the reference model, both the car and the motorcyclist were detected in time T, but the motorcyclist is missed in T + 1. In the next frame, which time T+2, the reference detector missed the car as well as the motorcyclist.

Because of the methodological characteristics of the object detector, the object detection result may vary depending on the confidence level even if objects have a similar shape. Thus, object detection can not be guaranteed even if it is found in the previous frame, the object may be missed in the next frame.

The tracking function compensates the frames where object detection is failed by maintaining the detection result. If a tracking function exists, even if the measurement is not received, the prediction is performed through a Kalman filter for a certain frame so that the object can be maintained without missing. If measurement information, which is detection result from the detector, is renewed during the tracking process, object tracking can be continuously performed without detection discontinuity.

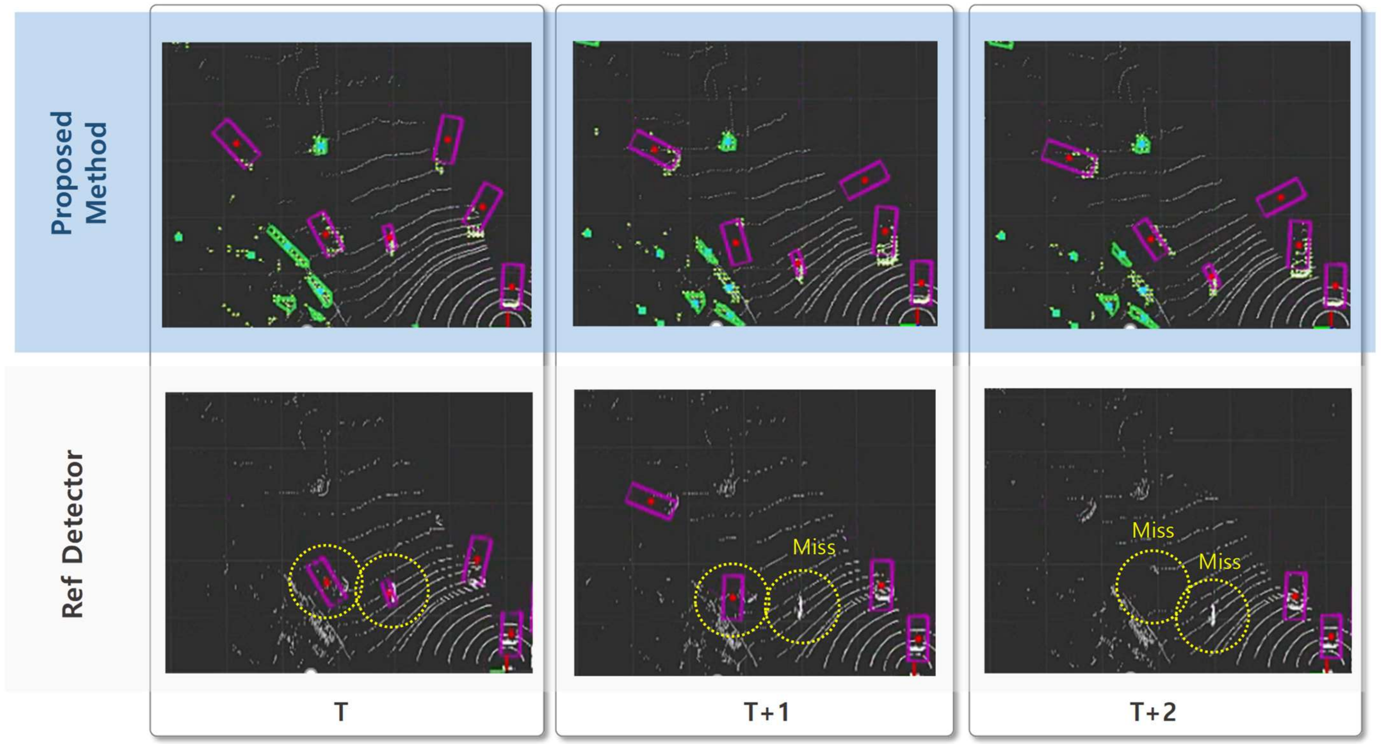


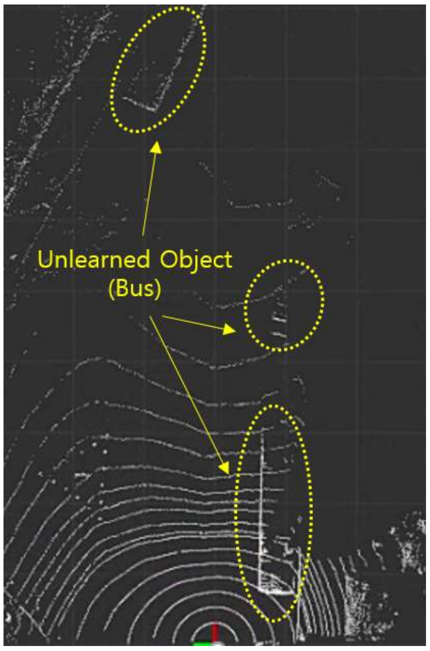
Figure 8.4. Qualitative analysis of object tracking effect

Figure 8.5 shows a scene of experimental results of the proposed approach and the reference detector when the bus is not trained. In this scene, the effect of unknown object detection through non-ground point extraction and clustering can be seen.

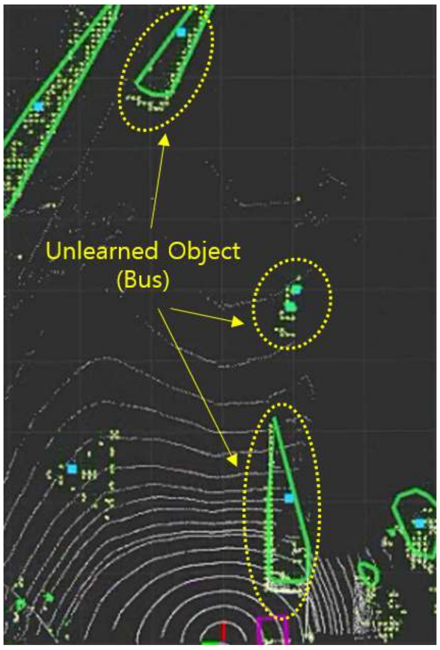
As shown in the left of the figure, the proposed approach finds points that are considered to belong to obstacles on the road by using ground extraction. after that, boundary polygons of unknown objects are obtained through clustering using these non-ground points.

In the figure, in addition to the bus, other unknown objects or areas where ego vehicle should detour such as the flower beds, trees, and the poles are exist.

On the other hand, in the case of the reference detector, it was not able to detect a bus because it is not trained about the bus. Unlike a car or van, the bus has a very large size and a shape closer to a cuboid box, so it is very difficult to recognize the bus as a car or van. Even if, the bus is luckily recognized as a vehicle, it will fail by losing detection continuity for the reason mentioned above because it lacks the tracking function.



(a) Does not detect unknown object



(b) Detect unknown object as polygon

Figure 8.5. Qualitative analysis of unknown object detection

Both tracking and unknown object detection appear simultaneously in Figure 8.6. For pedestrians, the reference detector finds one at time T, and detects all three pedestrians in the next frame. However, it fails to secure detection continuity by loosing one pedestrian again at T + 2. On the contrary, in the case of the proposed approach, the detection was continuously maintained for the first two frames by keeping detecting two pedestrians, and the remaining one is additionally detected in time T + 2.

In the figure, a bus on the left approaches over time, and the distance between the bus and ego vehicle is gradually closer.

As mentioned above, because the reference detector is not trained about the bus, it was not detected even though the bus is close.

The proposed algorithm is also not trained about the bus. On the other hand, the method proposed in this paper detects the bus as an unknown object and finds boundary information in the form of a polygon.

The poles are lined up on either side of the road in this driving course. Unlike the reference detector, the perception algorithm in this paper also finds them. Furthermore, the current road is an uphill road, and if a point belonging to the ground is incorrectly filtered as a non-ground point in the process of ground extraction, it may be detected as an unknown object, and false positives may be generated. But, as shown in the figure, there is no false-positive result because non-ground extraction goes well even on the slope.

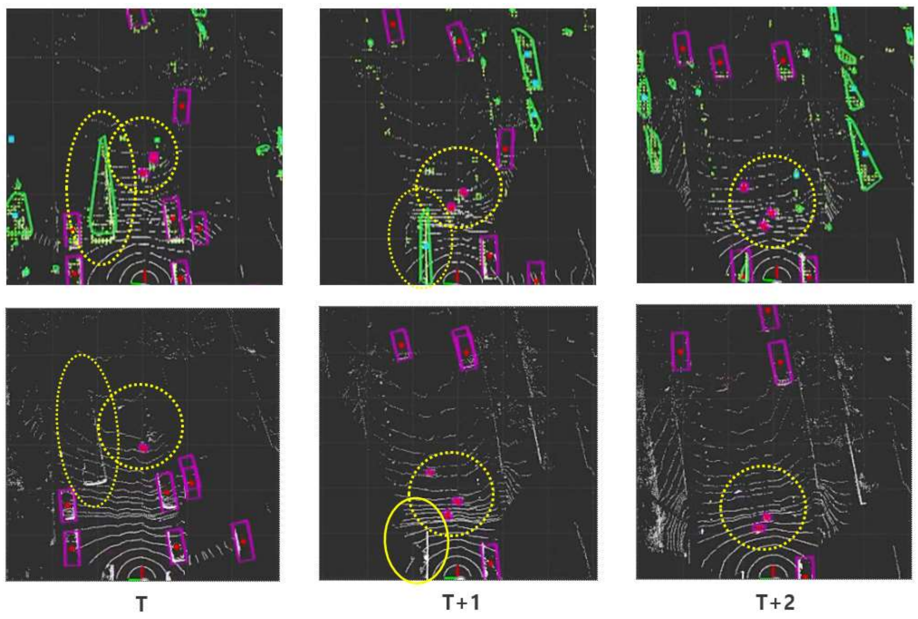


Figure 8.6. Qualitative analysis of test driving

8.3. Quantitative Analysis

Test results are analyzed through the weighted multi-object tracking accuracy defined above. The performance comparison of the proposed method and reference detector is calculated based on wMOTA. Both networks are trained on cars, pedestrians, and cyclists. Object detection is considered successful if they find the object and classify them as anything among the three categories. In case that they classify it as an unknown object, detection is also considered successful. In addition, the effect of false negative is included in the score to reflect the role of continuous detection. The more false negatives occur, the greater the deduction

Table 11 shows a quantitative comparison of the two networks. The proposed method scored higher than the reference model when it comes to wMOTA. This is a penalty for missing objects in the middle because the reference model lacks the tracking function, while the proposed method can maintain detection without detection missing between frames even when detection measurement information is not renewed through tracking. Also, in the case of buses, both models were not trained. In the reference model, not only the bus was not detected, but sometimes it is limited to a few frames when it is detected as a car. On the other hand, in the proposed model, the bus can be detected as unknown object detection, so it scored higher than the reference model in wMOTA.

For the maximum detection range, the reference model has a longer detection range, and it is related to the tracking function. When the object is tracked, the maximum detection distance may be shorter than the reference model that detects immediately because the object is not immediately assigned an ID when an object is detected, and an ID is assigned when more than a certain frame is continuously detected.

Table 11. Comparison of approaches based on proposed performance metric

Specifications	Proposed Method	Ref Detector
wMOTA	**0.9311**	0.8508
Maximum detection range	62 m	**66 m**

Chapter 9 Conclusions and Future Works

This dissertation has proposed a LiDAR-based autnomous driving perception module based on a neural network method and point cloud clustering method. The proposed module is a kind of hybrid object detector and tracker using neural network and point clustering simultaneously. In addition, a new performance index was proposed by supplementing the unpractical part of the performance index of the existing object detection, and the vehicle test result was validated via this index.

Most object detection studies using the existing lidar point cloud have utilized supervised learning techniques. This has a methodological limitation that objects not learned in the learning process cannot be found in the actual infer process because the network learns from the data set. Also, since these datasets have different characteristics for each country and region, it is impossible to prepare and learn appropriate datasets for all situations.

In addition, in the actual driving process, there are many obstacles that have not been learned on the road, so it is impossible to cope with infinite cases with the learning method through the dataset. In other words, we needed to overcome the data-dependent limitations of supervised learning.

Therefore, in this dissertation, through the non-ground point extraction, the points considered as objects can be selected and clustered to find the unknown objects in the form of unknown objects. In addition, continuous object detection was performed through clustering and object tracking for objects that could be missed during the object detection process. Through these methods, we have succeeded in detecting various objects that can be encountered in actual driving than the existing object detector.

In addition, the items evaluating the performance of the existing object detector did not consider the detection priority and continuity, so they could not show practical performance. In this study, we proposed a new performance index considering these, and the performance of the perception module of this study was verified through this performance index and vehicle test.

For the future works, More rigorous performance index can be proposed by selecting the ROI that should be more important in the driving process. In addition, it will be possible to examine how much the performance index can guarantee the reliability of actual autonomous driving through interworking with control and planning modules.

Bibliography

LIU, W., ANGUELOV, D., ERHAN, D., SZEGEDY, C., REED, S., FU, C.-Y., BERG, A. C. 2016. SSD: single shot multibox detector. *European conferecne on computer Vision (ECCV)*, 21-37.

GIRSHICK, R, B. 2015. Fast R-CNN. *International conference on computer vision (ICCV)*, 1440-1448.

REN, S., HE, K., GIRSHICK, R, B. & SUN, J. 2015, Faster R-CNN: towards real-time object detection with region proposal networks. *Neural information processing systems (NIPS)*, 91-99.

REDMON, J., DIVVALA, S., GIRSHICK, R. & FARHADI, A. 2015. You only look once: Unified, real-time object detection, *Computer vision and pattern recognition (CVPR)*, IEEE, 779-788

REDMON, J., FARHADI, A. 2016. YOLO9000: better, faster, stronger. *Computer vision and pattern recognition (CVPR)*, IEEE, 6517-6525

CHEN, X., MA, H., WAN, J., LI, B., XIA, T. 2017, Multi-view 3d object detection network for autonomous driving, *Computer vision and pattern recognition (CVPR)*, IEEE, 1, 2, 5, 6.

KU, J., MOZILFIAN, M., LEE, J., HARAKEH, A., WASLANDER, S. 2018, Joint 3d proposal generation and object detection from view aggregation, *International conference on intelligent robots and systems (IROS),* IEEE/RSJ, 1, 2, 5, 6.

LI, B., ZHANG, T., XIA, T., 2016, Vehicle detection from 3d lidar using fully convolutional network, *Computer vision and pattern recognition (CVPR)*, IEEE, 2.

SIMON, M., MILZ, S., AMENDE, K., GROSS, H. 2018, Complex-YOLO: Real-time 3d object detection on point clouds, *arXiv:1803.06199*, 1, 2, 8.

YANG, B., LUO, W., URTASUN, R. 2018, PIXOR: Real-time 3d object detection from point clouds, *Computer vision and pattern recognition (CVPR)*, IEEE, 1, 2, 6, 8.

ENGELCKE, M., RAO, D., WANG, D., TONG, C., POSNER, I., 2017, Vote3deep: Fast object detection in 3d poin tclouds using efficient convolutional neural networks, *International conference on robotics and automation (ICRA)*, IEEE, 2.

QI, C., SU, H., MO, K., GUIBAS, L. 2017, Pointnet: Deep learning on point sets for 3d classification and segmentation, *Computer vision and pattern recognition (CVPR)*, IEEE, 2, 3.

ZHOU, Y., TUZEL, O. 2018, Voxelnet: End-to-end learning for point cloud based 3d object detection, *Computer vision and pattern recognition (CVPR)*, IEEE, 1-8.

QI, C., LIU, W., WU, C., SU, H., GUIBAS, L. 2018, Frustum pointnets for 3d object detection from rgb-d data, *Computer vision and pattern recognition (CVPR)*, IEEE

LEHNER, J., Nessler, B., Hochreiter, S., 2019, Patch refinement - localized 3d object detection, *CoRR, abs/1910.04093*.

SHI, S., WANG, X., LI, H., 2019, Pointrcnn: 3d object proposal generation and detection from point cloud, *Computer vision and pattern recognition (CVPR)*, IEEE, 770-779.

TIPALDI, C., DEWAN, G., Burgard, W. 2016, Motion-based detection and tracking in 3d lidar scans, *International conference on robotics and automation (ICRA)*, IEEE, 4508-4513.

WENG, X., KITANI, K., 2019, A baseline for 3d multi-object tracking, *arXiv:1907.03961*, 1-3.

WANG, Z., JIA, K. 2019, Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection, *International conference on intelligent robots and systems (IROS),* IEEE/RSJ, 3-6.

CHEN, Y., LIU, S., SHEN, X., JIA, J. 2019, Fast point r-cnn, *International conference on computer vision (ICCV)*, IEEE, 2-6.

YANG, Z., SUN, Y., LIU, S., SHEN, X., JIA, J. 2019, STD: sparse-to-dense 3d object detector for point cloud, *International conference on computer vision (ICCV)*, IEEE, 2-8

SHI, S., WANG, Z., WANG, X., LI, H., 2019, Part-a^2 net: 3d part-aware and aggregation neural network for object detection from point cloud, *arXiv:1907.03670*, 2, 3.

LANG, A., VORA, S., CAESAR, H., ZHOU, L., YANG, J., BEIJBOM, O. 2019, Pointpillars: Fast encoders for object detection from point clouds, *Computer vision and pattern recognition (CVPR)*, IEEE, 1-7.

YAN, Y., MAO, Y., LI, B. 2018, Second: Sparsely embedded convolutional detection, 18(10), 1, 2, 3, 5, 6, 7, 8.

LIN, T., GOYAL, P., GIRSHICK, R., HE, K., DOllAR, P, 2017, Focal Loss for Dense Object Detection, , *Computer vision and pattern recognition (CVPR)*, IEEE, 2.

REDMON, J., FARHADI, A., 2018, YOLOv3: An Incremental Imrpovement, 2, 4

IANDOLA, F., HAN, S., MOSKEWICZ, M., ASHRAF, K., DALLY, W., KEUTZER, K., 2016, SqueezNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, 2

ALI, W., ABDELKARIM, S., ZIDAN, M., ZAHRAN, M., SALLAB, A. 2018, Yolo3d: End-to-end real-time 3d oriented object bounding box detection from lidar point cloud, *European conference on computer vision (ECCV).*

BELTRAN, J., GUINDL, C., MORENO, F. 2018, BirdNet: a 3d object detection framework from lidar information, *International conference on intelligent transportation systems (ITSC).*

IOFFE, S., SZEGEDY, C. 2015, Batch Normalization: Aceelrationg deep network training by reducing internal covariate shift, *arXiv:1502.03167.*

CHU, P., CHO, S., SIM, S., KWAK, K., CHO, K. 2017, A Fast ground segmentation method for 3d point cloud, *Journal of information processing system (JIPS).*

ZERMAS, D., IZZAT, I., PAPANIKOLOPOULOS, N. 2017, Fast segmentation of 3d point clouds: a paradigm on lidar data for autonmous vehicle applications, *International conference on robotics and automation (ICRA)*, IEEE.

CHO, S., KIM, J., IKRAM, W., CHO, K., JEONG, Y., UM, K., SIM, S. 2014, Sloped terrain segmentation for autonomous drive using sparse 3D point cloud, *Sci World*, 1-10.

ZHANG, M., MPRRIS, DD., FU, R., 2015, Ground segmentation based on loopy belief propagation for sparse 3d point clouds, *International conference on 3D vision*, 615-622.

ASVADI, A., PREMEBIDA, C., PEIXOTO, P., NUNES, U. 2016, 3D lidar-based static and moving obstacle detection in driving environments: an approach based on voxels and multi-region ground planes, r*obotics and autonomoys system*, Elsevier, 299-311.

MOOSMANN, F., PINK, O., STILLER, C. 2009, Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion, *Intelligent vehicle symposium,* IEEE, 215-220.

VELAS, M., SPANEL, M., HRADIS, M., HEROUT, A., 2017, CNN for very fast ground segmentation in velodyne lidar data, *arXiv: 1709.02128*.

CHOI, J., ULBRICH, S., LICHTE, B., MAURER, M., 2013, Multi-target tracking using a 3d-lidar sensor for autonomous vehicles, *Conference on intelligent transportation systems*, IEEE, 881-886.

DABBIRU, L., GOODIN, C., SCHERRER, N., CARRUTH, D. 2020, Lidar data segmentation in off-road environment using convolutional neural network, *SAE world congress*, IEEE.

WANG, H., LOU, X., CAI, Y., CHEN, L. 2018, A 64-line lidar-based road obstacle sensing algorithm for intelligent vehicles, *Scientific programming*, Hindawi.

YOO, J., LEE, C., YANG, M., YOON, K. 2016, Online multi-object tracking via structural contraint event aggregation, *Computer vision and pattern recognition (CVPR)*, IEEE.

SHARMA, S., ANSARI, J., MURTHY, J., KRISHNA, K. 2018, Beyond Pixels: leveraging geometry and shape cues for online multi-object tracking, *International conference on robotics and automation (ICRA)*, IEEE.

Kalman, R., 1960, A new approach to linear filtering and prediction problems, Journal of basic engineering.

GEIGER, A., LENZ, P., STILLER, C., URTASUN, R. 2013, Vision meets robotics: the kitti dataset, *International journal of robotics research (IJRR)*, 32(11):1231-1237.

LIN, T., MAIRE, M., HAYS, B., PERONA, P., RAMANAN, D., DOLLAR, P., ZITNICK, C. 2014, Microsoft COCO: common objects in context, *European conferecne on computer Vision (ECCV)*, 2.

YANG, B., LIANG, M., URTASUN, R. 2018, HDNET: Exploting HD maps for 3d object detection, *Conference on robot learing (CoRL)*, 1, 6.

SHIN, K., KWON, Y., TOMIZUKA, M. 2018, Roarnet: A robust 3d object detection based on region approximation refinement, *arXiv: 1803.06199*, 1, 2, 8.

YANG, Z., SUN, Y., LIU, S., SHEN, X., JIA, J. 2018, IPOD: intensive point-based object detector for point cloud, *arXiv: 1812.05276*.

LIANG, M,, YANG, B., WANG, S., URTASN, R. 2018, Deep continuous fusion for multi-sensor 3d object detection, *European conferecne on computer Vision (ECCV)*.

ROSIQUE, F., NAVARRO, P., FERNANDEZ, C. & PADILLA, A. 2019, A systematic review of perception system and simnulators for autonomous vehicles research, *Sensors*, vol. 19, no. 3, p. 648.

MEYER, G., LADDHA, A., KEE, E., VALLESPI-GONZALEZ, C., WELLIGNTON, C. 2019, Lasernet: An efficient probabilitic 3d object detector for autonomous driving, *Computer vision and pattern recognition (CVPR)*, IEEE.

LIANG, M., Yang, B., CHEN, Y., HU, R., URTASUN, R., 2019, Multi-task multi sensor fusion for 3d object detection, *Computer vision and pattern recognition (CVPR)*, IEEE.

초 록

실시간 자율주행 인지 시스템을 위한 신경망 네트워크와 군집화 기반 미학습 물체 감지기 통합

최근 몇 년간, 센서 기술의 발전과 컴퓨터 공학 분야의 성과들로 인하여 자율주행 연구가 더욱 활발해지고 있다. 자율주행 시스템에 있어서 차량 주변 환경을 인식하는 것은 안전 및 신뢰성 있는 주행을 하기 위해 필요한 가장 중요한 기능이다. 자율주행 시스템은 크게 인지, 판단, 제어로 구성되어 있는데, 인지 모듈은 자율주행 차량이 경로를 설정하고 판단, 제어를 함에 앞서 주변 물체의 위치와 움직임을 파악해야하기 때문에 중요한 정보를 제공한다.

자율주행 인지 모듈은 주행 환경을 파악하기 위해 다양한 센서가 사용된다. 그 중에서도 LiDAR은 현재 많은 자율주행 연구에서 가장 널리 사용되는 센서 중 하나로, 물체의 거리 정보 획득에 있어서 매우 유용하다.

본 논문에서는 LiDAR에서 생성되는 포인트 클라우드 raw 데이터를 활용하여 장애물의 3D 정보를 파악하고 이들을 추적하는 인지 모듈을 제안한다. 인지 모듈의 전체 프레임워크는 크게 세

단계로 구성된다. 1단계는 비지면 포인트 추정을 위한 마스크 생성, 2단계는 특징 추출 및 장애물 감지, 3단계는 장애물 추적으로 구성된다.

현재 대부분의 신경망 기반의 물체 탐지기는 지도학습을 통해 학습된다. 그러나 지도학습 기반 장애물 탐지기는 학습한 장애물을 찾는다는 방법론적 한계를 지니고 있다. 그러나 실제 주행 상황에서는 미처 학습하지 못한 물체를 마주하거나 심지어 학습한 물체도 놓칠 수 있다. 인지 모듈의 1단계에서 이러한 지도학습의 방법론적 한계에 대처하기 위해 포인트 클라우드를 일정한 간격으로 구성된 3D 복셀(voxel)로 분할하고, 이로부터 비접지 점들을 추출한 뒤 미지의 물체(Unknown object)를 탐지한다.

2단계에서는 각 복셀의 특성을 추출 및 학습하고 네트워크를 학습시킴으로써 객체 감지기를 구성한다. 마지막 3단계에서는 칼만 필터와 헝가리안 알고리즘을 활용한 다중 객체 탐지기를 제안한다. 이렇게 구성된 인지 모듈은 비지면 점들을 추출하여 학습하지 않은 물체에 대해서도 미지의 물체(Unknown object)로 감지하여 실시간으로 장애물 탐지기를 보완한다.

최근 라이다를 활용한 자율주행 용 객체 탐지기에 대한 연구가 활발히 진행되고 있으나 대부분의 연구들은 단일 프레임의 물체 인식에 대해 집중하여 정확도를 올리는 데 집중하고 있다. 그러나 이러한 연구는 감지 중요도와 프레임 간의 감지 연속성 등에 대한 고려가 되어있지 않다는 한계점이 존재한다. 본 논문에서는 실시간 성능을 얻기 위해 이러한 부분을 고려한 성능 지수를 제안하고, 실차 실험을 통해 제안한 인지 모듈을 테스트, 제안한 성능 지수를

통해 평가하였다.

주요어: 자율 주행, 라이다 센서, 인공 신경망, 객체 감지, 다중 객체 추적

학 번: 2013-20687