Sensor Data Fusion of Lidar and Camera for Road User Detection

Xinyu Gao

Master of Science Thesis

TU Delft Delft University of Technology

Sensor Data Fusion of Lidar and Camera for Road User Detection

Master of Science Thesis

For the degree of Master of Science in Vehicle Engineering at Delft University of Technology

Xinyu Gao

August 3, 2018

Student number: 4548604
Project duration: September, 2017 – July, 2018
Supervisors: Prof. dr. D. M. Gavrila, TU Delft
ir. J. F. M. Domhof, TU Delft

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of
Cognitive Robotics

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Sensor Data Fusion of Lidar and Camera for Road User Detection

by

Xinyu Gao

in partial fulfillment of the requirements for the degree of

Master of Science Vehicle Engineering

Dated: August 3, 2018

Supervisor(s):

Prof. dr. D. M. Gavrila

Ir. J. F. M. Domhof

Reader(s):

Dr. J. F. P. Kooij

Dr. W. Pan

Abstract

Object detection is one of the most important research topics in autonomous vehicles. The detection systems of autonomous vehicles nowadays are mostly image-based ones which detect target objects in the images. Although image-based detectors can provide a rather accurate 2D position of the object in the image, it is necessary to get the accurate 3D position of the object for an autonomous vehicle since it operates in the real 3D world. The relative position of the objects will heavily influence the vehicle control strategy. This thesis work aims to find out a solution for the 3D object detection by combining the Lidar point cloud and camera images, considering that these are two of the most commonly used perception sensors of autonomous vehicles. Lidar performs much better than the camera in 3D object detection since it rebuilds the surface of the surroundings by the point cloud. What's more, combing Lidar with the camera provides the system redundancy in case of a single sensor failure. Due to the development of Neural Network (NN), past researches achieved great success in detecting objects in the images. Similarly, by applying the deep learning algorithms to parsing the point cloud, the proposed 3D object detection system obtains a competitive result in the KITTI [1] 3D object detection benchmark.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor Prof. dr. D. M. Gavrila for his support during the thesis work. He provides a lot of helpful suggestions on how to conduct a scientific research. The the powerful TITAN V GPU supplied by him also makes it possible for my proposed idea to be implemented. I also have to say much thanks to my daily supervisor ir. J. F. M. Domhof. Whenever I came across a problem that I cannot solve, he can always give me some suggestions on how to solve it and how to see the problem from a different aspect. He is always patient and inspiring when I asked him for help. I think the most important thing that he taught me is that how to find out the key problem and how to make a plan to solve the problem.

Finishing this master thesis is quite a big challenge to me. Through the thesis research, I figure out how to conduct a scientific research towards a specific problem. It is always hard to conduct a research alone. However, the school provides a lot of resources helping me out. The courses from the first year really equip me with the knowledge required for the thesis work. There are always professors and colleagues willing to give some generous help.

I am so glad that I chose to study here in TU Delft for my master's degree two years ago. I really had a great time here in the Netherlands. I will never forget the beautiful landscape and lovely people here.

Delft, University of Technology — Xinyu Gao
August 3, 2018

Chapter 1

Introduction

1-1 Background

Autonomous vehicle has been a hot topic both in the research field and in the industry during the last few years. The development of the new technology has also made a great impact on the public. The autonomous vehicles will definitely change the lifestyle of people all over the world.

Autonomous vehicle is not a new concept, it starts in the 1980s [19] and has been widely researched during the past 30 years. There are a lot of advantages the autonomous vehicle can bring us. Autonomous vehicles will greatly reduce of the price of the transportation. If people all use the shared autonomous vehicles for the transportation, there will be less CO_2 emission. What's more, the autonomous vehicle will make the city a better place for humans to live in instead of a place where cars stuck everywhere. Autonomous vehicles will also enhance the road safety and cause fewer fatalities than the human-driving vehicles.

Early products in the form of Advanced Driver-Assistance Systems (ADAS) have already been applied to the vehicles nowadays. The two of the most widely used ADAS are Lane Keeping Assistance (LKA) and Adaptive Cruise Control (ACC). LKA is a system helping to control the heading of the vehicle [20] and ACC helps the car to keep a safe distance from the front car. The ADAS is attracting large expectations and undergoing vast improvements in recent years. The goal of such a system is to protect both the occupants and the Vulnerable Road Users (VRU) to enhance the road safety and to make the driving easier. With the development of the state-of-the-art technologies, the ADAS on board nowadays makes the vehicle closer to the real autonomous vehicle. However, there are still some technical gaps between those ADAS systems and a real autonomous vehicle. One of the most important challenge is that the autonomous vehicle requires a great perception system to perceive the environment. Only when the autonomous vehicle has a great perception system, the vehicle control system can take a meaningful control strategy.

The perception module has to perform multiple tasks including building a detailed map, localization, recognizing objects and so on. This thesis work will mainly focus on the object

recognition task. Knowing where and what the obstacles are is rather import for an autonomous vehicle, especially when the obstacles are VRU like pedestrians and cyclists. Any recognition failure might arise serious consequences like the personal injury or death.

1-2 Sensors and Data Format

The perception systems for an autonomous vehicle nowadays are mainly equipped with three types of sensors: camera, radar and Lidar. This thesis work will focus on using the camera and Lidar for the object recognition. The camera is an optical instrument for recording or capturing images [21] and is the most widely used sensor for the perception task because it is accurate, cheap and quite similar to the way how humans perceive the world. Most Lidar used on the autonomous vehicle is a rotating sensor that keeps transmitting lasers (light amplification by stimulated emission of radiation) and receiving the reflected lasers. The interval between the transmitted laser and reflected laser can be used to determine the distance between the obstacle and the sensor. The interior rotating mirror distributes the laser ray in any direction, which provides Lidar with a 360° field of view. Lidar provides the point cloud of the surroundings as shown in Figure 1-1. Each point represents the relative position of the obstacle surface point to the sensor.

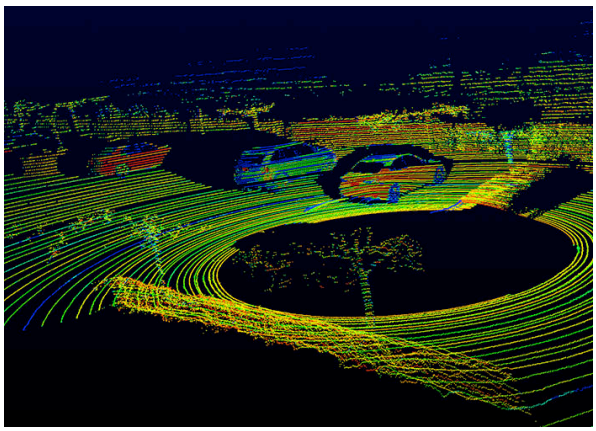
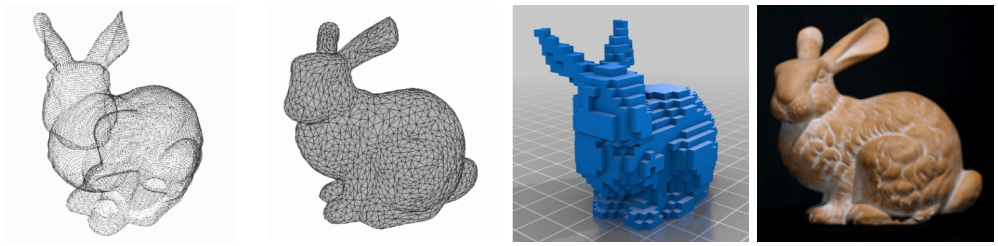


Figure 1-1: Point cloud

Due to the development of machine learning algorithms in the field of Computer Vision (CV), algorithms to detect objects in the camera image are becoming more mature, which makes the image-based detection results rather reliable. The image-based detection algorithms like Dalal-Triggs [16], Single-Shot-Detector (SSD) [6], Fast R-CNN [22], Faster R-CNN [5], Mask R-CNN [23], etc., have achieved great success over the years. On the other hand, the strong computational capability of the new Graphics Processing Unit (GPU) like NVIDIA TITAN V, NVIDIA TITAN XP, etc., and even some AI platform designed for autonomous driving like NVIDIA DRIVE PX2 makes these complex algorithms to be possible to run in real time with the help of CUDA. CUDA is a parallel computing platform and programming model that enables dramatic increases in computing performance by harnessing the power of GPU [24]. However, a big problem the camera is facing is that it suffers from illumination changes and weather conditions. The image-based detectors sometimes perform really bad under certain illumination conditions like the dark night. Another shortage is that camera is not

good at getting the relative 3D position of the object to the sensor. Since the autonomous vehicle operates in the real 3D world, perception systems equipped with only cameras are not sufficient for the autonomous driving.



(a) Point Cloud **(b)** Mesh **(c)** Volumetric **(d)** Projected View

Figure 1-2: Common representations of 3D point clouds [2]

The Lidar point cloud provides the accurate depth information which can be used for 3D object detection and it performs better than the camera in the localization of the object [25]. Since the point cloud is composed of unordered and scattered points, there are several commonly used representations for such 3D data as shown in Figure 1-2. Among all the related works trying to implement object detection algorithms in the point cloud, most of them choose to project the point cloud onto the image plane to get an RGB-D image (RGB image with depth information for each pixel) as shown in Figure 1-2d. Other works like VoxelNet [9] discretizes the point cloud into 3D voxel grids and represents each grid with the feature of those points inside the grid as shown in Figure 1-2c. Recent researches like Pointnet [2], Pointnet++ [26] create a novel type of Neural Network (NN) that directly consumes the point cloud as shown in Figure 1-2a. The point cloud is probably the closest 3D representation to the raw sensor data so it is the most expected representation when dealing with object detection tasks.

Combining the point cloud and image information can make the most of both sensors and provide a more convincing detection results, which is also the research topic of this thesis.

1-3 Problem Statement

There are two research questions in the thesis: **What is the best solution to build a 3D object detector?** and **What is the best solution to fuse the data from Lidar and camera for 3D object detection?** For now,there are mainly two types of sensors that the existing 3D object detection algorithms are using: stereo cameras and Lidar. Lidar can easily rebuild the 3D surface of the surroundings. However, for the stereo vision, two cameras have to see the same pixel at the same time, which might be hard when occlusion occurs. What's more, for detecting objects far away, two cameras have to be further apart, which makes the stereo vision less advantageous. So, the Lidar is used in the thesis for 3D object detection and the first research question becomes **What is the best solution to build a Lidar-based 3D object detector?** in the thesis context.

The Lidar-based object detection is still a new research topic. A few years ago, after the research on autonomous vehicle becomes popular, more and more researchers start using Lidar for object detection tasks since it retrieves more accurate 3D information of the surroundings.

However, the object detection algorithms in the Lidar point cloud so far are still not good enough.

The sensor data fusion of Lidar and camera for the object recognition is also a new research topic. Generally, there are three main types of fusion structure as shown below in the Figure 1-3. The three main types of fusion are:

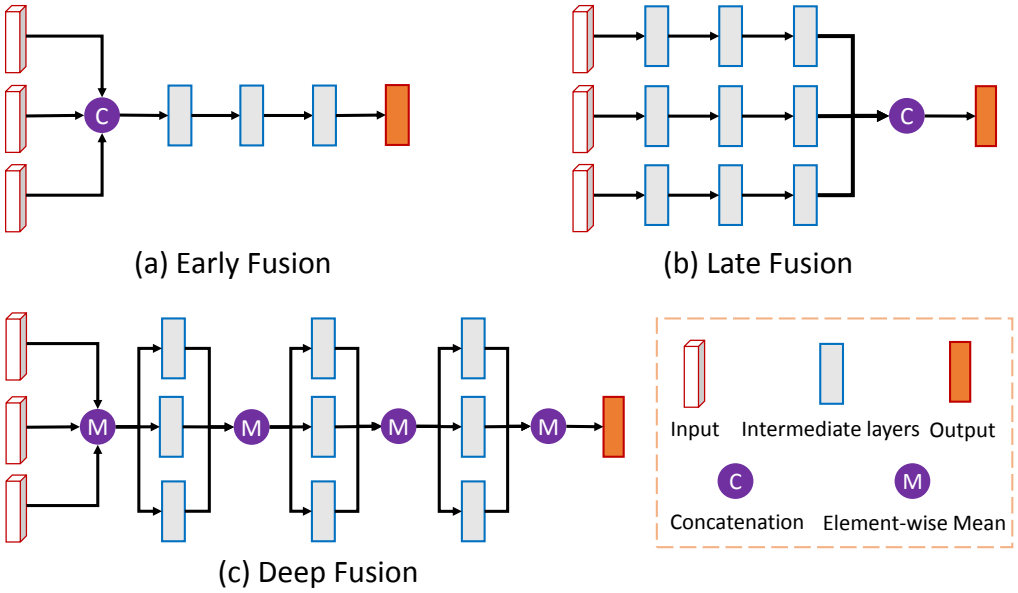


Figure 1-3: Different fusion structures [3]

- `Early fusion`
  In an early fusion structure, only one classifier is used in the system. The feature is extracted from multiple sensors. The system might need a preprocessing stage to either combine the raw data or combine the feature maps from different sensors.

- `Late fusion`
  In a late fusion scheme, independent classifiers are applied to individual sensor data. For example, in the thesis context, the Lidar-based detector and image-based detector work separately in the beginning. A final fusion stage is used to combine the detection results according to the low-level detectors and their certainties [27].

- `Deep fusion`
  With the development of the Neural Network (NN), researchers find it possible to fuse the information from individual sensors in the network. Wang et al. [28] proposes the Deeply-fused Net. It repeats a base module which combines shallow and deep subnetworks to construct a network with an exponentially increasing path. So, basically, Deep fusion is also a type of Early Fusion, but it has a special design of the network.

1-4 Thesis Outline

This thesis report presents the proposed approach and the results from the thesis work. A 3D road user detection system using the sensor data from the camera and Lidar is implemented. This thesis report is organized as follows: Chapter 2 gives an introduction to the state-of-the-art researches working on object detection, either using camera or Lidar. Chapter 3 explains the designed detection system in detail. Chapter 4 presents the experiments designed for the system validation and the parameter tuning together with the results of the experiments. Chapter 5 is the conclusion of the thesis work and some recommendations for the future research.

Chapter 2

Related Works

Object detection has arisen a lot of research interest over the years. As one of the main task of the autonomous vehicle perception system, the importance of 3D object detection is obvious since the later vehicle control algorithms need the accurate position of the obstacles to make a better evasive action. According to Section 1-2, Lidar and camera are two of the most widely used sensors in the automated vehicle perception systems. In this chapter, related works among three different research topics are introduced: the image-based object detection, the Lidar-based object detection and the multimodal object detection.

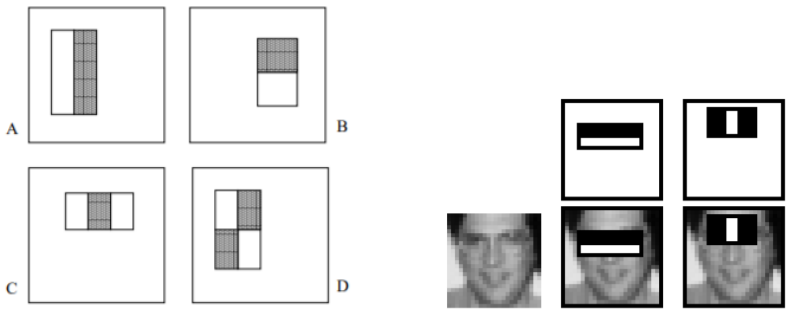
2-1 Image-based Object Detection

Due to the wide usage of Computer Vision (CV) in all aspects of modern life like medical, industry, public security, etc., the parsing algorithms of camera images have been thoroughly researched. The image-based object detection is also a hot research topic in the field of CV.

2-1-1 Classical detection system

About ten years ago, researchers were still focusing on creating hand-crafted features to represent objects in the image. In 2003, Viola and Jones detector (VJ detector) [29] was applied to the pedestrian detection. The VJ detector was originally designed for the face detection. It extracts the Haar features from the image patch. The patches shown in Figure 2-1a are overlayed on the face area in Figure 2-1b. The Haar feature is calculated by subtracting the sum of the pixel values within the white rectangles from the sum of the pixel values in the black rectangles.

In 2005, Dalal and Triggs created a new feature representation named Histogram of Oriented Gradients (HOG) for the pedestrians detection as illustrated in Appendix B-1-1, which mainly cues on silhouette contours [16]. A few years later, HOG was used as the building block for the Deformable Part Model (DPM) [18] which gives the state-of-the-art pedestrian features



(a) Rectangle features [30] **(b)** Example of Feature Value [30]

Figure 2-1: Feature of VJ detector

before the popularity of the Convolutional Neural Networks (CNN). After the DPM feature achieving success, a lot of detectors apply DPM in the feature extraction.

Felzenswalb et al. [18] used a latent SVM classifier to do the classification after extracting the DPM features. Park et al. [31] treat objects of different size differently because finite sensor resolution poses an undeniable limit to the scale invariance. Recognizing an object occupying 3 pixels is much harder than the one occupying 300 pixels. DPM is applied for the feature extraction for the large-scale objects and a rigid template is applied for scoring the small objects. Yan et al. [32] also applies a similar approach which treats the different size objects differently.

2-1-2 Region-based Convolutional Neural Networks

As one of the pioneering researches applying CNN to the object detection, Girshick et al. [33] made a great progress in the image-based object detection research. The authors created a popular detection pipeline named Region-based Convolutional Neural Networks (R-CNN), which first proposes Region of Interests (ROI) and then classifies the ROIs using the CNN. Researchers found that CNN is able to extract more distinctive features than the former hand-crafted methods. Due to its good performance in the image parsing, CNN also becomes the most successful Neural Network (NN) architecture so far.

The first step is to get the ROIs. Girshick et al. [33] propose ROIs with a selective search method [4] which first applies a segmentation algorithm named hierarchical grouping [34] to the image as shown in Figure 2-2. Different proposals of different size come from the segmentation blocks. Then, each ROI is fed into the CNN to be classified. This algorithm is super slow because it performs a CNN forward pass for each object proposal, without sharing computation [22]. Girshick [22] makes some improvements based on his former work [33]. He feeds the whole image and multiple ROIs into a fully convolutional network. Each ROI is pooled into a fixed-size feature map and then mapped to a feature vector by Fully Connected Layer (FC) [22]. This algorithm is 18.3 times faster than [33] when training and 213 times faster than [33] when testing.

Ren et al. [5] propose the Faster R-CNN which makes a big improvement based on the former works. They introduce a Region Proposal Network (RPN) for ROI proposals. The structure of Faster R-CNN detection algorithm is shown in Figure 2-3. The whole image is first fed into

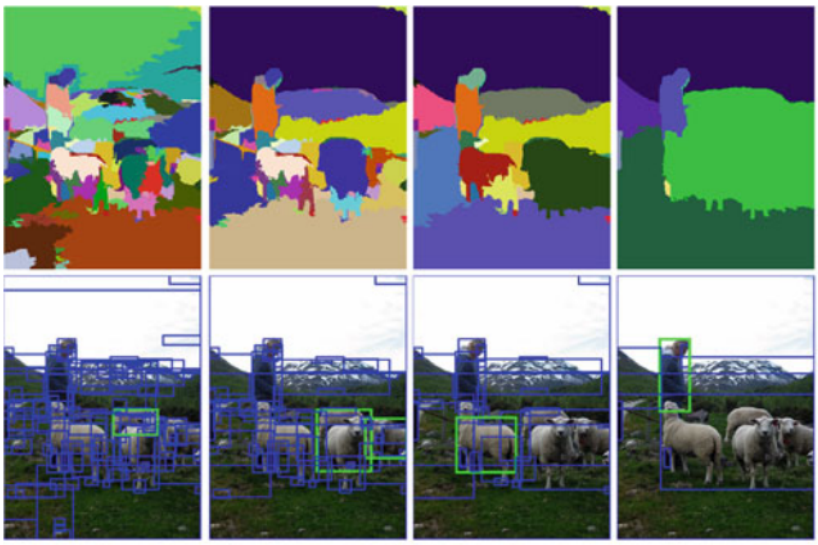


Figure 2-2: Selective search [4]

a few CNN layers to extract a feature map. The RPN generates high-quality region proposals, which are used by Fast R-CNN for detection [5]. The RPN is merged with Fast RCNN [22] by sharing the feature map. The final detection system has a frame rate of 5 Frames Per Second (FPS) and achieves state-of-the-art object detection accuracy on PASCAL VOC [35] and MS COCO [36] datasets.

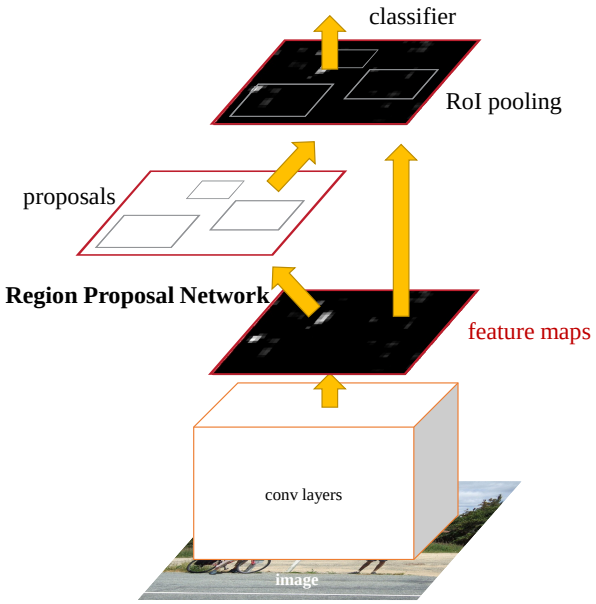
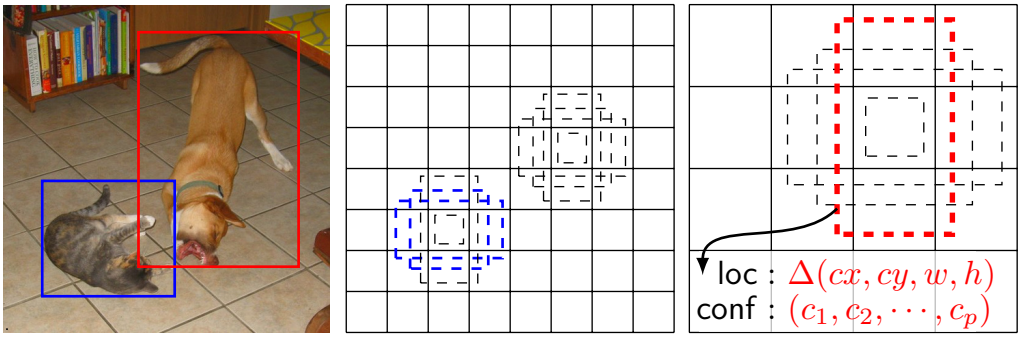


Figure 2-3: Faster R-CNN [5]

2-1-3 Detection without region proposals

Instead of detecting objects in two steps like R-CNN methods, some researchers are seeking algorithms that do the detection in a single feed-forward pass, like Single-Shot-Detector (SSD) [6], You Only Look Once (YOLO) [37], etc. Rather than doing independent processing for region proposals, the bounding box regression is treated as a subtask of the network. Finally, the network produces a collection of bounding boxes and scores for the presence of object class instances in those boxes [6].

In SSD, the image is fed to the base network with a standard architecture used for high-quality image classification [6]. The convolutional feature layers are added to the base network later. These feature maps are divided into some coarse grids as shown in Figure 2-4. Here the authors chose two different feature map splits: 8-by-8 and 4-by-4 for the objects in different scales. For each cell of the feature map, a set of base bounding boxes of different aspect ratios are evaluated (as shown in dash lines). For each box, they predict both the shape offsets and the confidence score for all object categories in the CNN.



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Figure 2-4: SSD framework [6]

Detection algorithms without region proposals are simpler in the structure and faster while inference comparing to the R-CNN methods. However, the R-CNN tends to provide more accurate results since it deals with the bounding box proposal separately which makes the predicted bounding box more accurate.

2-2 Lidar-based Object Detection

Lidar-based object detection systems use the point cloud as the input. Dealing with the point cloud is much more difficult than the image since it is unordered, scattered data structure. As introduced in Section 1-2, the point cloud has multiple commonly used representations. The point cloud detection algorithms heavily depend on the encoding method of this 3D data. Different encoding methods and different parsing algorithms lead to different solutions for the Lidar-based detection. There are three most commonly representation methods among the existing works. The first representation, which is also the most widely used method, is to *project* the point cloud onto the image plane or multiple views like front views, bird's eye views, etc. Then the point cloud is encoded as one image or a few images that can be fed into

the image parsing networks. The second representation is to voxelize the point cloud into 3D grid cells, which is also called *Volumetric Voxelization*. This converts the point cloud into an organized data structure. The third representation is to keep the point cloud as its origin and parse the point cloud directly. In this section, the detection algorithms applying these three different representation methods are introduced.

2-2-1 2D projection

Generally, there are three different kinds of point cloud projection methods. The first one is to project the point cloud onto the image plane and to get a depth map, which is also called the RGB-D projection since projection in this way gives some pixels in the RGB image a depth information. Papers such as [38][39] are applying this kind of projection. The image-based detectors can also be applied here to parse the extra depth map.

The second commonly seen method is to project the point cloud onto a spherical image, which will also be explained in Section 3-2-1. Each pixel of such a spherical image represents the distance from the scanned point to the sensor. This projection method is especially suitable for those rotating Lidar like Velodyne HDL64E. Since the Lidar is a rotating sensor, a spherical image makes use of most of the scanned points and keeps as much information from the point cloud as possible. Li et al. [25] apply this projection method and then use a fully convolutional network to parse the spherical image for the object detection.

The third projection method is to project the point cloud to the bird's eye view. Papers like [3][40] also combine the information from the point cloud bird's eye view map and all achieve great performance on 3D object detection.

2-2-2 Volumetric voxelization

To keep the data in 3D, some researchers choose to voxelize the point cloud, which means that the 3D space of the point clouds is divided into small grid cells of a certain size.

Song et al. [7] divide the 3D point cloud into cubic cells of size 0.1 meter and a feature vector is extracted from the points inside each cell. The features used in this paper are hand-crafted features like the point density feature, 3D shape feature, 3D normal feature and Truncated Signed Distance Function (TSDF) feature [41], which are shown in Figure 2-5.

TSDF is a measure of the self-occlusion. For each cell, it is further divided into $6 \times 6 \times 6$ voxels. TSDF value of each voxel is defined as the signed distance between the voxel center and the nearest object point on the line of sight from the camera [7]. The distance is clipped to be between -1 and 1 and the sign here indicates whether the cell is in front of or behind the surface [7]. After extracting the features, a linear classifier is applied to classify the feature vectors.

Wang et al. [42] apply a similar pipeline to [7]. They also divide the point cloud into 3D cells and apply a linear classifier on the extracted feature vector. However, they use a different feature representation and a different computational tractability. For the feature representation of each cell, they compute three shape factors, the mean and variance of the reflectance values of points, and a binary occupancy feature that is 1 for an occupied cell and 0 for an empty cell [42]. Engelcke et al. [43] improve Wang's work by substituting the

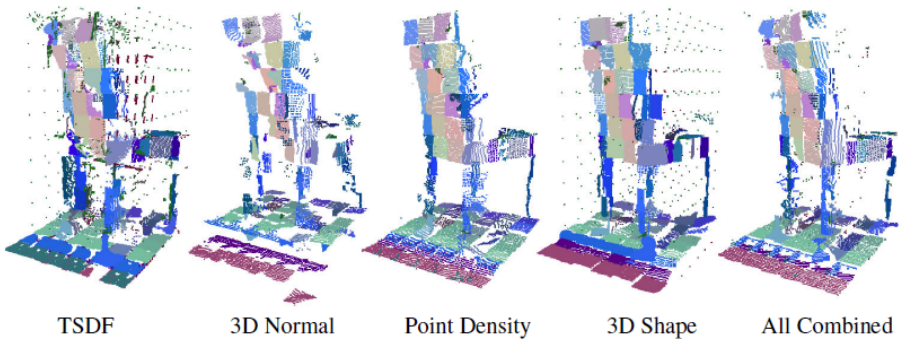


Figure 2-5: Visualization of 3D Features [7]

Support Vector Machine (SVM) classifier with a 3D CNN since a linear classifier in the case of the sliding window is equivalent to convolution [42]. For the computational tractability, they prove that the sparse convolution in the 3D point clouds is mathematically equivalent to the process of voting [42].

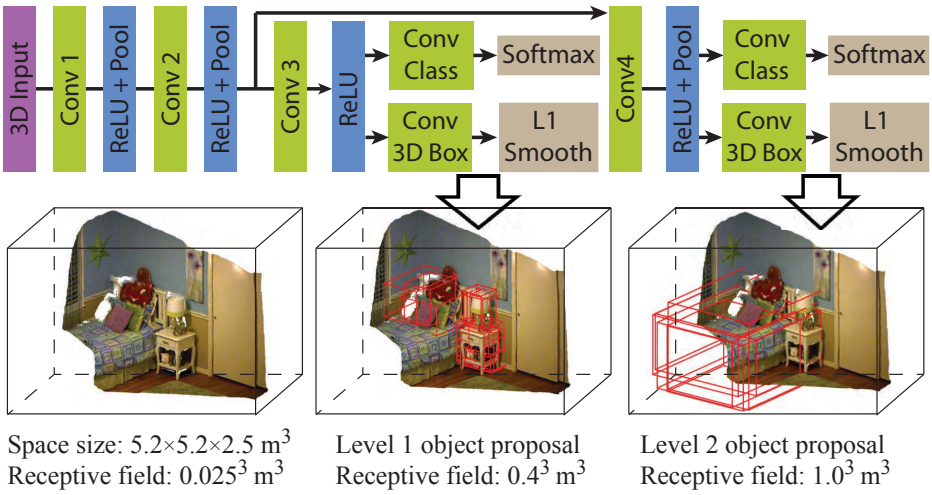


Figure 2-6: Multi-scale 3D amodal RPN of [8]

Song et al. [8] also make some improvements to their former work [7]. The point cloud is first voxelized and each voxel is encoded with the TSDF feature. Inspired by the RPN created by Faster R-CNN [5], this paper creates the first 3D RPN. RPN, which tells the machine where to focus on, is rather important since the computation in 3D is much more time-consuming than that in the 2D situation. Since different objects have different 3D size, they used a multi-scale 3D RPN which is shown in Figure 2-6. After the ROI have been proposed, a 3D Non-maximum suppression (NMS) is performed before each proposed region is fed into the object recognition network. The structure of the object recognition network is shown in Figure 2-7. The object recognition network shows that the RGB image feature vector and the point cloud feature vector are concatenated into a single vector that is further fed into a fully connected layer, which provides a simple fusion method for deep network features.

Zhou et al. [9] use a different method to avoid making hand-crafted features for each voxel.

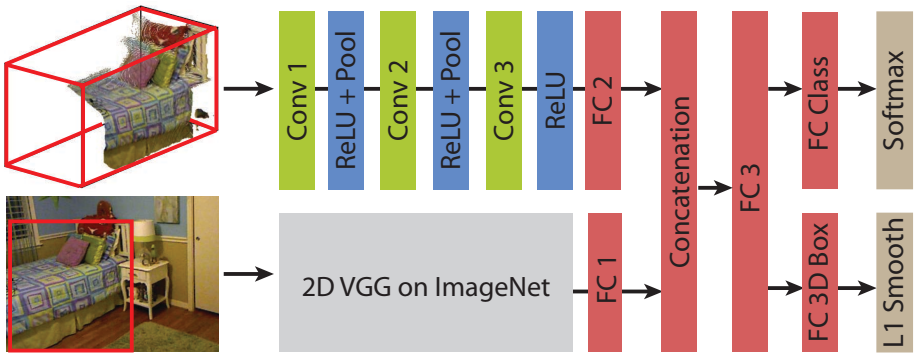


Figure 2-7: Object Recognition Network of [8]

They introduce a network named Voxelnet to simultaneously extract features and predict accurate 3D bounding boxes from the point cloud. The architecture of the network is shown in Figure 2-8. The points in each voxel are fed into multiple Voxel Feature Encoding (VFE) layers to extract local features. Then a 3D convolution layer further aggregates all local features. Finally, an RPN is applied to estimate the bounding box of the object. This paper gives the best Lidar 3D detection results in the KITTI benchmark so far. The VFE layer is similar to the architecture of Pointnet [2], which is one of the pioneer network for the point cloud parsing.

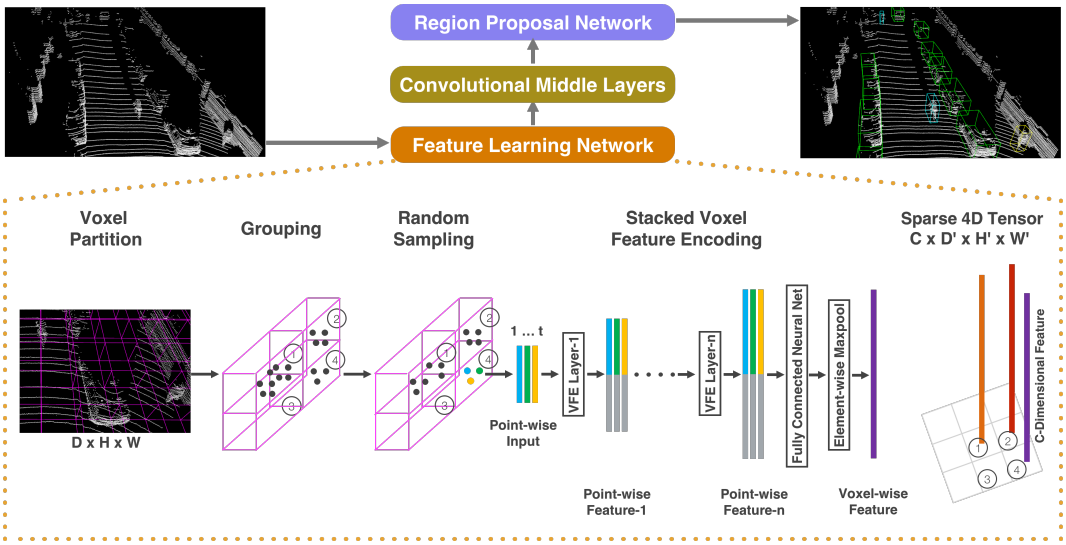


Figure 2-8: Architecture of the Voxelnet [9]

2-2-3 Original point cloud

The point cloud is probably the closest 3D representation to the raw sensor data. This avoids any kind of the man-made bias for the feature representation and makes the feature more precise.

Pointnet [2] is one of the pioneer work in feeding the original point cloud into the networks.

Each point is represented by its coordinates and sometimes with extra features such as colour, normal, etc. The network will have an $n \times 3$ array as the input if only the coordinates (x, y, z) are used where n corresponds to the number of points. The architecture of the network is shown in Figure 2-9. There are three special designs in the network. First, the network needs to be invariant to the input permutation since the point cloud fed to the network is unordered. A max pooling layer is applied to satisfy this demand, which is a simple symmetric function to aggregate the information from each point [2]. Second, the structure combines the local and global features in the segmentation network. Third, they apply an affine transformation matrix to the input coordinates by a mini-network (T-net in Figure 2-9). This mini-network aligns all input set to a canonical space to make the learned feature by the point set is invariant to certain geometric transformations [2].

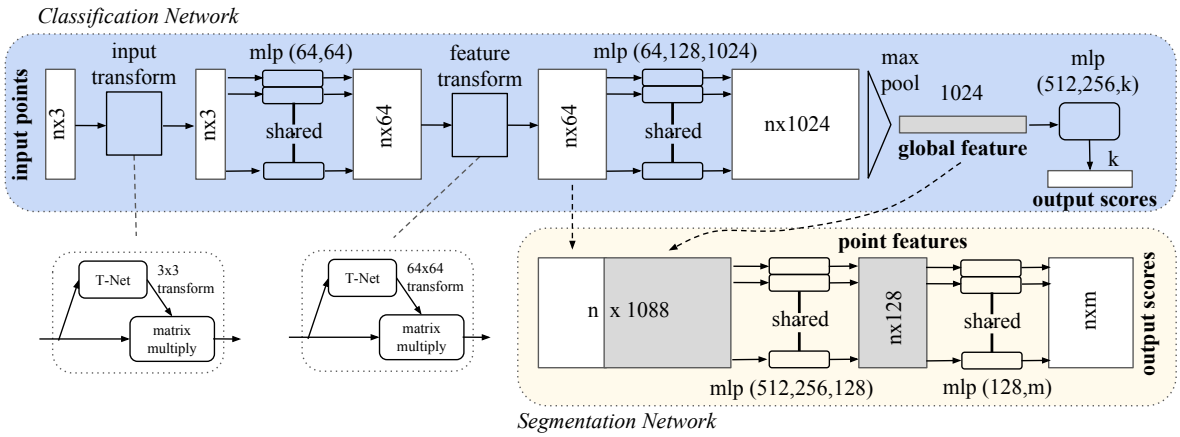


Figure 2-9: Architecture of the Pointnet [2]

Pointnet is originally designed for the indoor scene point cloud classification and segmentation tasks. Researchers are trying to apply Pointnet to the road scene. The most successful works are Voxelnet [9] as introduced in Section 2-2-2 and Frustum PointNets [10].

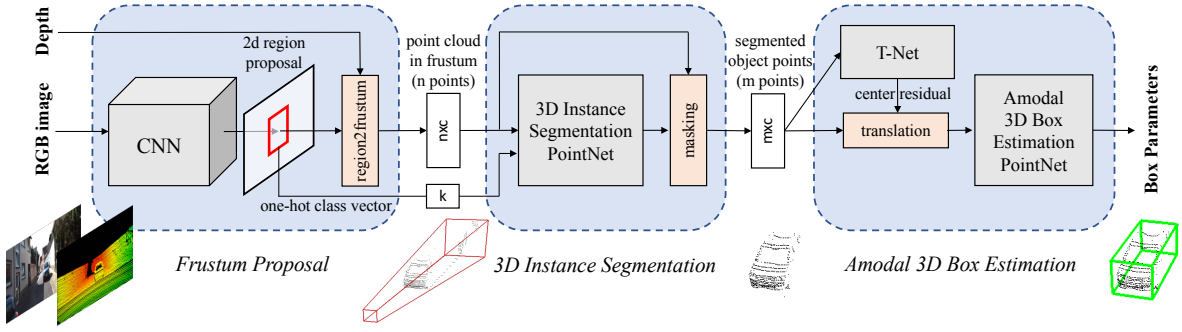


Figure 2-10: Architecture of the Frustum PointNets [10]

Frustum PointNets [10] is among the top ranking 3D detection algorithms in KITTI 3D object detection benchmark (accessed August 3, 2018). This paper uses image-based detection results for the region proposals in the point clouds. Pointnet [2] is applied in this work to parse the ROIs in the point cloud. Compared to Pointnet, the main contribution of this paper is to efficiently propose possible 3D locations of the object. The authors take advantage of

the image-based detection methods and propose 3D frustum of the object by extruding 2D bounding boxes from image-based detectors [10]. Basically, there are three modules in the system pipeline which is shown in Figure 2-10. The first module is *Frustum Proposal*. It adopts a Feature Pyramid Networks (FPN) [44] based model as the image-based detector to propose 2D ROIs. The object class also comes from the image-based detection results. According to the camera projection matrix, the 2D bounding box can be lifted to a frustum that defines a 3D search space for the object. The second module is *3D Instance Segmentation*. They did not apply a 3D object location regression because there might be occluding objects and background in the frustum as well. A Pointnet-based segmentation network is applied here. The third module is the *Amodal 3D Box Estimation* that provides the 3D bounding box of the object. A T-Net [10] is first applied to estimate the object center. Then the author transforms the coordinate such that the predicted center becomes the origin [10]. The output will be an amodal 3D bounding box of the object. Though the detection results are satisfying, there is a shortage of this method. The model is not robust enough since it relies on the image-based detector for the ROI proposal. As introduced in Section 1-2, the camera suffers from the weather conditions. The system is still suffering from illumination changes.

2-3 Multimodal Object Detection

One of the main concern in this thesis is about the Lidar and camera sensor fusion. In this section, the multimodal object detection is also referring to object detection algorithms combining Lidar and camera. As introduced in Section 1-3, there are three main fusion methods: early fusion, deep fusion and late fusion.

2-3-1 Early fusion systems

An early fusion system fuses the sensor data in the feature level and applies a unique classifier that is able to retrieve information from several sensors [27]. A preprocessing stage is needed to extract feature vectors from different types of sensor data. Here, three different types of early fusion methods are introduced and the biggest difference between these methods is the design of the data preprocessing stage.

Premebida et al. [38] project the point cloud onto the image plane and generate a depth map aligned with the RGB image. Since the point cloud is a sparse data structure, an upsampling algorithm is applied to the sparse depth map to create dense alignment with the RGB image. Then the upsampled depth map is treated as extra image channel. This paper extracts the HOG feature from the depth map and the RGB image. Finally, an SVM classifier is trained from both HOG features of RGB image and depth map.

Gonzalez et al. [39] make an improvement on the Premebida's work. They project the point cloud in the same way with Premebida's work. However, different from Premebida's work, they also extract Local Binary Pattern (LBP) features except for HOG features both from the depth map and RGB image. This multi-cue features may provide a more discriminative feature than a single one. On the other hand, this paper splits the training data according to the orientation of the object to train an orient-discriminative classifier.

Another commonly used early fusion scheme is to use the point cloud to propose 3D ROIs and project the ROIs to the RGB image plane for the image-based classification. Jafari et al. [45] do the early fusion in this way. The first step is to segment the points into three categories: *objects*, *fixed structures* and *ground plane*. The fixed structures are removed before further processing. Then a Random Sample Consensus (RANSAC) plane-fitting algorithm is applied here to find the ground plane. After getting the ground plane, all the points that have been segmented as *objects* are projected to the ground plane forming point blobs. Then the 3D ROIs are extracted from the point blobs on the ground [45]. Then the 3D ROIs are back-projected onto the RGB image plane for the 2D proposals, which will be fed into the image-based detector. However, the fusion scheme is only applied to the close range objects detection. For the far range objects, only the image is used for detection because of the limitation of the distance of the point clouds.

2-3-2 Late fusion systems

For a late fusion system, one or several low-level classifiers are applied to different sensor data independently. In the final stage, all the low-level classification results are combined using different fusion methods such as Bayesian classifier, Global Nearest Neighbor, fuzzy logical model, Dempster-Shafer theory, etc. Since the image-based detection and the Lidar-based detection are dealt with separately, basically, all the image-based detectors and Lidar-based detectors can be applied here for the low-level detection. This can make the most of the available successful detectors.

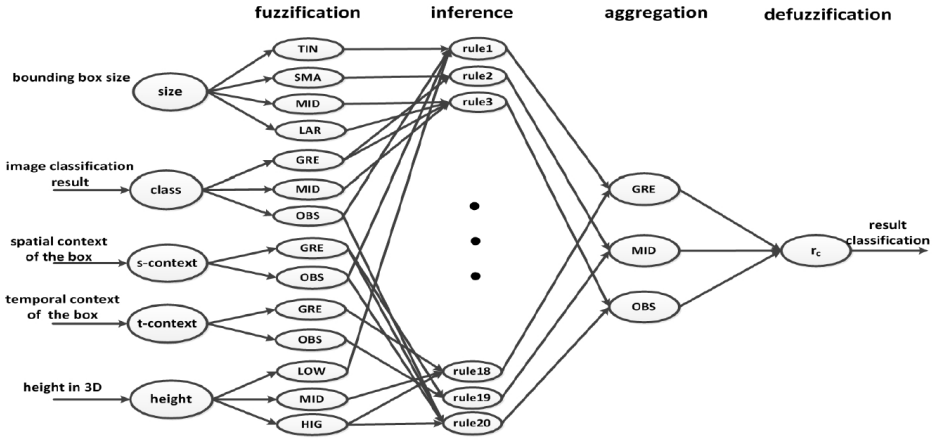


Figure 2-11: Fuzzy Logic Model [11]

Zhao et al. [11] apply a late fusion scheme in their work. For the Lidar-based detection, three steps are applied here for the obstacle detection. The first step is to voxelize the point cloud. Then, the ground points are separated according to a RANSAC plane-fitting algorithm [46]. Finally, the above ground points are clustered according to 3D adjacency. All the clusters are possible obstacles. For the image-based detection, both bottom-up and top-down classification is applied. The image is divided into small batches and each batch is classified according to its feature vectors in the bottom-up classification. Then a top-down contextual analysis process is applied to correct the classification mistakes. In the final stage,

a fuzzy logic inference fusion method is applied as shown in Figure 2-11. There are 5 inputs fed into the fuzzy logic model: the size of the obstacle, the image classification result, the spatial context, the temporal context and the absolute height of the candidate obstacles. The output is the final detection results for the obstacles. All the five input are inferred from the detection results of the camera and Lidar. There are 20 rules applied in this model that are based on the human experience and knowledge of the road scene. No exact class of the obstacle is further determined. A problem with the fuzzy logic model is that there might be mistakes in human experience which might result in mistakes in the rule definition.

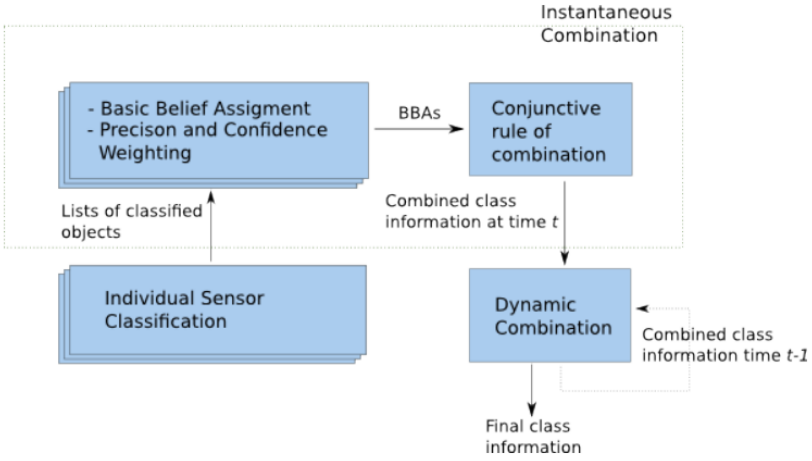


Figure 2-12: Fusion Architecture of [12]

Vu et al. [12] apply a different way of the late fusion, which is called Dempster-Shafer theory. The authors use a 2D Lidar here. The point cloud from the Lidar is on the same horizontal plane. The detection methods of the objects might be quite different from the 3D one. For the image-based detection, a Dalal and Triggs detector [16] is applied here. Vu makes an improvement to this by applying a Sparse HOG feature method which selects a few discriminative areas for the feature extraction. What's more, Vu uses the Lidar and Radar data for the ROI proposals in images for a faster detection. In the final stage, a late fusion approach based on Dempster-Shafer theory [47] is implemented to combine instantaneous information from current environment state gathered from multiple sensors. The approach applies a rule of combination taking into account classification evidence from different sources of evidence (Lidar/camera) [48]. After the fusion of current information, the proposed approach fuses it with the previous combination results. The basic fusion architecture is shown in Figure 2-12. There are three inputs here for the proposed approach: several lists of detected objects and their class information, the reliability of the sources of evidence and the precision detection for certain type of classes [48]. The final output of the proposed method comprises a list of objects with combined class information [48].

2-3-3 Deep fusion systems

The former two types of fusion are the conventional ways of sensor fusion. Wang et al. [28] name the conventional fusion methods as the shallow fusion and they create a new type of

fusion: Deep Fusion. This method relies on the development of the neural networks and the fusion is implemented in the network. NN has proved its capability of learning the features from a large amount of data. In the research of multimodal sensor fusion, researchers are also seeking methods to make the network learn to fuse the information from two different sensor data.

Song's Deep sliding shapes [8] fuses the two features in the network. The architecture of the network is shown in Figure 2-7. It is shown that the voxelized point cloud and the RGB image are first fed into different networks to extract feature vectors. Then the two feature vectors are concatenated in a layer and further fed into a fully connected layer for the classification and the bounding box regression.

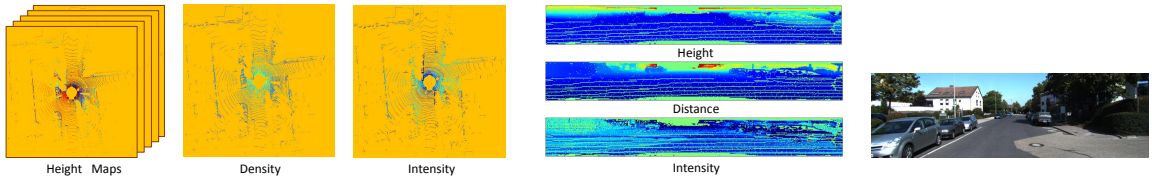


Figure 2-13: Multi-view representation of MV3D [3]

Different from Song's [8], MV3D [3] projects the point cloud into multiple views. The first view is the bird's eye view representation. The bird's eye view is composed of the height map, intensity map and density map according to different clues. It is voxelized into 2D grids of size 0.1m. The height map comes from the height of the points in each cell. The intensity map uses the reflectance value of the point as the clue. The density map is the number of points in each cell. Another view is the front view. The front view is also a spherical projection of the point cloud as in [49]. The multi-view representation is shown in Figure 2-13. The 3D object proposals are generated from the bird's eye view. The bird's eye view is better for object proposals than the front view. First, the objects need to be detected preserve the size in bird's eye view. Second, there will be no occlusion problems in bird's eye view. Third, the vertical locations of the road users, such as *Pedestrian*, *Cyclist* and *Car*, have less variance since they all lie on road.

For the sensor data fusion, MV3D [3] also uses the idea of combining features of different sensor data in the network. The network architecture of the MV3D is shown in Figure 2-14. As illustrated before, the 3D object proposals (represented by 3D bounding boxes) are extracted from the bird's eye view. MV3D first designs a set of 3D prior boxes according to the ground-truth object sizes in the training set [3]. After getting the corresponding bird's eye view of the 3D prior boxes, the bird's eye view of the point cloud is searched for 2D boxes similar to prior boxes. There are seven parameters for each 3D bounding box including the center point coordinates $(x,\ y,\ z)$, the size of the 3D bounding box (l, w, h) and the orientation of the bounding box (γ). The proposal from the bird's eye view provides five of the parameters $(x,\ y,\ l,\ w,\ \gamma)$. The height of the box and z come from the object height and camera height separately. After getting the 3D object proposals, the 3D bounding boxes are projected onto the feature maps of each view. Then features from all views are fused with the Region-based Fusion Network. The Region-based Fusion Network of this paper is more complicated than Song's [8], which fuses multi-view features hierarchically. The final step is the classifier and the oriented 3D box regression.

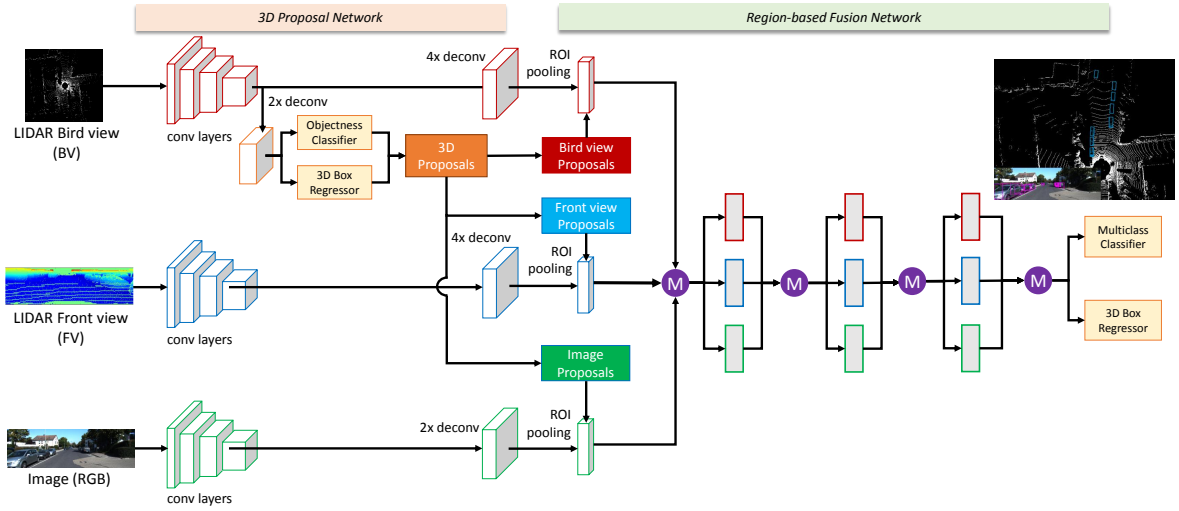


Figure 2-14: Architecture of the MV3D network [3]

2-3-4 Comparisons of different fusion structures

Early fusion Basically, the deep fusion is also one kind of early fusion, which combines different information clues in the network. The early fusion method combines different sensor data in an early stage. So, the feature map combines different clues from different sensors, which makes the feature map more discriminative. What's more, early fusion makes the system cleaner in the system structure.

However, one shortage of the early fusion systems is that the system lacks redundancy. If a single sensor is not working properly, the performance of the whole system will be largely affected. For example, the camera might have a poor performance in detecting objects at night.

Late fusion Since the detection is first carried out for every single sensor in the late fusion systems, it is possible to apply the existing successful detectors for a single sensor, such as the successful image-based detectors nowadays. It is also possible for a late fusion system to provide the preliminary results from every single sensor, which provides the system redundancy in case of a single sensor failure.

A shortcoming of the late fusion systems is that the fusion stage should be carefully designed to provide more reliable results than the single sensor detector.

2-4 Theories

2-4-1 Object classification

Object classification is one of the hottest research fields in Machine Learning (ML). In the context of this thesis, object classification refers to judge which category a certain part of an image or the point cloud belongs to. The categories will usually be defined according to the

context. In this thesis work, the road users are mainly concerned such as *Cars*, *Pedestrians*, *Cyclists*, etc.

The classification is executed by a classifier. Instead of feeding the raw data (the image and the point cloud) into the classifier, the researchers usually will feed the feature vector extracted from the image or the point cloud to represent the raw data. The ML process here is to find the mapping function f between the input feature vector ($\mathbf{x}$) and the output class decision (y) as illustrated in Eq. (2-1).

$$y = f(\mathbf{x}) \tag{2-1}$$

The feature can be a hand-crafted one such as HOG, DPM, etc., extracted from the image. They come from the human knowledge about the raw data. More detailed information will be given in Appendix B-1. Since the popularity of the NN, researchers find that the network is able to learn a more distinctive feature than the hand-crafted one.

2-4-2 Neural Network

NN provides a new idea of doing the classification. A NN is basically composed of the artificial neurons as shown in Figure 2-15. Each input is multiplied by a weight factor w_i and added together with a bias term. The sum is fed into an activation function which is usually a non-linear function to get the output. Such an artificial neuron is also named as a perceptron. The mathematical explanation of the perceptron can be described in Eq. (2-2), where f is the activation function.

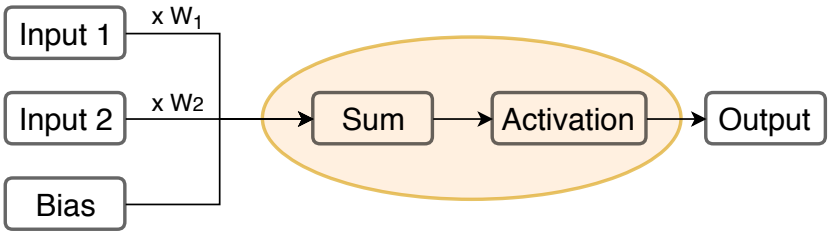


Figure 2-15: Artificial neuron

$$Output = f(\sum_{i=0}^{n} w_i x_i + b) \tag{2-2}$$

Connecting multiple perceptrons together makes a simple NN example as shown in Figure 2-16. The inputs are called input layer and the output is called output layer. The layers between the input layer and the output layer are called hidden layers. In the thesis context, the input can be the image or the point cloud and the output is the classification result. A loss will be designed to measure the difference between the output and the true value. The training of the NN is to optimize the weight and bias values to minimize the designed loss. Then the

gradient descent can be backpropagated through multiple layers, from the output layer back to the input layer.

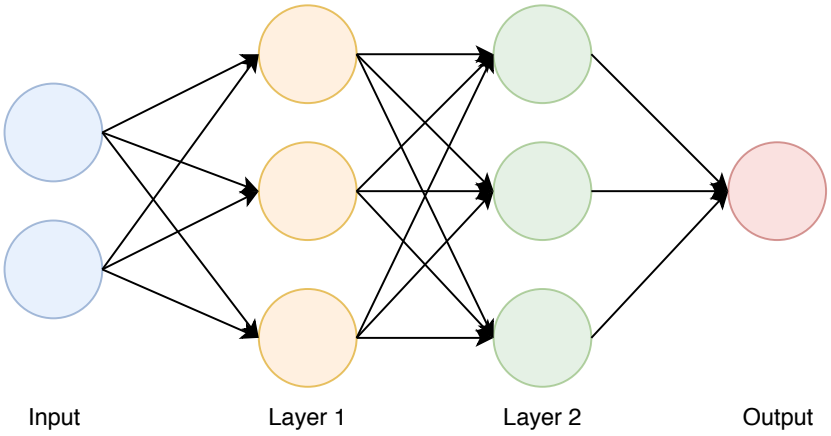


Figure 2-16: Multiple Perceptron Network

2-4-3 Convolutional Neural Networks

For the image parsing, the most successful network is CNN. CNN is mostly composed of convolutional layers and pooling layers.

A convolutional layer is basically a perceptron with special inputs and weights. A grayscale image is a 2D matrix and the colour image is a 3D matrix with an extra colour dimension. The weight used in the convolutional layer is called kernels, which has the number of same dimensions as the image. A simple 2D convolutional layer example is shown in Figure 2-17. The kernel will slide across the input image in a certain stride through both columns and rows. The stride in the given example is 1 both in columns and rows. For each position the kernel passing by, the result of the convolution is the sum of the element-wise multiplication of the kernel and its receptive field (the red patch in the example figure). There is a problem here that the feature map generated in this way is smaller than the input image. If the feature map has to keep the same size with the input image, a zero-padding step can be applied to the input image as shown in Figure 2-18.

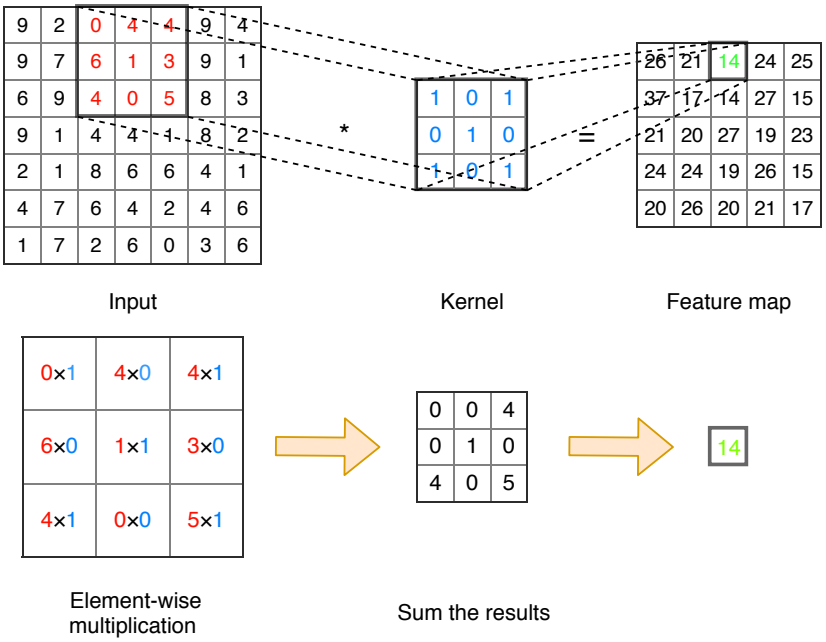


Figure 2-17: Convolution example with kernel size 3×3 and stride 1

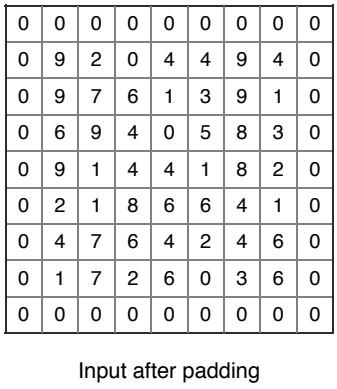


Figure 2-18: Zero padding

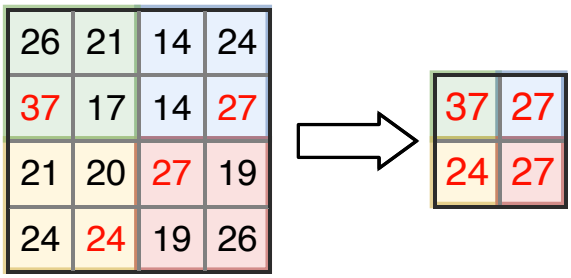


Figure 2-19: 2 by 2 max pooling

The second most commonly seen layer type is the pooling layer. Pooling layers will subsample the input image, which reduces the memory use and the number of parameters in the network. A simple example of the 2 by 2 max pooling is shown in Figure 2-19. Except for the max pooling, another commonly seen pooling method is the average pooling which calculates the average.

Instead of dealing with the images, CNN is also capable of dealing with all data format that can be encoded as a multi-dimensional tensor. As introduced in Section 2-2-3, Pointnet [2] encodes the point cloud into a $N \times 3$ matrix and use a CNN to parse the point cloud as well.

2-4-4 Recurrent Neural Network

Different from the normal neuron in the feed forward network as described in Section 2-4-2, the recurrent neuron sends the output back to itself as shown in Figure 2-20.

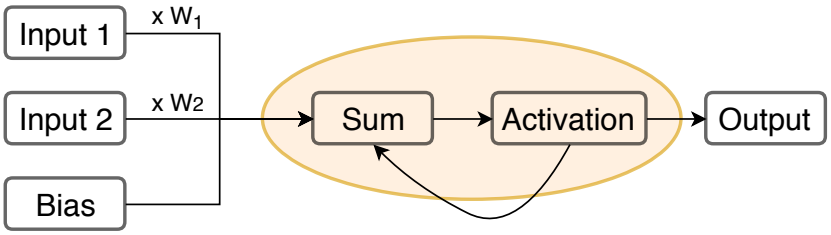


Figure 2-20: Recurrent neuron

Recurrent Neural Network (RNN) is usually used to deal with the sequential data. A simple RNN layers with the data flow overtime is shown in Figure 2-21. RNN is also flexible in their inputs and outputs, for both sequences and single vector values.

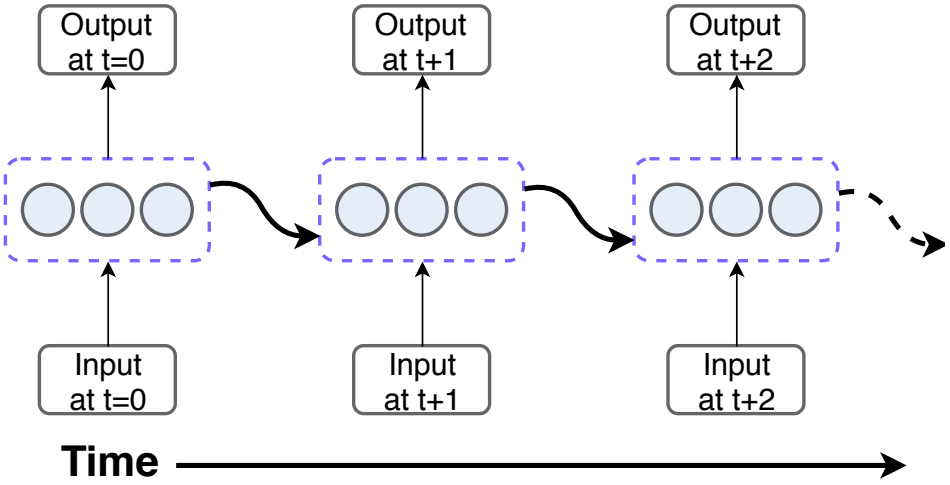


Figure 2-21: Recurrent neuron layer and the data flow over time

2-4-5 Loss

Except for the network architecture, the loss function is another key component of NN. The loss function measures how far off the predicted value is from the true value. Several types of the loss function are introduced in this section.

Classification loss

The L1 loss is the absolute difference between the predicted value and the truth value as following:

$$l1\ loss = \sum_{i=0}^{n} |y_i - f(x_i)| \tag{2-3}$$

where y_i is the truth value and $f(x_i)$ is the predicted value.

The L2 loss is the square difference between the predicted value and the truth value as following:

$$l2\ loss = \sum_{i=0}^{n} (y_i - f(x_i))^2 \tag{2-4}$$

where y_i is the truth value and $f(x_i)$ is the predicted value.

Hinge loss is one of the most commonly used loss function in classification tasks. The equation describing the hinge loss is:

$$hinge\ loss = \sum_{j \neq c} (s_j - s_c + 1) \tag{2-5}$$

where s_c is the classification score of the correct class and the s_j is the classification score of the other classes. Hinge loss is specially designed for the SVM classifier.

Cross-entropy loss is another commonly used loss function in classification tasks. The input of the cross-entropy function should be the class probability vector that is composed of the probability value between 0 and 1 and the elements of the class probability vector sum up to 1. So it is usually used with the softmax activation function which is shown in Eq. (3-8). The formula of the cross-entropy loss is:

$$cross\text{-}entropy\ loss = -\sum_{c=1}^{N} y_c \log(p_c) \tag{2-6}$$

where N is the number of classes, y_c is the true probability value for class c and p_c is the predicted probability value for class c.

Regression loss

Huber loss is the loss commonly used for regression tasks. It is less sensitive to the outliers. The function of the Huber loss is shown in Eq. (2-7) where y is the true value , $f(x)$ is the predicted value and δ is the hyperparameter.

$$Huber\ loss = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| < \delta \\ \delta\,|y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \tag{2-7}$$

Chapter 3

Proposed Approach

As illustrated in Section 2-3-4, both the early fusion and the late fusion have their advantages and disadvantages. In the proposed approach, a late fusion system is built since it provides redundancy and robustness. Without any human intervention, the future autonomous vehicles have to deal with all possible situations that might happen while driving. The redundancy and robustness might be one of the top requirements for an autonomous vehicle because of the injury and fatality to human it might cause in case of the detection failure. It is quite necessary for an autonomous vehicle to operate normally under all kinds of circumstances. Considering that different sensors are good at operating in different circumstances, the late fusion might be a better choice for the future autonomous vehicle.

An overview of the system built in the thesis work is shown in the Figure 3-1. The image-based detector and the Lidar-based detector first work separately. The image-based detector supplies the detection results as the 2D bounding box in the image and the classification vector of each detected object. Elements of the classification vector is the probability of the detected object belonging to each class. The Lidar-based detector supplies the detection results as the 3D bounding box in the point cloud and the classification vector of each detected object. After getting the results from these two single sensor detectors, a final fusion stage will combine the two different results to a final result, which can be more convincing. Details of each module of the system will be introduced in this chapter.

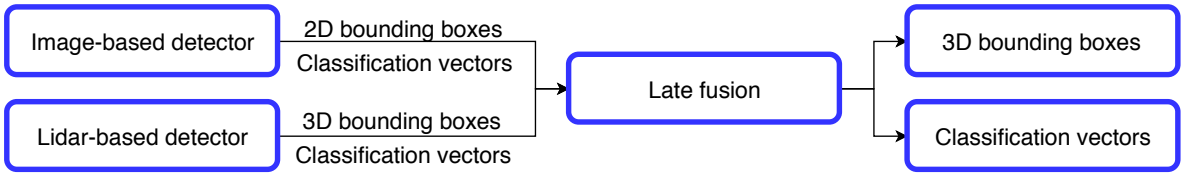


Figure 3-1: System overview

3-1 Image-based Object Detection

There are many state-of-the-art works performing really well on the image-based detection tasks as illustrated in Section 2-1. The output expected from an image-based detector is the object location (2D bounding box) in the RGB image and the classification vectors of the detected objects. So, basically, all state-of-the-art works can be used as the image-based detector in the system since they all can provide the expected results. After searching most of the existing image-based detectors, SqueezeDet is not only accurate but also small, fast and energy-saving [13], which makes it a real-time detection system that can reach 57.2 Frames Per Second (FPS) with 1242x375 image resolution. Finally, the SqueezeDet [13] is chosen as the image-based detector used in the thesis.

3-1-1 SqueezeDet network

The detection pipeline of SqueezeDet [13] is shown in Figure 3-2. The input image first goes through a convolutional neural network and becomes a low-resolution, high dimensional feature map. Then, the feature map is fed into a *ConvDet* which predicts tens-of-thousands of possible bounding boxes. The final step filters the detection results and only the top N detection results with the highest probabilities are kept. A final Non-maximum suppression (NMS) step is also performed to remove the duplicated detection results.

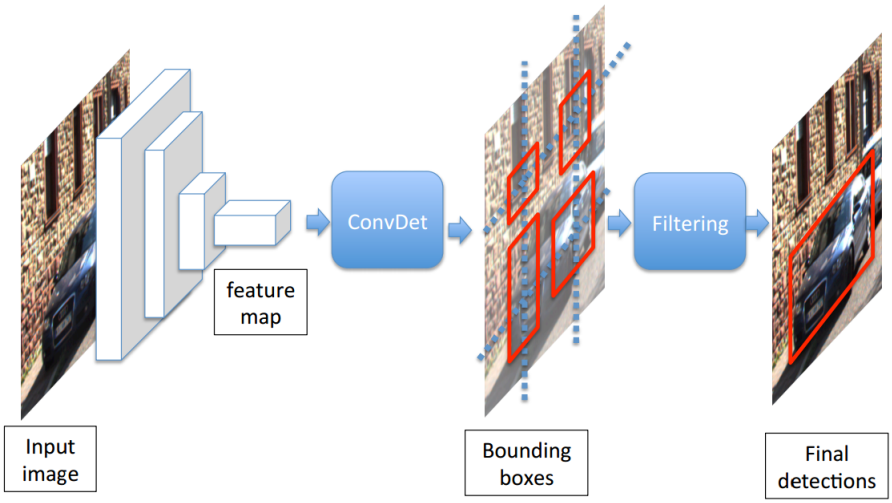


Figure 3-2: SqueezeDet detection pipeline [13]

The cornerstone of this algorithm is the *ConvDet* layer. *ConvDet* is basically a convolutional layer that predicts the bounding box coordinates and the class probabilities as shown in Figure 3-3. Similar to Single-Shot-Detector (SSD), SqueezeDet also divides the image into $H * W$ grids. In each grid, K anchor boxes are set as the reference box. The *ConvDet* works as a sliding window that goes through $H * W$ positions in the feature map and each position corresponds to the grid center of the original image. In each position, *ConvDet* computes $K * (4 + 1 + C)$ values as the prediction in the current grid. K is the number of the anchor boxes. 4 corresponds to the residual value of the box center (Δx and Δy) and the box size

(Δw and Δh). 1 corresponds to the predicted confidence score which will be described as in Eq. (3-1). C corresponds to the class probabilities of each class. So the output of the *ConvDet* has a size of $H * W * K * (4 + 1 + C)$.

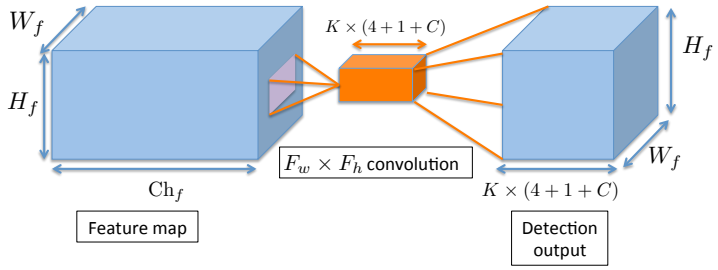


Figure 3-3: *ConvDet* layer [13]

3-1-2 SqueezeDet training loss

During training, a confidence score is defined similar to You Only Look Once (YOLO) [37] as following:

$$\text{confidence score} = Pr(\text{Object}) * IOU^{\text{pred}}_{\text{truth}} \tag{3-1}$$

where $Pr(\text{Object})$ is the classification probability of the object existing in the bounding box and Intersection Over Union (IOU) measures the overlap between the predicted bounding box and the ground-truth bounding box. The confidence score is a measure of the accuracy of the detection results and is also used in the training loss.

The designed loss for the SqueezeDet training is composed of three parts as follows:

$$\begin{aligned} L_{squeezedet\ loss} &= L_{box\_reg} + L_{conf\_reg} + L_{cls} \\ L_{box\_reg} &= \frac{\lambda_{bbox}}{N_{obj}} \sum_{i=1}^{W}\sum_{j=1}^{H}\sum_{k=1}^{K} I_{ijk}[(\delta x_{ijk} - \delta x^G_{ijk})^2 + (\delta y_{ijk} - \delta y^G_{ijk})^2 + (\delta w_{ijk} - \delta w^G_{ijk})^2 \\ &\quad +(\delta h_{ijk} - \delta h^G_{ijk})^2] \\ L_{conf\_reg} &= \sum_{i=1}^{W}\sum_{j=1}^{H}\sum_{k=1}^{K} \frac{\lambda^+_{conf}}{N_{obj}} I_{ijk}(\gamma_{ijk} - \gamma^G_{ijk})^2 + \frac{\lambda^-_{conf}}{WHK - N_{obj}} \bar{I}_{ijk}\gamma^2_{ijk} \\ L_{cls} &= \frac{1}{N_{obj}} \sum_{i=1}^{W}\sum_{j=1}^{H}\sum_{k=1}^{K}\sum_{c=1}^{C} I_{ijk} l^G_c \log(p_c) \end{aligned} \tag{3-2}$$

Bounding box regression loss The first part is the L2 loss (Section 2-4-5) for the bounding box regression. Each anchor box is compared to the ground-truth bounding box that has the biggest overlapped area with it. I_{ijk} equals to 1 if the k-th anchor at position-(i, j) has the largest overlap with a ground-truth box, and to 0 if no ground-truth is assigned to it [13]. This way, only the loss generated by the anchors having corresponding ground-truth bounding boxes are included [13].

Confidence regression loss The second part of the loss is the L2 loss (Section 2-4-5) for the confidence score regression. γ_{ijk} is the output from the *ConvDet* representing the confidence score and γ^G_{ijk} is the actual confidence score calculated by Eq. (3-1). $\bar{I}_{ijk}\gamma^2_{ijk}$ ($\bar{I}_{ijk} = 1 - I_{ijk}$) is a term used to penalty those anchors that do not have corresponding ground-truth bounding boxes. λ^+_{conf} and λ^-_{conf} are two weights used to balance the two loss components.

Classification loss The last part of the loss is the cross-entropy loss (Section 2-4-5) for the classification.

3-2 Lidar-based Object Detection Design

Inspired by the Region-based Convolutional Neural Networks (R-CNN) works like [33][22][5] detecting the objects in two steps: region proposal and classification, the Lidar-based detector proposed in this thesis work also takes similar two steps. The first step is to find out possible objects in the point cloud using a segmentation algorithm. All clusters from the segmentation might be a possible object in reality. The second step is to feed the clusters into a classification and bounding box regression network to determine which class each cluster belongs to and to estimate an amodal bounding box of the object. An overview of the Lidar-based detector is shown in Figure 3-4.

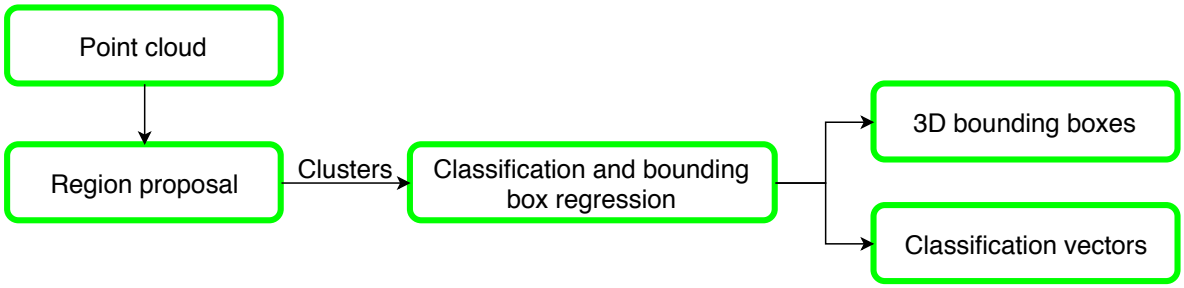


Figure 3-4: Lidar-based detector overview

Region proposal The are multiple existing point cloud segmentation algorithms. Generally, there are three different types of point cloud segmentation. The first group segments the point cloud in the 3D domain. Choe et al. (2012) [50] first remove the ground and then segment the rest point cloud with a nearest neighbour approach. Douillard et al. (2014) [51] extracted a hand-crafted 3D features for the segmentation. The second group projects the point cloud onto the ground plane and the segmentation is carried out in the image plane, such as [52][53]. The third group segments the point cloud in the range image such as [14][54][15]. Two segmentation methods are introduced here: **Depth clustering** [14] and **SqueezeSeg** [15].

3-2-1 Depth clustering

Depth clustering is an easy and fast segmentation algorithm which can produce good segmentation results [14]. After getting the point cloud from the Lidar, the point cloud is first

projected onto a spherical range image for faster computation. The number of rows of this range image is the number of laser beams in the vertical direction. For a Velodyne HDL-64E, there will be 64 rows in the range image. The number of columns is related to the range readings per 360°resolution of the laser. The value in each pixel of the range image represents the distance of the scanned point to the sensor. An example range image is shown in the top left of Figure 3-5. After such conversion, the sparse and unordered point cloud is transferred into a dense and organized image, which allows faster computation. Depth clustering is composed of two parts: Ground removal and Segmentation.

Ground removal

In order to make the clusters more likely to be the real road users in the thesis context, the ground points should be removed. Similar to [14], three assumptions are made here to estimate the ground points. First, the Lidar is roughly horizontally mounted to the vehicle. Second, the curvature of the ground is low [14]. Third, there are some pixels in the last row of the range image is reflected from the ground. In another word, the lowest laser ray can see the ground at some point.

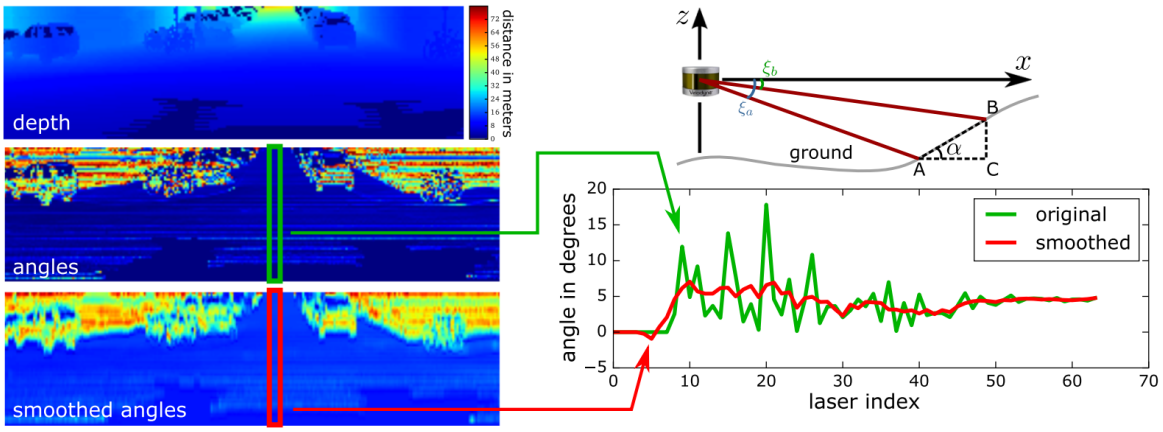


Figure 3-5: *Top left*: range image, *Middle left*: α angle image, *Bottom left*: smoothed α angle image, *Top right*: definition of angle α [14]

After making such assumption, the range image is converted into an α angle image by computing the α angle for every two neighbouring points in each column as shown in top right of Figure 3-5. The angle α represents the inclination of the line connecting two neighbouring points in the same column of the range image. $R_{r-1,c}$, $R_{r,c}$ denote the distances from point A and point B of row $r-1$ and r to the sensor, which can be read from the range image. The angle α can be computed from Eq. (3-3):

$$\begin{aligned} \alpha^r_{r-1,c} &= atan2(\|BC\|, \|AC\|) = atan2(\Delta z, \Delta x) \\ \Delta z &= |R_{r-1,c} sin\xi_a - R_{r,c} sin\xi_b| \\ \Delta x &= |R_{r-1,c} cos\xi_a - R_{r,c} cos\xi_b| \end{aligned} \tag{3-3}$$

where ξ_a and ξ_b are defined in top right of Figure 3-5.

Then, replacing the distance value in each pixel of the range image by $\alpha^r_{r-1,c}$ (c represents the column and r represents the row) makes an α angle image as shown in the middle left of Figure 3-5. Since Lidar will provide a substantial amount of outliers when scanning, a SavitskyGolay filter [55] is used here to smooth the α angle image. This filter performs least-squares optimization to fit a local polynomial for a given window size to the data [14].

After getting the smoothed α angle image, the ground points labelling is carried out. The bottom row of the α angle image is first labelled according to the α angle value. Those columns of the bottom row are labelled as the ground if the angle value is smaller than 45°. According to the third assumption made before, there must be some columns of the bottom row been labelled as the ground. Starting from these ground points on the bottom row, an N4 neighborhood search is executed to find out if those neighbouring points around the labelled ground points belong to the ground as well. The threshold used here is to judge if the angle difference $\Delta\alpha$ between the neighbouring points and the ground points is less than 5°. If $\Delta\alpha$ is less than 5°, the neighbouring point will also be labelled as the ground.

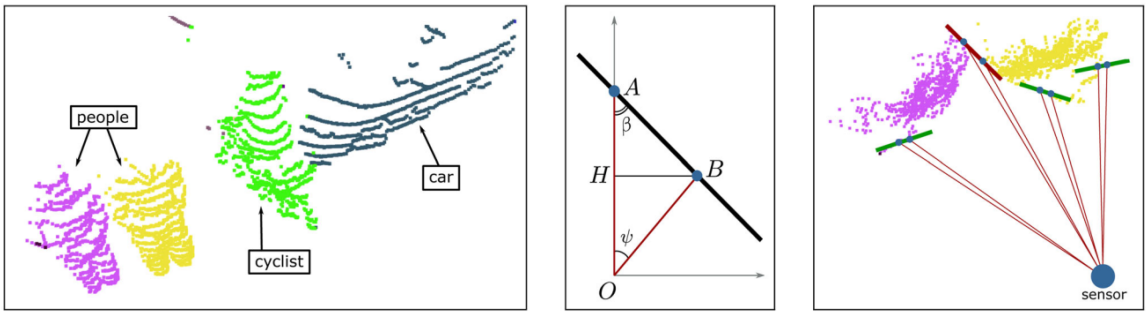


Figure 3-6: Angle-based segmentation [14]

Segmentation

After removing the ground points from the range image, the rest points will be segmented. The segmentation is also an angle-based method. Similar to the ground removal step, the segmentation algorithm is judging if the neighbouring points belong to the same object in an N4 neighborhood. Starting from the top left corner of the range image, the algorithm passes through all the pixels in the range image from top to bottom, left to right. For each point that does not have an assigned cluster, the algorithm will search its N4 neighborhood. For each neighbouring point (denoted as A) of the current point (denoted as B), angle β is calculated as shown in the middle of Figure 3-6, where O is the position of the sensor and OA/OB represent the laser ray from the sensor to A/B. The angle β can be calculated as follows:

$$\beta = atan2(\|BH\|, \|HA\|) = atan2(d_2 sin\psi, d_1 - d_2 cos\psi) \tag{3-4}$$

where ψ is related to the angular resolution either in the column direction or the row direction.

The intuition behind the angle β is that it will become rather small if the depth difference between neighbouring points is substantially large, in which case, it is highly possible that the

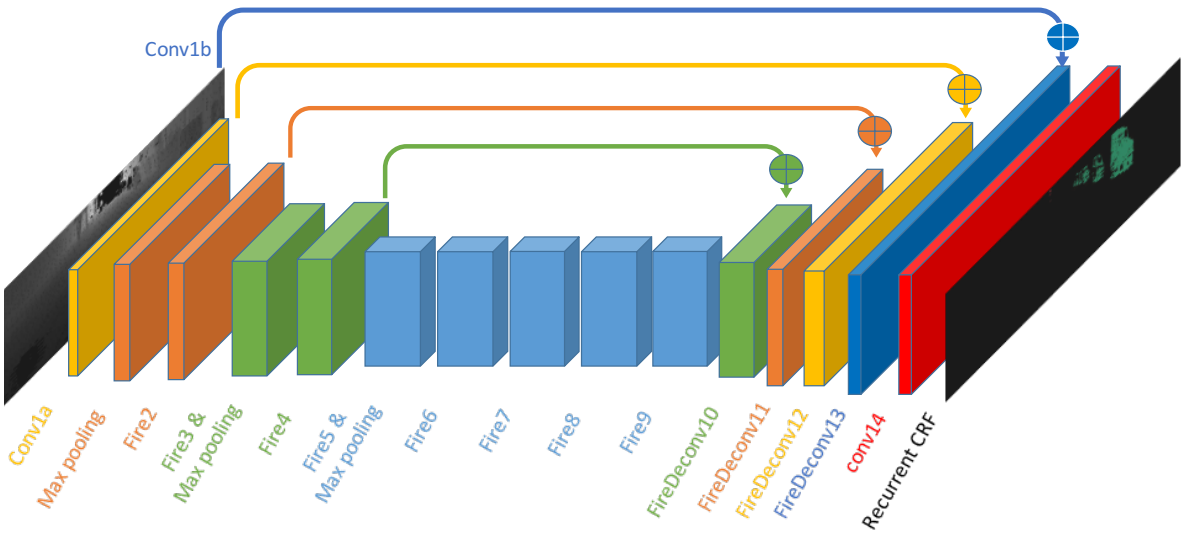


Figure 3-7: SqueezeSeg network architecture [15]

neighbouring two points belong to two different objects. Setting a threshold θ to the angle β makes it possible to decide whether the neighbouring points belong to the same object. If β is smaller than the threshold θ, the two points will be segmented into two different clusters. Otherwise, the two points will be considered belonging to the same cluster. The computation is directly approached in the range image. A possible failure circumstance is that the object is planar, such as a wall, and oriented nearly parallel to the laser beam [14]. In this case, a single object will be separated into multiple clusters.

3-2-2 SqueezeSeg

Similar to Depth clustering [14], SqueezeSeg [15] also segments the point cloud in the range image as described in Section 3-2-1. However, different from Depth clustering, SqueezeSeg uses Convolutional Neural Networks (CNN) and Conditional Random Field (CRF) to parse the range image for the semantic segmentation. The CRF model is reformulated as a Recurrent Neural Network (RNN) module and can be trained end-to-end together with the CNN model [15].

Network architecture

The network architecture of SqueezeSeg is shown in Figure 3-7. The input is the range image and the output is the label prediction for each pixel in the range image. The structures of the *FireModule* and the *FireDeconv* layers are shown in Figure 3-8. Comparing to the normal convolution and deconvolution layers, the *FireModule* and the *FireDeconv* have less parameters and make the training and inference faster [15].

Due to the loss of low-level details in down-sampling operations such as max-pooling, semantic segmentation label maps predicted by CNN models tend to have blurry boundaries [15]. Accurate point-wise label prediction requires understanding not only the high-level semantics

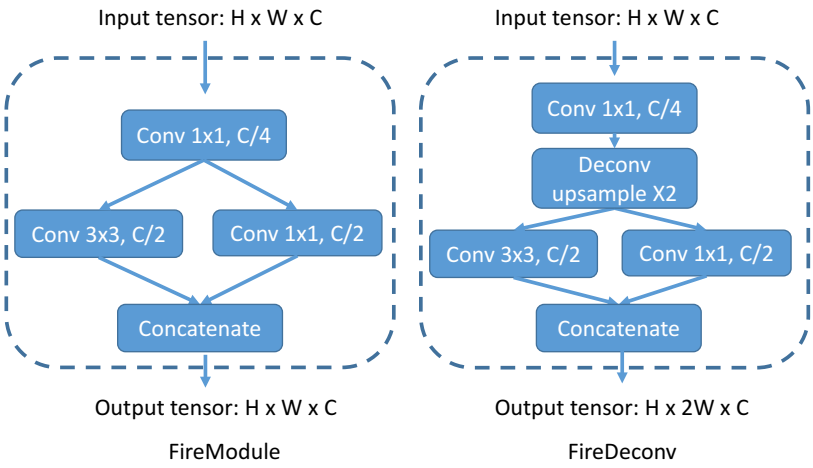


Figure 3-8: Structure of a *FireModule* (left) and a *FireDeconv* (right). [15]

of the object and scene but also low-level details [15]. Following [56], SqueezeSeg uses a CRF to refine the label map generated by the CNN [15]. The CRF model employs the following energy function for the label prediction $\mathbf{c}$:

$$E(\mathbf{c}) = \sum_i u_i(c_i) + \sum_{i,j} b_{i,j}(c_i, c_j) \tag{3-5}$$

where c_i denotes the predicted label of the i-th point. The unary potential term $u_i(c_i) = -\log P(c_i)$ considers the predicted probability $P(c_i)$ from the CNN classifier [15]. The binary potential terms define the "penalty" for assigning different labels to a pair of similar points and is defined as $b_{i,j}(c_i, c_j) = \mu(c_i, c_j) \sum_{m=1}^{M} \omega_m k^m(\mathbf{f_i}, \mathbf{f_j})$ where $\mu(c_i, c_j) = 1$ if $c_i \neq c_j$ and 0 otherwise, k^m is the m-th Gaussian kernel that depends on features $\mathbf{f}$ of point i and j and ω_m is the corresponding coefficient [15]. In SqueezeSeg, two Gaussian kernels are used:

$$\omega_1 \exp\left(-\frac{\|\mathbf{p_i} - \mathbf{p_j}\|^2}{2\sigma_\alpha^2} - \frac{\|\mathbf{x_i} - \mathbf{x_j}\|^2}{2\sigma_\beta^2}\right) + \omega_2 \exp\left(-\frac{\|\mathbf{p_i} - \mathbf{p_j}\|^2}{2\sigma_\gamma^2}\right) \tag{3-6}$$

where $\mathbf{p}(\tilde{\theta}, \tilde{\phi})$ is the angular position and $\mathbf{x}(x, y, z)$ is the cartesian coordinates. σ_α, σ_β, σ_γ are three hyperparameters chosen empirically [15].

Minimizing the above CRF energy function yields a refined label assignment [15]. In the SqueezeSeg, the minimization of the energy function is reformulated as a RNN module as shown in Figure 3-9.

After getting the probability map from the CNN layers, it is filtered by the Gaussian kernel calculated according to Eq. (3-6). This step is also called message passing in [57] since it essentially aggregates probabilities of neighbouring points. Then, a 1×1 convolution layer is applied to the aggregated probability map to re-weight the probability map and use a "compatibility transformation" to decide how much it changes each point's distribution [15]. The initial probability map is added to the output of the 1×1 convolution and fed into a softmax function. This procedure is iterated 3 times to achieve an accurate label map.

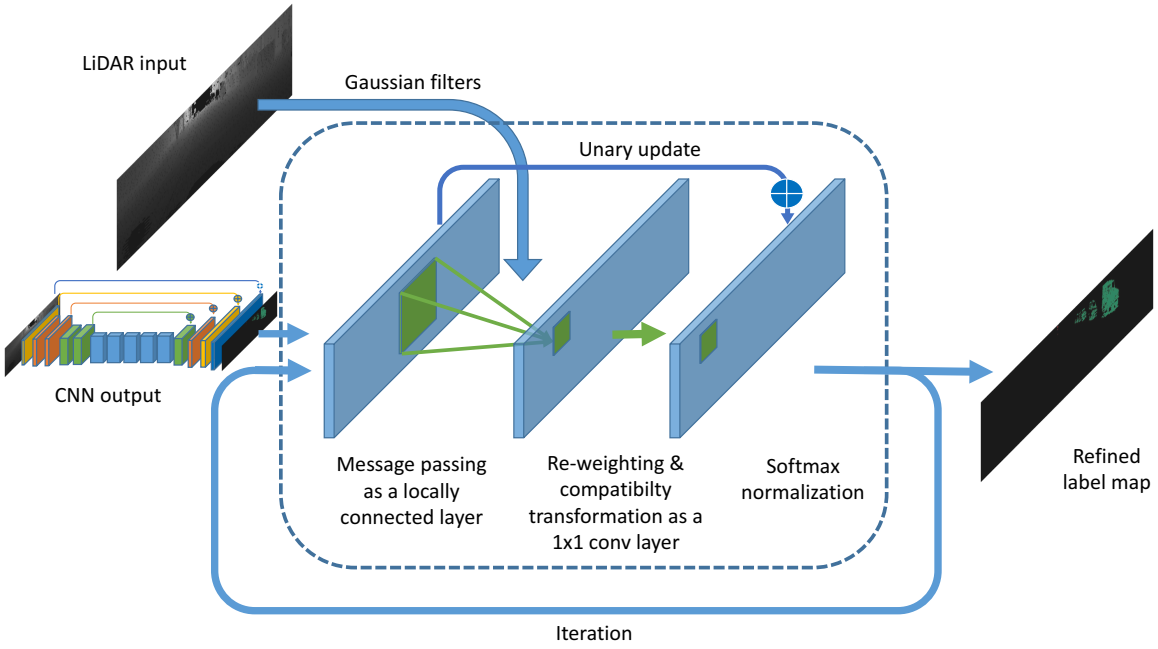


Figure 3-9: Conditional Random Field (CRF) as an RNN layer. [15]

Instance-level segmentation

Since SqueezeSeg only provides the semantic segmentation of the point cloud, the instance-level labels are then obtained by applying a conventional clustering algorithm: Density-Based Spatial Clustering of Applications with Noise (DBSCAN).

DBSCAN first finds the *core samples* in the point cloud. A *core sample* is a point with a dense neighborhood. Two parameters are used to decide a *core sample*: the number of neighbouring points (*min_samples*) and the maximum distance (*eps*) [58]. More formally, a core sample is defined as a sample in the dataset such that there exist *min_samples* other samples within a distance of *eps* [58]. A cluster is a set of core samples that can be built by recursively taking a core sample, finding all of its neighbors that are core samples, finding all of their neighbors that are core samples, and so on [58]. A cluster also has a set of non-core samples, which are samples that are neighbours of a core sample in the cluster but are not themselves core samples [58].

3-2-3 Pointnet classification and box regression

The region proposal step introduced in Section 3-2-1 and 3-2-2 will segment the point cloud into clusters that are possible to be the target objects. However, knowing that there are possible objects in the point cloud is not enough for the perception system of the autonomous vehicle. It's necessary that the object class and an amodal 3D bounding box can also be provided which can help the later system do a better judgment. So, in the second step of the Lidar-based detector, a classifier and a bounding box regressor are designed to classify the object and estimate the object bounding box.

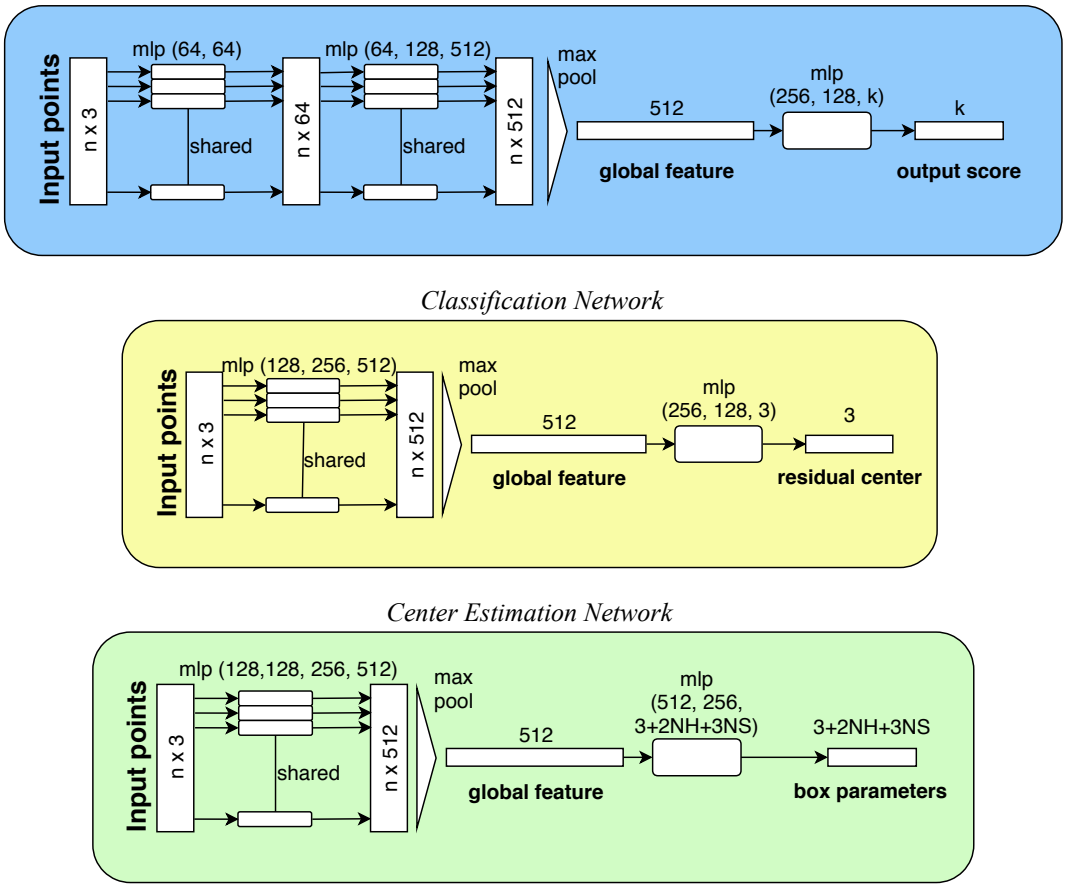


Figure 3-10: Three different Pointnets used in the thesis work. They are modified according to the networks in [2] and [10]. The MLP extracts the feature from each point and the max pooling layer aggregates the global feature from all the point features.

Network details

Pointnet [2] and it's optimized version Pointnet++ [26] are two of the most successful networks so far that directly consume the point cloud and learn the point cloud features. As introduced in Section 1-2, there are multiple encoding methods of the point cloud. To extract the feature from the data, the data should be kept in its original formation and should not pass through too many preprocessing stage. That's why the authors of Pointnet and Pointnet++ feed the point cloud directly into the network. Due to the success achieved by Pointnet in the classification benchmark in ModelNet40 [59], this thesis work modifies the original Pointnet for the road users classification and bounding box regression.

Three different Pointnets are applied in the thesis: a classification network, a center estimation network and an amodal 3D box estimation network. The basic structures of the three different Pointnets used in the thesis work are shown in Figure 3-10. Before feeding the clusters into the network, the cluster point cloud coordinates will be centralized to the center of the coordinate system as shown in Figure 3-12.

Classification network As can be seen in the top image of Figure 3-10, the input of the classification network is a $n \times 3$ matrix, where n is the number of points in each object and 3 corresponds to the x, y, z information channel of each point. Extra information channels like the reflectance and norm of each point can also be used as input. However, results in the experiment show that the coordinates information is already enough for the classification and bounding box regression.

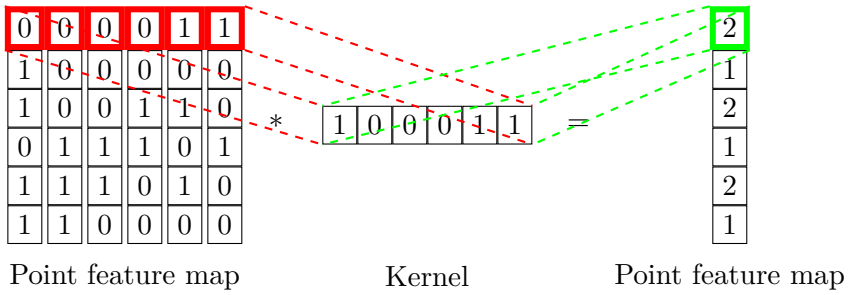


Figure 3-11: MLP example

Since the point cloud is composed of unordered points, the calculated global feature should be invariant to the input permutations of the points. The designed symmetric function is as follows:

$$\text{global feature} = f(h(\mathbf{p_1}), h(\mathbf{p_2}), h(\mathbf{p_3}), ..., h(\mathbf{p_n})) \tag{3-7}$$

where $\mathbf{p_i}$ represents the information vector of i-th point, h is the MLP and f is the max pooling layer. The MLP extracts the feature from each point and the max pooling layer aggregates the global feature. Since the max pooling layer is a symmetric function, the global feature is invariant to the input permutations. The MLP used in the network is basically a convolutional layer (Section 2-4-3) with a one-dimensional kernel as shown in Figure 3-11.

After getting the global feature vector of the object point cloud, the feature vector is further fed into a few MLPs to get the output. The output of the classification network is a vector of length k, where k is the number of classes. Feeding this output vector into the softmax function (Eq. (3-8)) gives the classification vector whose elements represent the probabilities of the object belonging to each class. Choosing the index of the maximum argument in the classification vector gives the classification result.

$$\sigma(\mathbf{z}) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, ..., K \tag{3-8}$$

Center estimation network Since the center of the cluster point cloud is usually not the real center of the object, a center estimation network is built to estimate the amodal object center. The design of the center estimation network is quite similar to the classification network. The outputs of the center estimation network are the first stage residual values (Δx_1, Δy_1, Δz_1) between the point cloud center and the real object center as shown in the right figure of Figure 3-12.

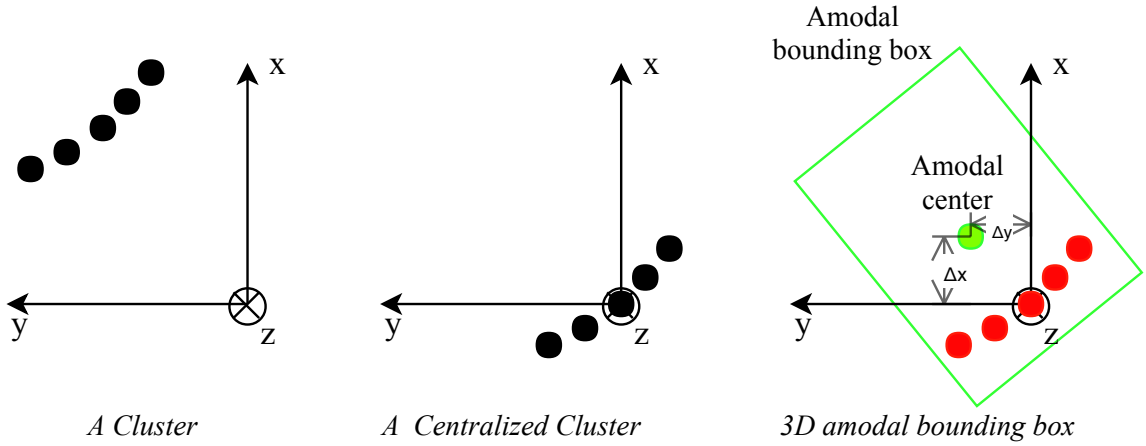


Figure 3-12: Before feeding the cluster into the network, the cluster point cloud coordinates will be centralized to the center of the coordinate system. After feeding the cluster into the network, the class and the bounding box will be estimated.

Amodal 3D Box estimation network The output vector of the amodal 3D box estimation network is a bit more complicated due to the parameters for a 3D bounding box. The first three values are the second stage residual values (Δx_2, Δy_2, Δz_2) between the point cloud center and the real object center. In the final step, the residual values from the two stages will be added up together to estimate the real object center as in Eq. (3-9). (The point cloud center is the origin of the coordinate system since the point cloud is centralized.)

$$\begin{aligned} x &= 0 + \Delta x_1 + \Delta x_2 \\ y &= 0 + \Delta y_1 + \Delta y_2 \\ z &= 0 + \Delta z_1 + \Delta z_2 \end{aligned} \tag{3-9}$$

It is assumed that all the bounding boxes are parallel to the z axis as shown in Figure 3-13. In this case, after getting the estimated amodal object center coordinates (x, y, z), four more parameters are required for the bounding box to be settled: *height* (h), *width* (w), *length* (l) and *heading angle* (γ) (around z axis). Following the previous works [10][5], a hybrid of classification and regression formulations is used here to estimate the box size and heading angle. NH anchor heading angles and NS anchor box sizes are predefined. Here, one anchor box is predefined for each class. The classification results of the classification network will decide which anchor box to use. The residual box size ($\Delta h, \Delta w, \Delta l$) for each predefined will also be predicted, which brings $3 \times NS$ more output values. The network will classify heading (NH scores for heading) to those predefined heading categories as well as predict residual values ($\Delta\gamma$) for each category, which brings $2 \times NH$ more output values. Finally, there are $3 + 2NH + 3NS$ output values from the amodal 3D Box estimation network.

Object point cloud resampling

In the original definition of Pointnet, there are no limitations on the number of points for each object fed into the network, so each cluster from the former step can contain any number

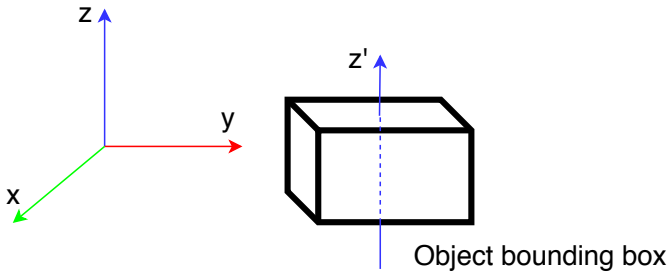


Figure 3-13: All the object bounding boxes are assumed to be parallel to the z axis.

of points. In this case, the objects cannot be concatenated together and the batch size of the Pointnet can only be 1. The experiment shows that if the training batch size equals 1, it leads to the training loss of Pointnet not converge after 0.9 million steps of training. The more detailed information will be presented in Section 4-3-3. A good solution is to resample all the clusters and to make them have the same number of points and to concatenate multiple clusters into a single tensor. Following Frustum Pointnet[10], the number of points in each object is chosen as 512.

Upsampling For the objects having less than 512 points, the point cloud has to be upsampled to make up for 512 points. One commonly used upsampling method is Moving Least Squares (MLS). It first fits a surface to the local neighborhood points within a certain distance and new points come from the surface. A newly released research PU-Net [60] use deep learning method for the point cloud upsampling. However, these methods are expensive in the computation and time-consuming for an online detection system. Following Frustum Pointnet[10], random points from the original point cloud are repeated until there are 512 points in the point cloud.

Downsampling For the objects having more than 512 points, the point cloud has to be downsampled to 512 points. One commonly used downsampling method is the farthest points downsampling. It chooses a random point as the starting point and finds the farthest point to the current point as the next point. This process is finished until 512 points are found. However, this method is expensive in the computation since it keeps computing the distances from one point to the rest points. Finally, a random downsampling method is applied in the thesis which randomly picks up 512 points in the point cloud.

Though the resampling method is simple, the experiment in Section 4-3-3 shows that resampling the objects does little influence on the results while the inference.

Networks training loss

The final multi task training loss is composed of two parts: object classification loss (L_{obj_cls}) and bounding box regression loss. The bounding box regression loss is composed of: stage one center regression loss (L_{c1_reg}), stage two center regression loss (L_{c2_reg}), box regression loss (L_{box_reg}), heading angle classification loss (L_{head_cls}), heading angle regression loss (L_{head_reg}) and the corner loss ($L_{corners}$) as follows:

$$L_{multi-task\ loss} = \lambda_1 L_{obj\_cls} + \lambda_2 (L_{c1\_reg} + L_{c2\_reg} + L_{box\_reg} + L_{head\_cls} + L_{head\_reg} + L_{corners}) \tag{3-10}$$

All the classification loss is the cross-entropy loss (Section 2-4-5) and all the regression loss is the Huber loss (Section 2-4-5). The last term corner loss ($L_{corners}$) is the L1 loss (Section 2-4-5) calculated by comparing the eight corners of the estimated bounding box and the eight corners of the ground-truth bounding box. λ_1 and λ_2 are the two coefficient to balance the classification loss and bounding box regression loss.

3-3 Multimodal Fusion Design

The final step of the system is to fuse the results from the image-based detector and the Lidar-based detector to get more convincing results.

3-3-1 Correspondence

In order to fuse the detection results from two different sensors, the first step is to find out the correspondence between the two types of results. The basic scheme is shown in Figure 3-14. Since the Lidar point cloud is a 3D data format while the image is a 2D data format, the detected 3D bounding box from the Lidar detector should first be projected to the RGB image plane to find the correspondence with the Image detection results in the same scope.

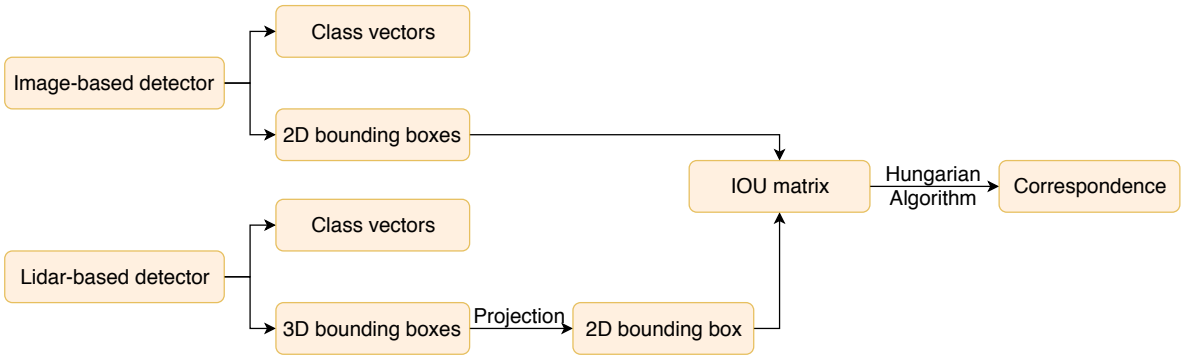


Figure 3-14: Scheme to find out the correspondence between image-based detection results and Lidar-based detection results

After projecting the 3D bounding box from Lidar detector to the image plane, a rough 2D bounding box can be extracted according to the projection of the 3D bounding box. After getting the 2D bounding box of the Lidar detection results, an IOU matrix can be computed by calculating the IOU of every two detected 2D bounding boxes from the image detector and the Lidar detector. The IOU matrix is shown in Figure 3-15. Each element of the IOU matrix $a_{(i,j)}$ represents the IOU between the two 2D bounding boxes of the i_{th} detection result of the Lidar detector and the j_{th} detection result of the image detector. So, each element $0 \leq a_{(i,j)} \leq 1$ where $0 \leq i \leq n-1$, $0 \leq j \leq m-1$. Those columns and rows that only have zeros inside will be removed from the matrix because there is no correspondence for these objects.

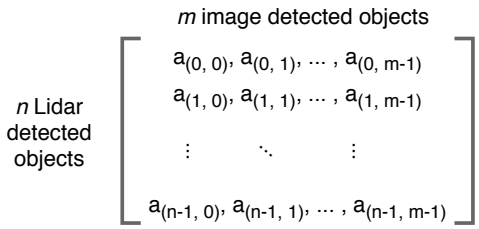


Figure 3-15: *n-by-m* IOU matrix

Hungarian algorithm

After getting the IOU matrix, the Hungarian algorithm is applied here to find the one-to-one correspondence. Hungarian algorithm is a good solution for the assignment problem. It consumes the cost matrix and finds the minimum cost assignment between the column and row indexes. So, before feeding the IOU matrix to the Hungarian algorithm, each element in the IOU matrix should be substituted by its additive inverse since the corresponding IOU need to be maximized. An example is given in Table 3-1.

	Image detected object 1	Image detected object 2	Image detected object 3	Image detected object 4
Lidar detected object 1	-0.62	**-0.76**	-0.26	-0.74
Lidar detected object 2	-0.83	-0.28	-0.40	**-0.91**
Lidar detected object 3	**-0.45**	-0.48	-0.09	-0.35

Table 3-1: Example results from the Hungarian algorithm. Each element of the IOU matrix is substituted by its additive inverse. The column and row indexes of the bold values are the indexes of the corresponding detection results found by the Hungarian algorithm. (Correspondence: Image detected object 1 & Lidar detected object 3, Image detected object 2 & Lidar detected object 1, Image detected object 4 & Lidar detected object 2)

After finding the corresponding detection results, the IOU between all the corresponding results will be checked. Those corresponding results that have a rather low IOU (smaller than the set threshold) will be given up.

3-3-2 Fusion of classification vectors

In the former step, the corresponding objects are founding in the two different results. Then the classification vectors of the corresponding results have to be fused since the classification from the two different detectors might be different. The combination of the classification can be implemented in multiple ways. Basically, the methods can be classified into two categories: Fixed combination rules and Trained combination rules.

Fixed combination rules

The fixed combination rules refers to those methods based on a fixed rule that does not depend on the data-driven training, such as mean, maximum and voting combination. Different classification vector can be combined with the mean value of each class probability. This is a simple combination but if some classifiers are badly trained, the output is not reliable. The maximum combination is to trust the most confident classifier.

The voting method is another commonly used method. Assume the output of all the classifiers form the decision vector $\mathbf{d}$ defined as $\mathbf{d} = [d_1, d_2, d_3, ..., d_n]^T$ where $d_i \in \{c_1, c_2, c_3, ..., c_m\}$, c_i denotes the label of the i-th class [61]. A binary characteristic function be defined as follows:

$$B_j(c_i) = \begin{cases} 1 & \text{if } d_j = c_i \\ 0 & \text{if } d_i \neq c_i \end{cases} \tag{3-11}$$

Then the general voting routine is defined as:

$$E(d) = c_i \quad \text{if } \forall_{t \in \{1,...,m\}} \sum_{j=1}^{n} B_j(c_t) \leq \sum_{j=1}^{n} B_j(c_i) \geq \alpha \cdot m + k(d) \tag{3-12}$$

where α is a parameter and $k(d)$ is a function that provides additional voting constraints [61]. A commonly used conservative voting rule is given if $k(d) = 0$ and $\alpha = 1$, meaning that the class is chosen when all classifiers produce the same output [61]. This rule can be liberalised by lowering the parameter α [61]. If $\alpha = 0.5$, it is the commonly known majority vote combination.

Another commonly known fixed combination rule is **Dempster-Shafer theory**, which is also used in the thesis work.

Dempster-Shafer theory

The Dempster-Shafer theory is a generalization of the Bayesian theory of subjective probability [48]. Whereas the Bayesian theory requires probabilities for each question of interest, Dempster-Shafer (DS) theory allows us to base degrees of belief for one question on probabilities for a related question [62], which makes it possible to represent evidence about the object features coming from individual sensor detectors and their classification likelihood into a common representation [12].

DS theory represents the world in a set of mutually exclusive propositions known as the frame of discernment (Ω) [48]. It applies belief functions to distribute the evidence about the propositions, which is also known as a Basic Belief Assignment (BBA) as described in Eq. (3-13) [48].

$$\begin{aligned} m(\emptyset) &= 0 \\ \sum_{A \subseteq \Omega} m(A) &= 1 \end{aligned} \tag{3-13}$$

In the application of this thesis, A represents the classification vector computed by feeding the output of the classification network into the softmax function in Eq. (3-8) and Ω represents all the possible classification vectors. Any subset A of Ω with $m(A) > 0$ for a particular belief function is called a focal element of that function [48]. Beliefs from different sources can be combined with specific fusion operators in specific situations. The most commonly used combining rule proposed by [62] is shown as follows:

$$\begin{aligned} m_{12}(A) &= \frac{\sum_{B \cap C = A} m_1(B) m_2(C)}{1 - K_{12}};\ A \neq \emptyset \\ K_{12} &= \sum_{B \cap C = \emptyset} m_1(B) m_2(C) \end{aligned} \tag{3-14}$$

where K_{12} is known as the degree of conflict. B and C are the class probabilities from the image-based detector and Lidar-based detector separately, while A is the class probabilities after fusion. Dempsters rule analyses each piece of evidence to find conflicts and uses it to normalize the masses in the set [48].

The DS theory is finally chosen as the applied combination rule in the thesis work since it is possible to combine the classification vectors from different sensors using a different confidence.

Trained combination rules

The trained combiner combines the classification vectors from different classifiers into a single feature vector and feeds them into a new classifier as shown in Figure 3-16.

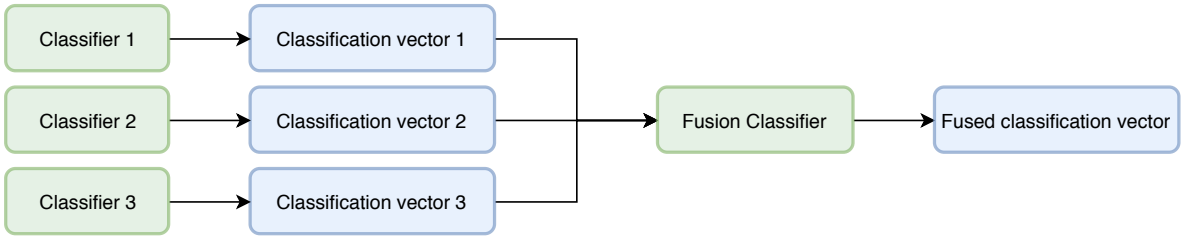


Figure 3-16: Trained combination rules

The fusion classifier can be designed as a classical classifier such as a Support Vector Machine (SVM) or a simple Neural Network (NN). The trained rules have potentially better performance than fixed rules. However, this combination rule is data-driven and requires high memory and longer time to run.

3-3-3 Residual detection results

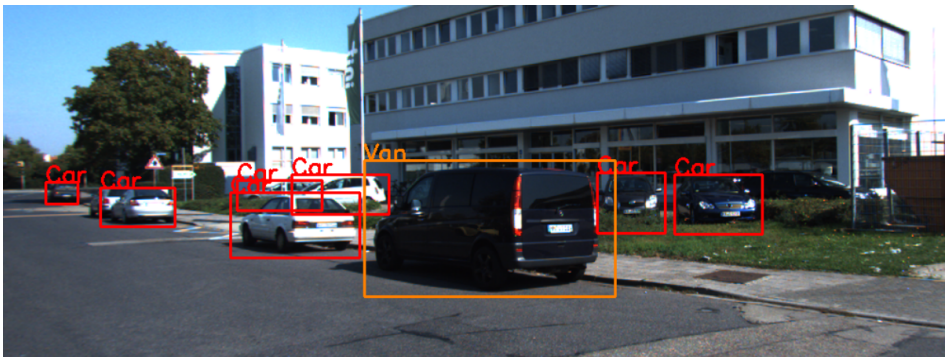
There are probably some detection results either from the image-based detector or from the Lidar-based detector that have no corresponding results.

For those detection results from the Lidar-based detector that do not have the correspondence in the image-based detection results, they can be either kept or given up since the

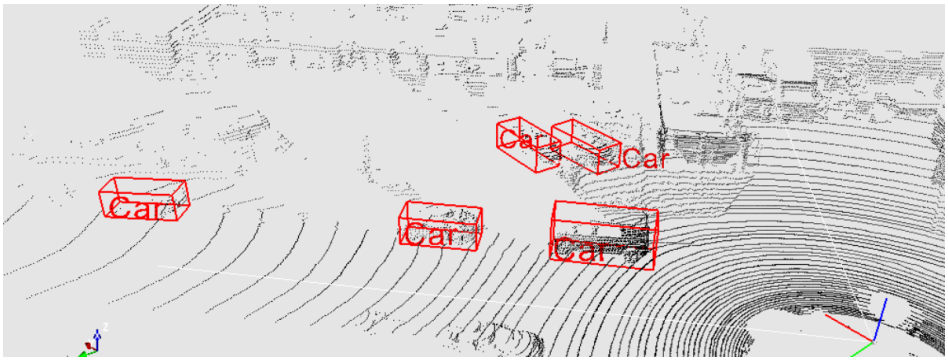
2D information can be extracted by projecting the 3D bounding box onto the RGB image. Whether to keep those Lidar detection results that do not have correspondence will be tested in Section 4-4. However, for those detection results from the image-based detector that do not have the correspondence in the Lidar-based detection results, they are given up since it is not possible to get the depth information from only a single RGB image and it is not possible to get the 3D position of the detection results. This will have to be improved in the future research.

3-4 Conclusion

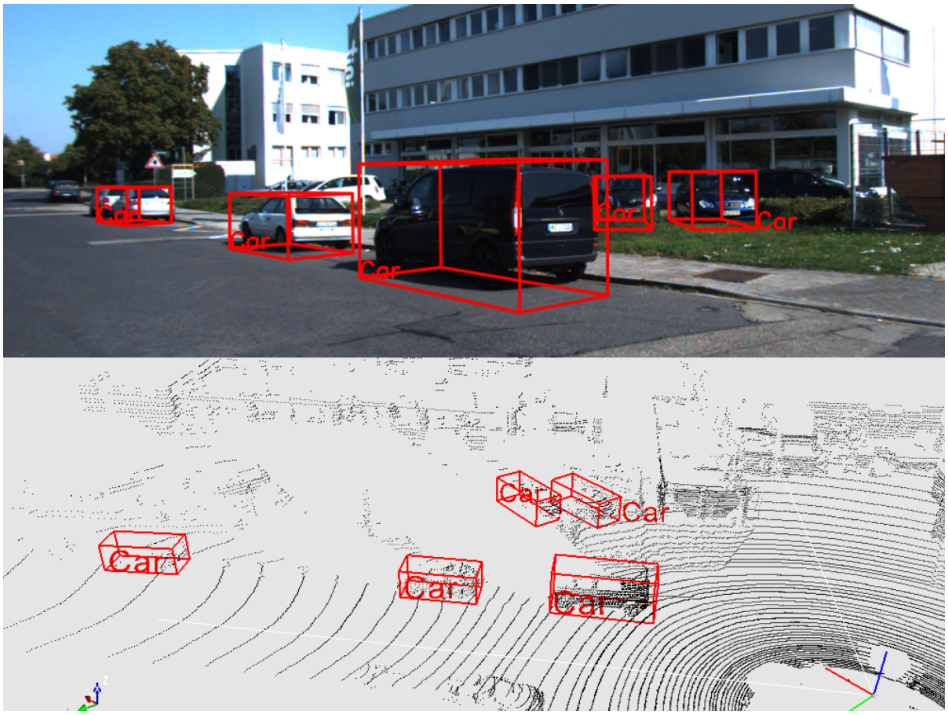
The example outputs of each module are shown in Figure 3-17. The image-based detection results are presented in the RGB image as the 2D bounding boxes and the class labels. The Lidar-based detection results are presented in the Lidar point cloud as the 3D bounding boxes and the class labels. The results after fusion are the final detection results. The 3D bounding boxes in the point cloud are projected onto the RGB image.



(a) Image-based detection results.



(b) Lidar-based detection results.



(c) Fusion results.

Figure 3-17: Example detection result.

Chapter 4

Experiments and Results

In this chapter, the dataset used in all the experiments, the designed experiments and the results are presented. The dataset used in the thesis work is KITTI [1] that will be introduced in Section 4-1. The training and validation split used in all the experiments is also presented in Section 4-1. The training parameters of SqueezeDet [13] and the image-based detection results will be presented in Section 4-2. The following Section 4-3 presents the validation and parameter tuning of each part of the Lidar-based detector (*Region proposal* and *Pointnet classification and regression*). Section 4-4 presents the results of the late fusion part. The comparisons of the proposed method to other state-of-the-art works are shown in Section 4-5. Typical failure modes are also presented in Section 4-6.

4-1 KITTI Dataset

KITTI Vision Benchmark Suite [1] is a dataset created by Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago. It is one of the biggest computer vision benchmarks in the autonomous driving research field. The dataset is collected by their autonomous driving platform Annieway. The setup of the sensors on Annieway is shown in the Figure 4-1. There are two grayscale cameras, two colour cameras, a Velodyne Laser-scanner and an inertial navigation system. The cameras and Lidar are all mounted on top of the car. Besides the raw dataset, KITTI also provided multiple benchmarks including stereo, flow, odometry, object detection or tracking benchmarks. The datasets provided in each benchmark are slightly different. This paper uses the dataset from the object detection benchmark which provides 7481 labelled frames as training data, and another 7518 unlabelled frames as testing data. Here, a frame means all the data (RGB image, point cloud, etc.) collected by different sensors at the same time step. Submitting the detection results of the testing data to the server will get a ranking among all the submitted methods.

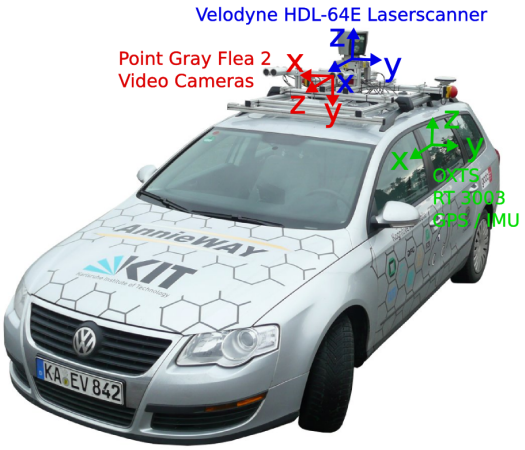
The label files from the object detection benchmark are supplied as text files. Each row in a label file describes a single ground-truth object, which contains the following information of

Column number	Name	Description
1	type	The label of the objects.
2	truncated	Float from 0 (non-truncated) to 1 (truncated), where truncated refers to the object leaving image boundaries.
3	occluded	Integer (0,1,2,3) indicating occlusion state: 0 = fully visible, 1 = partly occluded, 2 = largely occluded, 3 = unknown.
4	alpha	Observation angle of object.
5-8	bbox	2D bounding box of object in the image (0-based index): contains left, top, right, bottom pixel coordinates.
9-11	dimensions	3D object dimensions: height, width, length (in meters).
12-14	location	3D object location x, y, z of the mid-point of the object in camera coordinates (in meters).
15	rotation_y	Rotation ry around Y-axis in camera coordinates.

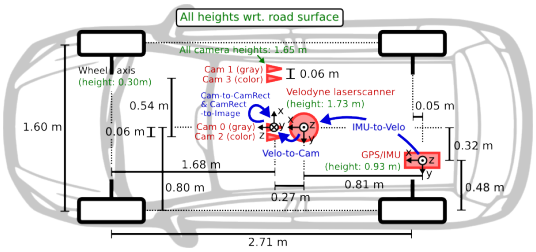
Table 4-1: KITTI label file format [1]

the ground-truth object as listed in the Table 4-1. There are eight classes labeled in KITTI: *Car*, *Van*, *Truck*, *Pedestrian*, *Person_sitting*, *Cyclist*, *Tram* and *Misc*.

As shown in the Figure 4-1a, camera and Lidar use two different coordinates (red for the camera and blue for the Lidar) and all the ground-truth information is given in the camera coordinates but it's easy to convert it to the Lidar coordinates according to the calibration information. In all the experiments below, the Lidar coordinates will be used since the Lidar-based object detection is one of the main tasks of the thesis.



(a) Fully equipped vehicle



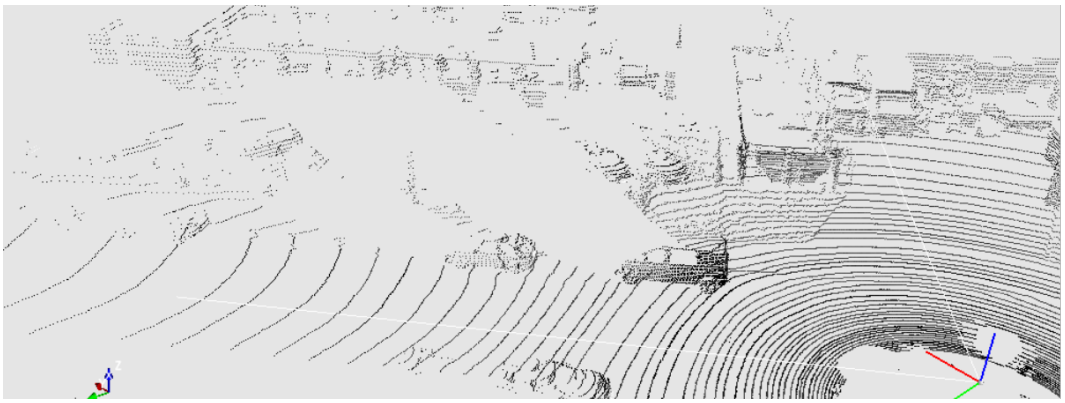
(b) Setup top view

Figure 4-1: Setup of KITTI's autonomous driving platform Annieway [1]

An example frame from KITTI is shown in Figure 4-2. The upper figure is the RGB image of size 1242×375. The bottom figure is the visualization of the point cloud corresponding to

the upper RGB image.

(a) KITTI image example



(b) KITTI point cloud example

Figure 4-2: Example KITTI frame [1]

The difficulty to detect objects in different circumstances is different. Some objects are harder to detect such as the occluded objects, objects in the far range, etc. The further an object is from the vehicle, the smaller the object is in the RGB image and the fewer points are reflected by the object in the point cloud. So, KITTI object detection benchmark provides three difficulty levels to evaluate the object detection system as listed in Table 4-2. These three levels are separated by the object size in the image, the occlusion and the truncation.

Mode	Description
Easy	The object is bigger than 40 pixels in the RGB image and fully visible in the RGB image. The maximum truncation is 15%.
Moderate	The object is bigger than 25 pixels in the RGB image and partly occluded in the RGB image. The maximum truncation is 30%.
Hard	The object is bigger than 25 pixels in the RGB image and difficult to see in the RGB image. The maximum truncation is 30%.

Table 4-2: Three difficulty levels of KITTI object detection benchmark [1]

Training and validation split

Since KITTI only provides the label files for the 7481 training frames, the training dataset is split into 3712 *training* frames and 3769 *validation* frames exactly the same with MV3D [3]. The split provided by MV3D [3] takes into account that the dataset in the KITTI object detection benchmark comes from the sequential raw data of KITTI, so there are some quite similar frames in the dataset. The split provided by MV3D [3] separates the different frames into the training and validation sets. After the training and validation split, the data composition is shown in Figure 4-3.

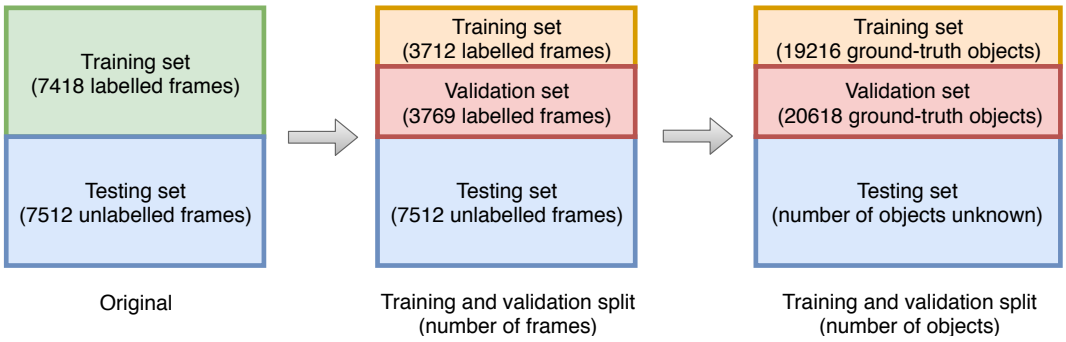


Figure 4-3: KITTI dataset for object detection benchmark

Unless otherwise specified, the presented results in this chapter like the results of the image-based detector, the results of the Pointnet classification and box regression and the results of the Lidar-based detection are all the results on the validation set.

4-2 Experiments on Image-based Detector

The validation of the image-based detector is based on the metric provided by the KITTI 2D object detection benchmark.

4-2-1 Metric

The metric used to measure the 2D object detection in the RGB image plane is the Average Precision (AP). AP is the summary of the precision-recall curve. Recall is basically a measurement of the fraction of ground-truth annotations covered by the detection results and precision is the fraction of the correct detection among all the detection results. Precision and recall can be calculated by the following function:

$$\begin{aligned} recall &= \frac{\#TP}{\#TP + \#FN} \\ precision &= \frac{\#TP}{\#TP + \#FP} \end{aligned} \tag{4-1}$$

where $\#TP$ is the number of True Positives (TP), $\#FN$ is the number of False Negatives (FN) and $\#FP$ is the number of False Positives (FP). The TP is judged by the Intersection Over

Union (IOU) between the detected 2D bounding box and the ground-truth 2D bounding box and the estimated probability. If both the IOU and the estimated probability are bigger than the thresholds, the detected object will be considered as a TP. So the thresholds for IOU and the estimated probability will have a large influence on the value of recall and precision. Choosing the different thresholds makes different pairs of precision and recall values. In the KITTI evaluation metric, the threshold for the IOU is fixed and the threshold for the estimated probability is changed to get the precision-recall curve. The AP is calculated by the following function:

$$AP = \sum_{n}(R_n - R_{n-1})P_n \tag{4-2}$$

where R_n denotes the n_{th} value of recall and P_n denotes the n_{th} value of precision.

4-2-2 Training

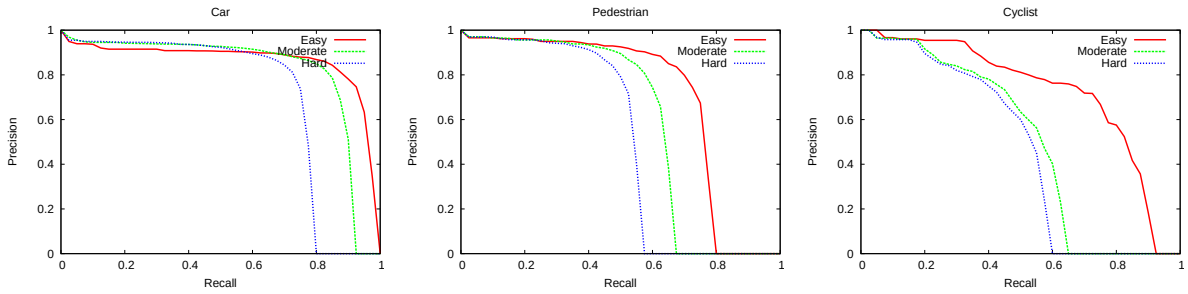
The hyperparameters of the training process are listed in the Table 4-3.

Optimizer	Stochastic Gradient Descent (SGD)
Initial learning rate	0.01
Learning rate decay factor	0.5
Learning rate decay step	10000
Batch size	20

Table 4-3: SqueezeDet training parameters

4-2-3 Results

The precision-recall curves of the SqueezeDet detection results on the validation set are shown in Figure 4-4. The average precision for *Car*, *Pedestrian* and *Cyclist* is listed in Table 4-4. The mean detection time for each image is 0.059s. More visualization of the image-based detection results will be shown in Appendix A-1.



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

Figure 4-4: Precision-recall curve of SqueezeDet detection results on validation set

	Easy	Moderate	Hard
Car	0.820	0.805	0.676
Pedestrian	0.676	0.586	0.507
Cyclist	0.708	0.503	0.457

Table 4-4: SqueezeDet 2D detection AP on validation set

4-3 Experiments on Lidar-based Detector

In this section, each module of the Lidar-based detector will be evaluated and the parameters will be tuned to optimize the output of each part.

4-3-1 Data preprocessing

KITTI ground-truth objects

Since the Pointnet from the Lidar-based detector deals with the object point cloud instead of the whole point cloud to decide the object class and the bounding box, the training of the Pointnet is based on the ground-truth object point cloud. Then, the original point cloud has to be preprocessed to get the ground-truth object point cloud.

According to the provided information for each ground-truth object as listed in Table 4-1, the position of the ground-truth object and the size of the bounding box are provided in the camera coordinates. After converting the camera coordinates to the Lidar coordinates, the position and size of the ground-truth bounding box can be obtained in the Lidar point cloud. Then, those points inside the bounding box becomes the ground-truth object point cloud. The object point cloud together with the bounding box parameters and the class label will be saved both for the training set and validation set.

The numbers of all the objects from both the training set and the validation set are listed in Table 4-5. The ground-truth objects that do not have any point inside are ignored. As given in Table 4-5, the number of ground-truth objects is unbalanced. KITTI provides much more training data for *Car* than the others.

Class	Training	Validation
Car	13941	14214
Van	1294	1580
Truck	481	597
Pedestrian	2183	2279
Person_sitting	56	166
Cyclist	720	888
Tram	205	264
Misc	336	630
Total	19216	20618

Table 4-5: Numbers of objects in KITTI object detection benchmark

Extra training data: *Miscs*

If using the Depth clustering (Section 3-2-1), there will be a problem that the class distribution of the clusters from the Depth clustering is different from the class distribution of the KITTI training data. In other words, the clusters can be any of the objects existing in reality like road users, bushes, buildings, fences, etc. However, the KITTI training data only provides ground-truth objects from eight classes as listed in Table 4-5.

Training Pointnet with only the data from KITTI ground-truth objects might leads to all the clusters be classified into these eight classes, which might cause a lot of false positives. A visualization of such a problem is shown in Figure 4-5. The bounding boxes in the image come from the projection of the Lidar detected bounding boxes. From the projected results in the image, it is easy to see that a lot of FP appears. For example, some fences are wrongly classified into *Truck*.

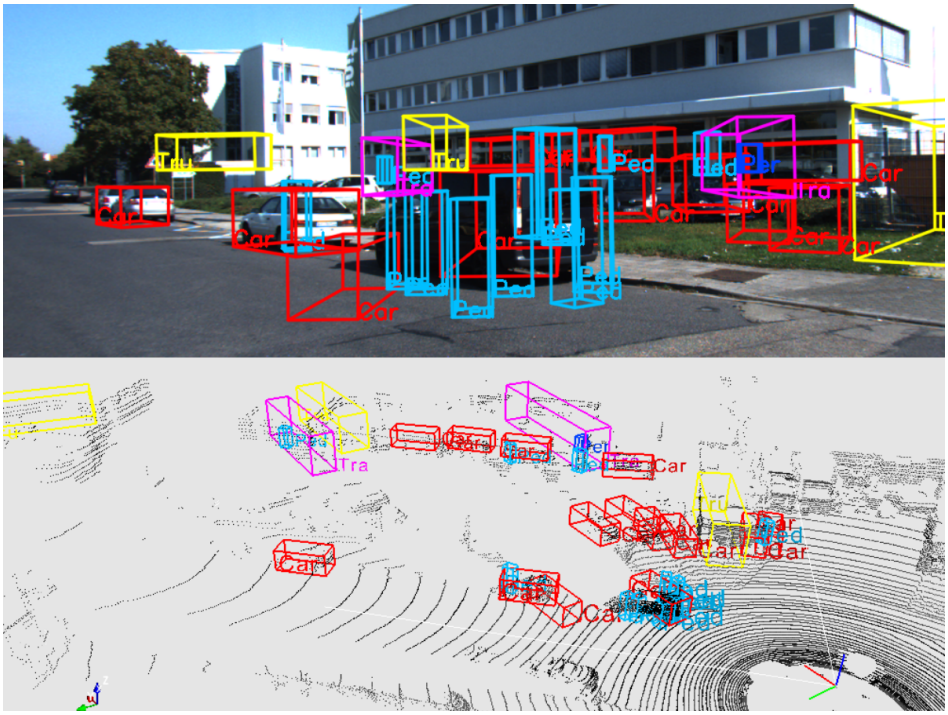


Figure 4-5: Training Pointnet with KITTI ground-truth objects might lead to all the clusters be classified into the eight classes labelled in KITTI. *Top*: Classification and box regression results projected to the image plane. *Bottom*: Classification and box regression results on the point cloud clusters. Bounding box color and class correspondence: Car, Van, Truck, Pedestrian, Person_sitting, Cyclist, Tram, Misc.

A possible solution is to add some more training data of all kinds to balance the class distribution of the training data and the clusters. The newly collected training data comes from the clusters in the training set. Those clusters in the training set that do not have overlapped area with the ground-truth objects are collected as extra training data and labelled as *Misc*. Since there are too many new *Miscs* from the clusters, only part of them are randomly selected as extra training data. The collected clusters are added according to the percentage and will be evaluated in Section 4-3-4.

4-3-2 Experiments on the region proposal

The region proposal is the first step of the Lidar-based detector, so it is quite important to ensure that the output of the segmentation algorithm makes sense. In other words, clusters from the segmentation algorithm should cover the target objects (road users) in the thesis context.

Metrics

$$\text{Segmentation recall} = \frac{\#\text{ found ground-truth object}}{\#\text{ all ground-truth object}} \tag{4-3}$$

The metric to evaluate the segmentation algorithm is defined as the recall of the results as described in Eq. (4-3), where # represents "*the number of*". A ground-truth object is defined as *found* if there exists a cluster that has a point-wise IOU over 0.7 with it. A more clear visualization of the definition is shown in Figure 4-6. The points inside the green box represent the ground-truth object and the points inside the red box represent the cluster. The point-wise IOU is defined as Eq. (4-4). If point-wise overlap ratio is bigger than 0.7, the ground-truth object is defined as found.

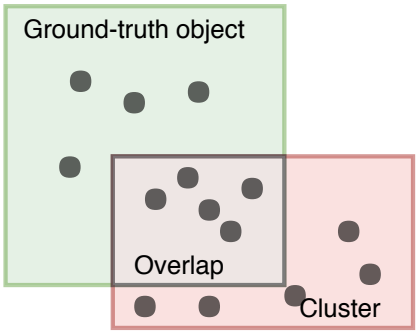


Figure 4-6: A ground-truth object (green) has a point-wise overlap with a cluster (red)

$$\text{Point-wise IOU} = \frac{\#\text{ Overlap points}}{\#\text{ Cluster points} + \#\text{ Ground-truth object points - }\#\text{ Overlap points}} \tag{4-4}$$

Results of Depth clustering

For the Depth clustering, different angle thresholds (θ) are set to the angle β and the segmentation results are shown in Figure 4-7.

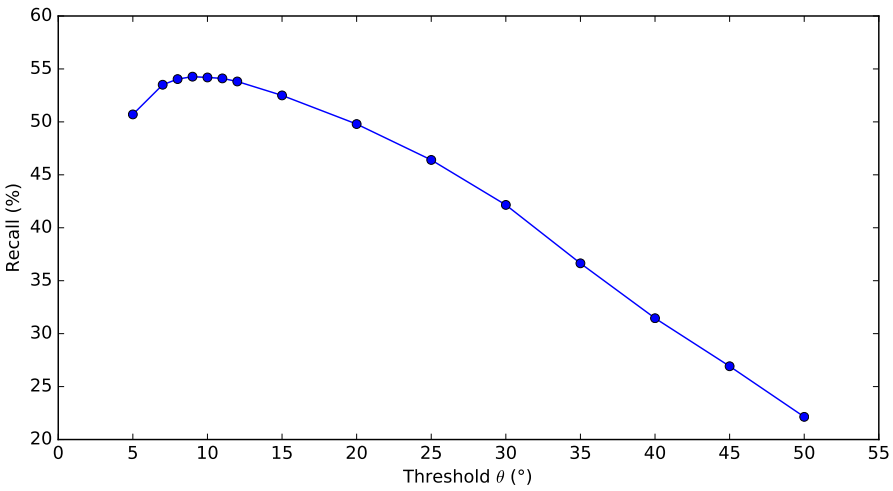


Figure 4-7: Depth clustering segmentation recall with different angle thresholds θ

As can be seen in the Figure 4-7, the segmentation recall is the best under the angle threshold as 9°. However, the recall is only 55% at the best performance. The result has been analyzed as follows.

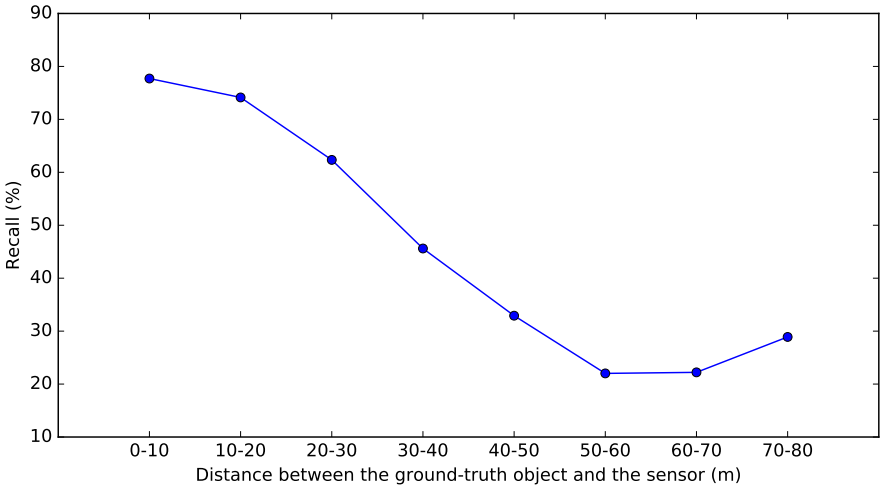


Figure 4-8: Depth clustering segmentation recall in different ranges ($\theta = 9°$)

Figure 4-8 shows that the distance from the object to the sensor has a large influence on the segmentation recall. The segmentation recall in the close range (0-20m) can reach about 75%. However, for the objects that are far away, the recall turns out to be really low as shown in Figure 4-8.

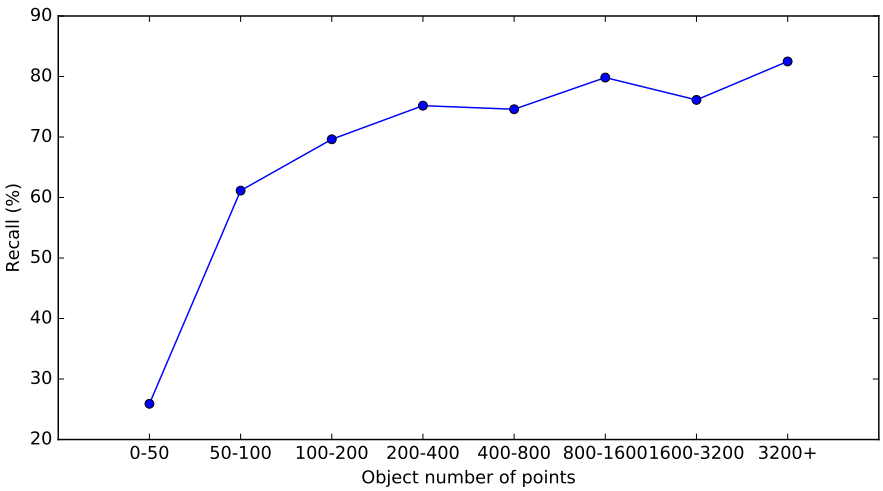


Figure 4-9: Depth clustering segmentation recall in different number of points in each object ($\theta = 9°$)

Figure 4-9 shows that the number of points also has a large influence on the segmentation recall. For those objects that have fewer points (less than 100 points), the segmentation performs worse. Among all the 39834 ground-truth objects in KITTI dataset, about one-third of the objects (14958 objects) have less than 50 points. Since the segmentation recall for objects having less than 50 points is very low, this has a large impact on the overall segmentation recall. There are two main reasons for an object having few points. The first one is that the object is far away from the sensor. The second one is that the object is occluded by other objects.

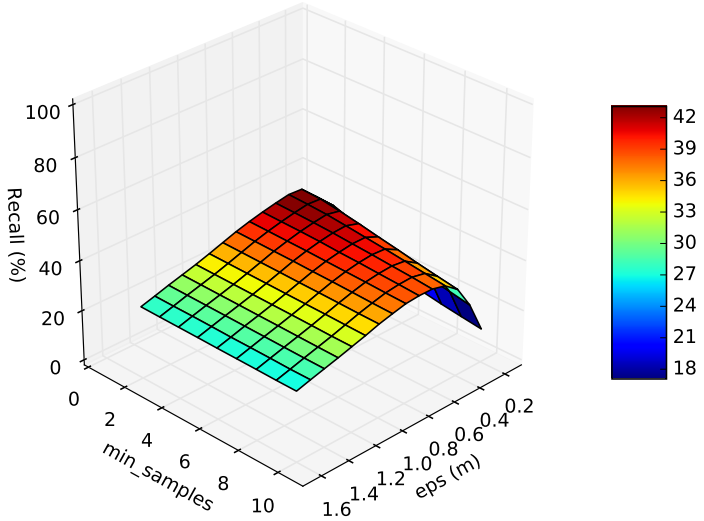


Figure 4-10: SqueezeSeg segmentation recall with different $min\_samples$ and eps

Results of SqueezeSeg

If using the combination of SqueezeSeg and DBSCAN for the region proposal, the two parameters that have to be tuned are the number of neighbouring points ($min\_samples$) and the maximum distance (eps) as described in Section 3-2-2. After applying different $min\_samples$ and eps, the segmentation results are shown in Figure 4-10.

From Figure 4-10, the segmentation recall reaches the top (43.40%) at $min\_samples = 2$ and $eps = 0.5$. However, the recall of the segmentation is still low.

Similar to the Depth clustering, the influence of the distance and the number of points is given in Figure 4-11 and Figure 4-12. The influence is quite similar to the Depth clustering. SqueezeSeg also performs better in the close range. However, SqueezeSeg performs better when the number of objects is around 1000.

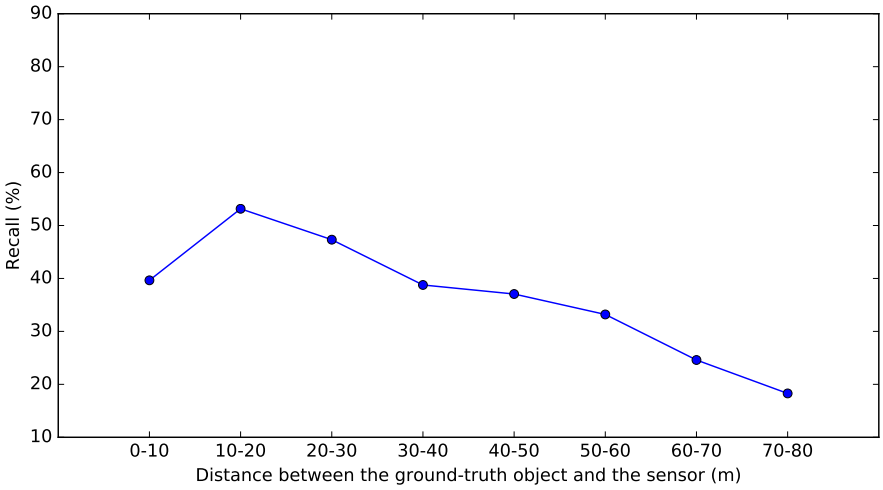


Figure 4-11: SqueezeSeg segmentation recall with different ranges ($min\_samples = 2$ and $eps = 0.5$)

4-3-3 Experiments on Pointnet

The modified Pointnet is the classifier and the bounding box regressor of the Lidar-based detection system. The validation of the Pointnet for classification and bounding box regression is to make sure that the modified Pointnet gives the meaningful outputs.

Training

As already said in Section 3-2-3, if the training objects are not resampled and all the training objects might have different numbers of points, the objects cannot be concatenated into a single array and the training batch size cannot be changed and is always 1. However, it turns out that training Pointnet with batch size 1 will make the model heavily over-fitting and the training loss will never converge.

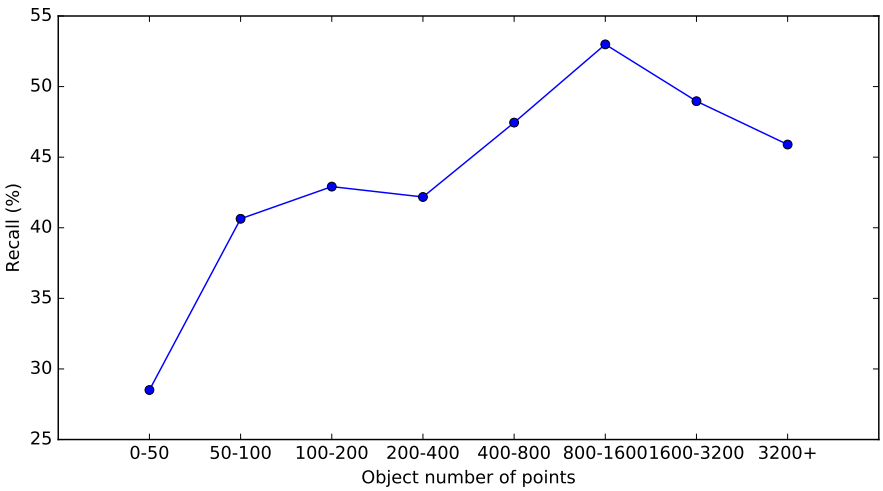


Figure 4-12: SqueezeSeg segmentation recall with different number of points in each object ($min\_samples = 2$ and $eps = 0.5$)

The proposed solution is to resample all the training objects to the same number of points using the method illustrated in Section 3-2-3. After resampling, all objects have 512 points and can be concatenated together in the form of three-dimensional arrays with size $Batch\ size \times 512 \times 3$, where 3 corresponds to the number of information channels that are used to represent each point. Here, the x, y, z coordinates of each point are used.

All the training objects used to train the Pointnet are collected from the training set and all the validation objects are collected from the validation set as described in Section 4-1. The training parameters of the Pointnet is listed in Table 4-6.

Optimizer	Adam
Initial learning rate	0.001
Learning rate decay factor	0.7
Learning rate decay step	20000
Batch size	32

Table 4-6: Pointnet training parameters

Results when training without extra *Miscs*

The training data used in this section is only the ground-truth objects from KITTI and no extra *Miscs* are added into the training data.

The classification performance is evaluated on the classification accuracy: the fraction of the correctly classified objects among all the objects. The bounding box regression performance is evaluated on the average IOU between the estimated bounding box and the ground-truth box and the bounding box accuracy. If the estimated bounding box and the ground-truth box have an IOU over 0.7, the estimated bounding box is considered as an accurate bounding box. The bounding box accuracy is the fraction of the correctly estimated bounding boxes

	Classification accuracy	Average bounding box IOU	Bounding box accuracy
Resampling	88.36%	0.80	76.97%
No Resampling	88.54%	0.79	77.14%

Table 4-7: Pointnet classification and bounding box regression results on the objects from the validation set

among the number of objects. The results are shown in Table 4-7. The validation objects are either resampled or unresampled before feeding into the network. The classification confusion matrix while resampling the objects is also shown in Table 4-8. For *Car*, *Pedestrian* and *Cyclists*, the classification accuracy is beyond 80%. The results is good since the 3D object point cloud scanned by the Lidar is incomplete since Lidar only sees one side of the object.

Actual \ Prediction	Car	Van	Truck	Pedestrian	Person sitting	Cyclist	Tram	Misc
Car	13904	230	8	32	0	20	2	18
Van	627	829	86	23	0	6	4	5
Truck	35	160	340	0	0	1	59	2
Pedestrian	42	0	0	2195	11	30	1	0
Person sitting	7	0	0	104	46	9	0	0
Cyclist	56	3	0	90	0	716	2	21
Tram	22	34	127	5	0	0	76	0
Misc	172	167	123	32	5	9	9	113

Table 4-8: Pointnet classification confusion matrix on the objects from the validation set

An important step while training is to resample all objects to 512 points. It is quite necessary to ensure the resampling makes little influence while the inference.

From Table 4-7, it can be seen that the chosen resampling method has little influence on the results of the Pointnet. However, resampling the objects makes the inference much faster than before. Since without resampling, the batch size while the inference can only be 1. The average inference time for each frame was $0.04548s$ before resampling. After resampling, the average inference time for each frame is $0.02747s$. Since resampling has little influence on the inference accuracy and makes the system faster, it will be applied in the future experiments.

Results when training with extra *Miscs*

As mentioned in Section 4-3-1, refining the model with extra *Miscs* is necessary. The extra training data (*Miscs*) comes from the clustering results. Those clusters from the training set that do not have any overlap with the ground-truth objects are randomly picked up as the extra *Miscs*. Difference portions of *Miscs* are added to the training data and the trained model in the previous section has been refined. The evaluation dataset is still based on the ground-truth objects from the validation set. The result is shown in Figure 4-13.

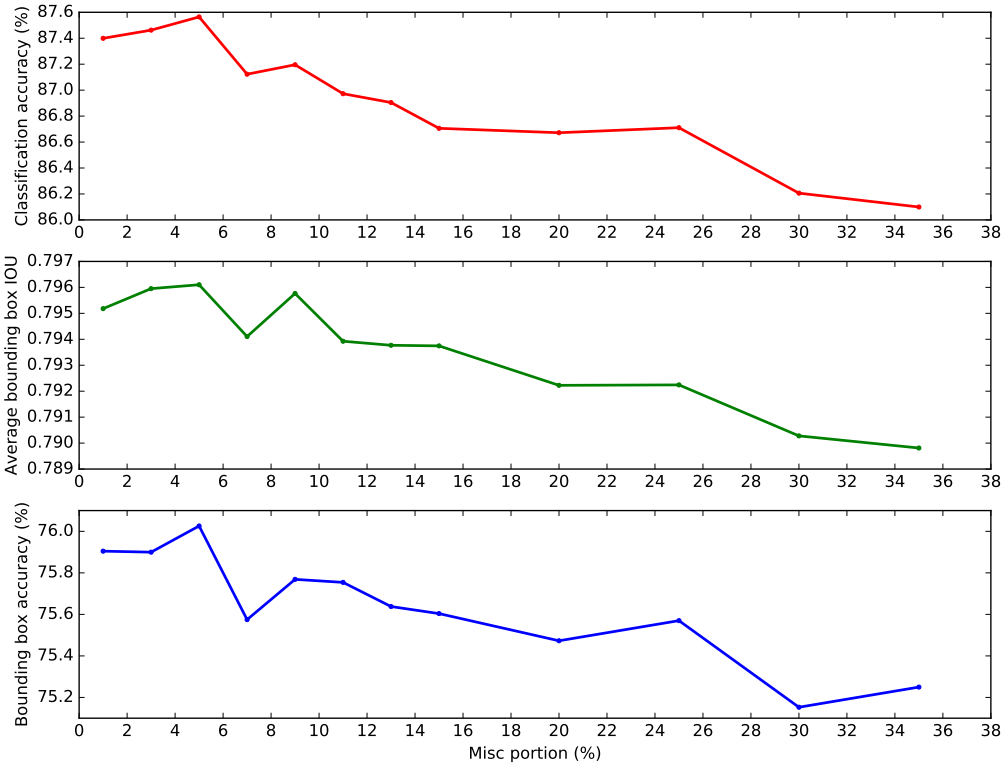


Figure 4-13: Pointnet classification and bounding box regression results on the objects from the validation set

From Figure 4-13, it is shown that adding too many *Miscs* to the training data will make the classification and bounding box regression results worse.

4-3-4 Experiments on Lidar-based detection system

The Lidar-based detection system is composed of the former two parts (*region proposal* and *Pointnet*). The validation of the Lidar-based detection system will be based on the metric for 3D object detection evaluation from the KITTI 3D object detection benchmark.

In Section 4-3-3, it shows that adding a different portion of the *Miscs* will influence the performance of the Pointnet. At the same time, it will also influence the Lidar-based detection results.

Metrics

The metric used to evaluate the Lidar-based detector is the 3D object detection metric same as the KITTI object detection benchmark. Similar to the 2D object detection metric described in Section 4-2-1, the performance of the 3D object detection is also evaluated on the AP

calculated from the precision-recall curve. The difference here is that the TP is judged by the IOU between two cuboids: the estimated 3D bounding box and the ground-truth 3D bounding box and the estimated probability.

Results with Depth clustering as the region proposal

After adding different portions of *Miscs* to the training data, the detection AP of the *Car*, *Pedestrian*, *Cyclist* under **Easy** mode is shown in Figure 4-14.

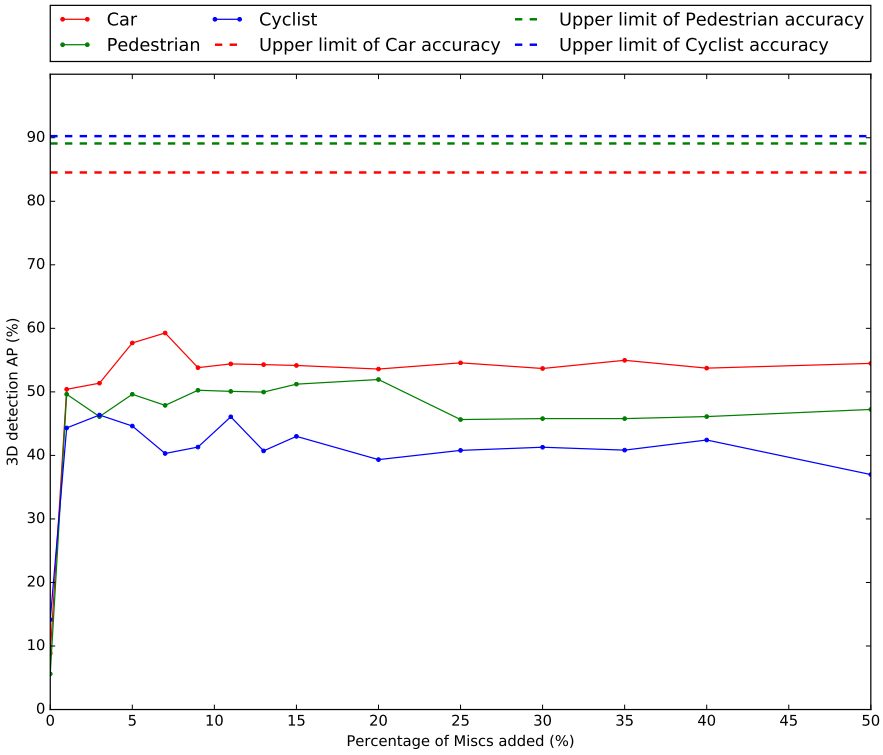


Figure 4-14: The performance of the Lidar-based detector (Depth clustering as the region proposal) on KITTI 3D object detection benchmark **Easy** mode. AP: Average Precision. The dash lines represent the results from the ground-truth object point cloud and the solid lines represent the results from the region proposal clusters.

Figure 4-14 shows that adding *Miscs* to the training data improves the detection AP a lot. For Car category, the AP improves from 22% to 65%. However, the influence of the *Miscs* portion on the detection results is not obvious. Combining the results from Section 4-3-3, adding 5% of the *Miscs* becomes the final decision.

The dash lines in Figure 4-14 are the results if feeding the ground-truth object point cloud into the Lidar-based detector instead of the clusters from the region proposal. The gap between the dash lines and the solid lines are mainly caused by the region proposal step. One reason is that the region proposal provides a lot of false positives. Another reason is that the region proposal does not give the right object point cloud, such as including multiple objects in a single cluster or separating a single object into multiple clusters.

Results with SqueezeSeg as the region proposal

After the parameters optimization for the number of neighbouring points ($min\_samples = 2$) and the maximum distance ($eps = 0.5$) in Section 4-3-2, the clusters from the SqueezeSeg and DBSCAN are fed into the Pointnet. The detection results are shown below in Table 4-9 together with the best performance Depth clustering can get.

Method	Car			Pedestrian			Cyclist		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Depth clustering + Pointnet	54.40	30.40	22.33	44.14	33.69	27.90	46.07	31.66	25.86
SqueezeSeg + Pointnet	42.32	36.72	33.47	9.36	7.54	6.38	19.62	14.98	14.46

Table 4-9: 3D object detection AP (%) on KITTI validation set of the Lidar-based detector

4-4 Experiments on Late Fusion

With all the parameters optimized in the former sections, the whole system combining the image-based detector and the Lidar-based detector together will be evaluated in this section. The parameters of the late fusion scheme will be optimized as well. The metric used here is the 3D object detection AP as well.

Results with Depth clustering as the region proposal

A rough idea should be acquired how each step of the late fusion influences the detection results. The detection results of different classes on the validation set are listed in Table 4-10(*Car*), Table 4-11(*Pedestrian*), Table 4-12(*Cyclist*). The parameters are explained in the following paragraphs.

Method	Multiple cluster settings	RGB filter	Lidar confidence	Camera confidence	IOU threshold	Easy	Moderate	Hard
Lidar only	9	-	-	-	-	54.40	30.40	22.33
Lidar only	9+30	-	-	-	-	53.50	35.56	22.08
Lidar only	9+30+50	-	-	-	-	52.86	34.94	21.62
Fusion	9	No	0.85	0.95	0.0	53.95	32.74	22.09
Fusion	9+30	No	0.85	0.95	0.0	48.14	38.91	27.92
Fusion	9+30	Yes	0.85	0.95	0.0	65.60	40.82	32.46
Fusion	9+30+50	Yes	0.85	0.95	0.0	**65.93**	42.96	33.80
Fusion	9+30+50	Yes	0.85	0.95	0.5	65.56	**50.01**	**40.77**

Table 4-10: 3D object detection AP (%) on KITTI validation set (Car)

Method	Multiple cluster settings	RGB filter	Lidar confidence	Camera confidence	IOU threshold	Easy	Moderate	Hard
Lidar only	9	-	-	-	-	44.14	33.69	27.90
Lidar only	9+30	-	-	-	-	43.72	31.96	26.27
Lidar only	9+30+50	-	-	-	-	31.47	23.39	19.10
Fusion	9	No	0.85	0.95	0.0	45.54	35.92	28.98
Fusion	9+30	No	0.85	0.95	0.0	46.18	34.78	28.96
Fusion	9+30	Yes	0.85	0.95	0.0	57.70	46.24	38.05
Fusion	9+30+50	Yes	0.85	0.95	0.0	**62.06**	46.45	38.33
Fusion	9+30+50	Yes	0.85	0.95	0.5	57.74	**48.63**	**40.35**

Table 4-11: 3D object detection AP (%) on KITTI validation set (Pedestrian)

Method	Multiple cluster settings	RGB filter	Lidar confidence	Camera confidence	IOU threshold	Easy	Moderate	Hard
Lidar only	9	-	-	-	-	46.07	31.66	25.86
Lidar only	9+30	-	-	-	-	23.44	16.76	14.28
Lidar only	9+30+50	-	-	-	-	20.95	15.77	14.32
Fusion	9	No	0.85	0.95	0.0	48.09	37.91	34.54
Fusion	9+30	No	0.85	0.95	0.0	33.34	23.75	20.77
Fusion	9+30	Yes	0.85	0.95	0.0	50.57	37.27	**35.87**
Fusion	9+30+50	Yes	0.85	0.95	0.0	53.45	38.67	34.64
Fusion	9+30+50	Yes	0.85	0.95	0.5	**55.38**	**39.58**	31.56

Table 4-12: 3D object detection AP (%) on KITTI validation set (Cyclist)

Applying multiple cluster settings means that the clusters come from different angle thresholds (θ) of the Depth clustering. This is quite necessary since setting a smaller angle threshold tends to get bigger clusters and setting a bigger angle threshold tends to get smaller clusters. If the clusters are too big, multiple objects could be segmented into a single cluster, which is hard to fix in a future step. If the clusters are small, a single object could be segmented into multiple clusters, which makes the Pointnet harder to recognize the object. A possible solution is to include multiple angle thresholds during the segmentation. However, this also causes some problems like that it brings more FP. Non-maximum suppression (NMS) can solve part of these duplicated detection results, but there are some irrelevant detection results from the multiple cluster settings that are hard to fix. That is the reason why adding multiple clustering settings will make the Lidar-only detection accuracy lower, which is especially obvious in the performance on *Pedestrian* (Table 4-11) and *Cyclist* (Table 4-12).

The IOU threshold is a parameter used when finding the correspondence as illustrated in Section 3-3-1. The IOU between the corresponding objects should have a bigger value than the IOU threshold to make the correspondence convincing and reliable.

RGB filter is a method to filter the detection results from Lidar detector. If RGB filter is applied, those Lidar-based detection results that do not have the corresponding image-based detection results will be given up. This can remove a lot of FP coming from the Lidar-based detector.

Lidar confidence and camera confidence are the parameters while applying the Dempster-Shafer theory. The confidence score is a measure of the reliability of the detection results

from a single sensor. A higher confidence score means the results coming from this single sensor is more reliable.

From all the three tables, the results show that adding RGB filter and multiple cluster settings at the same time will largely improve the detection accuracy.

After figuring about the effect of each step, the numerical parameters are fixed. First, the two confidence scores are set according to the accuracy of the single detector. Finally, the Lidar confidence is set to be 0.85 and the camera confidence is set to be 0.95. The final IOU threshold chosen for the correspondence is 0.5 according to the KITTI settings in the object detection benchmark.

Results with SqueezeSeg as the region proposal

SqueezeSeg is used to enhance the segmentation performance. The detection results with SqueezeSeg are shown in Table 4-13. Results from three different segmentation algorithms are all listed in the table. The third method is to combine all the clusters from both the Depth-clustering and SqueezeSeg.

Segmentation Method	Car			Pedestrian			Cyclist		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Depth-clustering	65.56	50.01	40.77	57.74	48.63	40.35	55.38	39.58	31.56
SqueezeSeg	62.24	55.77	48.09	23.41	23.07	17.11	38.18	31.13	27.62
Depth-clustering + SqueezeSeg	70.87	62.09	52.81	64.27	48.46	40.47	62.38	47.58	38.64

Table 4-13: 3D object detection AP (%) on KITTI validation set of the fusion system

4-5 Comparison

After the optimization of the former sections, the best result is got for the KITTI validation set. The optimized algorithm is to combine the clusters from both the Depth clustering and the SqueezeSeg. The final multiple cluster angle thresholds used for the Depth clustering are $9°, 30°, 50°$. RGB filter is applied. Lidar confidence is set to be 0.85 and the camera confidence is set to be 0.95. The IOU threshold is set to be 0.5.

4-5-1 KITTI validation set

There are a few top-ranking works in the KITTI 3D object detection benchmark like F-Pointnet [10], MV3D [3], VeloFCN [49], etc. applying a same training and validation split as described in Section 4-1. The comparisons to these works on the validation set are shown in Table 4-14 (Car). Only the comparisons on the *Car* category are shown below since most of these works only publish the results on *Car*.

From the comparisons on the validation set, the proposed late fusion system still has a gap comparing to the results of the state-of-the-art F-Pointnet [10]. However, the proposed system provides more redundancy comparing to these works, since it can provide some intermediate results from the image-based detector and the Lidar-based detector.

Method	Easy	Moderate	Hard
F-Pointnet [10]	**83.76**	**70.92**	**63.65**
MV3D [3]	71.29	62.68	56.56
VeloFCN [49]	15.20	13.66	15.98
Thesis (Fusion)	70.87	62.09	52.81

Table 4-14: 3D object detection AP (%) on KITTI validation set (Car only)

4-5-2 KITTI test set

Method	Car			Pedestrian			Cyclist		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
F-PointNet [10]	81.20	70.39	62.19	**51.21**	**44.89**	40.23	**71.96**	**56.77**	**50.39**
AVOD-FPN [40]	**81.94**	**71.88**	**66.38**	50.80	42.81	**40.88**	64.00	52.18	46.61
VoxelNet [9]	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37
MV3D [3]	71.09	62.35	55.12	-	-	-	-	-	-
F-PC_CNN [63]	60.06	48.07	45.22	-	-	-	-	-	-
Thesis	65.54	58.48	46.54	35.85	31.65	26.94	50.58	35.39	33.55

Table 4-15: 3D object detection AP (%) on KITTI test set (accessed August 3, 2018)

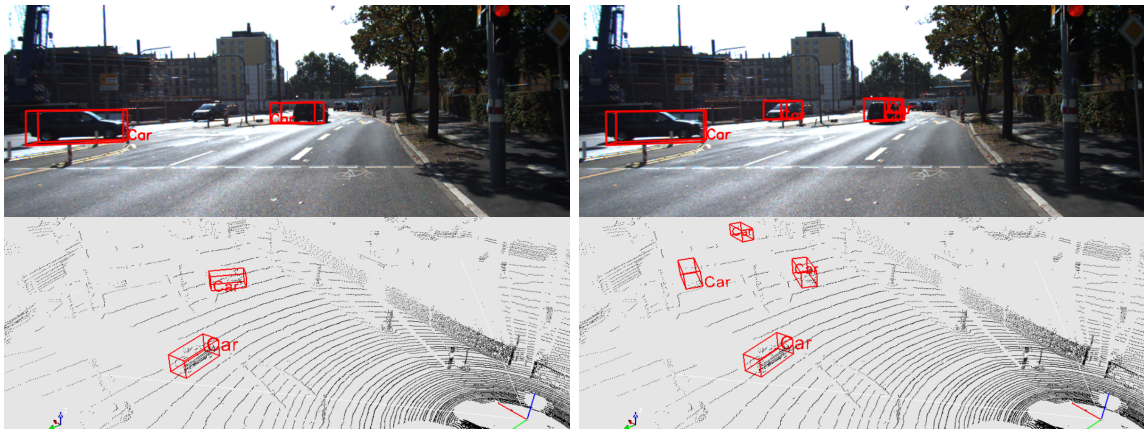
The detection results on the KITTI test set are shown in Table 4-15. For the *Car* detection, the results are better than the Pedestrian and Cyclist. The cause can be that KITTI provides a lot more *Car* instances in the training data than the *Pedestrian* and *Cyclist*. The proposed detection system ranks 17-th among all the submitted results and ranks 5-th among all the published works for *Car* class on the KITTI 3D object detection benchmark (accessed August 3, 2018).

4-6 Failure Analysis

There are a few typical failure modes that the proposed system might encounter.

4-6-1 Sparse points inside an object

For those objects far away, there will be sparse points reflected, which makes it difficult to be segmented from the whole point cloud and also makes the classification and the bounding box regression pretty hard.



(a) Detection results **(b)** Ground truth

Figure 4-15: Failure because of sparse points

As can be seen in the Figure 4-15, the two cars far away that really have few points inside are not detected. There is also a car with sparse points whose detected bounding box is not correct.

4-6-2 Occlusion

Another commonly seen failure mode is that the object is occluded by other things.

(a) Detection results **(b)** Ground truth

Figure 4-16: Failure because of the occlusion

As can be seen in Figure 4-16, the person behind the wall is failed to be detected and the bounding box of the car behind the wall is wrongly estimated as well.

Occlusion sometimes will make the object have few points, which makes the segmentation and classification both harder.

4-6-3 Close objects from the same class

When cars parking closely by the road, the detection might fail as well.

(a) Detection results **(b)** Ground truth

Figure 4-17: Failure when objects from the same class are really close

As can be seen in Figure 4-17, there are a few cars being missed in the left side since they are really close. There are also two cars being detected as a single car.

The close objects will make the segmentation difficult. It is not easy to separate the close objects. If the segmentation fails, the classification and bounding box regression will fail as well.

Chapter 5

Conclusion

5-1 Conclusion

As mentioned in Section 1-3, the motivation of the thesis work is to propose a 3D object detection system by using the sensor data from the Lidar and camera for the autonomous vehicle. The two main research questions of the thesis are: **What is the best solution to build a Lidar-based 3D object detector?** and **What is the best solution to fuse the data from Lidar and camera for 3D object detection?**

5-1-1 Lidar-based object detection

There are several challenges in the 3D road user detection using Lidar. The first challenge is that the points from the background are dominant in the point cloud, which makes the ROI proposal difficult in the point cloud. The second challenge is that the scanned object point cloud is incomplete since Lidar only sees one side of the object. The third challenge is that it is not easy to deal with the scattered, unordered Lidar point cloud. The fourth challenge is that the resolution in the far range is low and few points are reflected by the objects far away, which makes the detection range of the Lidar limited.

In this thesis, two point cloud segmentation algorithms are applied. The Depth clustering brings not only the target objects but also the objects that are not concerned about in this thesis context. Although this becomes a big challenge for the later classifier to judge whether the clusters belong to the target objects in the thesis context, it is necessary for a future autonoumous vehicle to be able to detection all kinds of objects. The Depth clustering used in the thesis can still be applied if more classes have to be taken into consideration. The SqueezeSeg combined with DBSCAN is a deep-learning based segmentation algorithm. The final detection performance with SqueezeSeg might be better if combining the SqueezeSeg network with Pointnet into an end-to-end network.

The Pointnet is one of the state-of-the-art networks which are able to parse the point cloud directly. In their original work [2], the dataset used provides the full point cloud of an object.

In the thesis results, it turns out that modified Pointnet (Section 3-2-3) is able to estimate the class and the amodal 3D bounding box even with part of the object points since the object point cloud provided by Lidar is incomplete. However, in order to improve the accuracy of the Pointnet, the points of an object should be able to represent the object well, which means that the number of points should be enough and the points should cover as much the object surface area as possible. What's more, Pointnet requires the point cloud of different objects having the same number of points. In this thesis, the upsampling is implemented by the random duplication and the downsampling is implemented by the random pickup. This works for the current research, but in the future work, a better resampling method can be explored (Section 5-2-2).

5-1-2 Late fusion

The first step of the late fusion is to find the correspondence between detection results from the Lidar and camera. The information used in the thesis to find the correspondence is the Intersection Over Union (IOU) between the bounding boxes of the two detection results in the image plane. It works most of the time but sometimes fails when the object is occluded by the other things. After getting the corresponding results, the classification vectors from the two different sensors are combined by Dempster-Shafer theory, which provides a more convincing probability.

There are some detection results that have no corresponding detection results that cannot go through the late fusion scheme. A possible solution is to include multiple cameras to provide a wider field of view. This is feasible since a camera costs much less than a Lidar.

5-1-3 Detection results analysis

The 3D object detection system proposed in the thesis work is getting competitive results with the former state-of-the-art works like MV3D [3], etc. The designed system is good at the detection the objects in the close range which have enough points. What's more, the Lidar-based detector and image-based detector work separately, which provides the system redundancy in case of a single sensor failure. The Lidar-based detector also compensates the shortcoming of the camera-based object detection that camera suffers a lot from some certain weather conditions and illumination changes.

However, there are some typical failure patterns as described in Section 4-6. They are sparse points inside an object, occlusion and close objects from the same class. These three failure modes mainly challenges the Lidar-based detection system.

5-2 Recommendations

5-2-1 Point cloud segmentation

Since the segmentation of the point cloud is the first step of the Lidar-based detection system, it is quite necessary to improve the segmentation results. The segmentation results from the two algorithms used in the thesis work (Depth clustering and SqueezeSeg) are still not so

good as expected. Future works can be done to explore a better segmentation algorithm. Recent works like SPLATNet [64] that segments the original point cloud are also possible to be applied in the pipeline. SPLATNet allows flexible specifications of the lattice structure enabling hierarchical and spatially-aware feature learning [64].

5-2-2 Object point cloud resampling

The resampling method proposed in the thesis work can be improved. The upsampling can be implemented in the network as PU-Net [60] which get a rather good reconstruction of the object point cloud. PU-Net learns multilevel features per point and expand the point set via a multibranch convolution unit implicitly in feature space [60].

5-2-3 Different classification model

Based on Pointnet [2], many works improve the architecture of the network. Pointnet++ [26] is an improved work by the authors of Pointnet, which extracts the deep hierarchical feature from the point cloud. It achieves better performance and generalizability in complex scenes and is able to deal with non-uniform sampling density [26]. DGCNN builds a different classification model which incorporates local neighborhood information [65].

5-2-4 Bounding box fusion

In the thesis context, the fusion of 2D and 3D bounding boxes is not implemented. 2D bounding box from the image-based detector tends to provide a more convincing boundary in the RGB image but 3D bounding box from the Lidar-based detector can provide more information of the 3D position and object size. A fuzzy logic model could be applied to combine different information from the 2D bounding box and the 3D bounding box, which might provide a better 3D detection result.

5-2-5 Dataset

As mentioned in Section 4-3-1, the class distributions of the KITTI dataset and the Depth clustering segmentation result are different. In Section 4-3-1, random percentage of the *Miscs* are collected as the extra training data. However, it might be better to first feed the *Miscs* into the Pointnet trained with the ground-truth objects and choose those *Miscs* that are wrongly classified as the extra training data. This will make the extra *Miscs* more meaningful and help improve the classification accuracy.

On the other hand, the Neural Network (NN) methods for the road user detection are heavily data-driven. For now, the KITTI [1] is almost the only open source dataset available in the context of 3D object detection in the road scene. Only 8 classes are labelled in KITTI and only the *Car* have enough number of instances. Bigger dataset should be collected with more classes in the future.

Appendix A

Visualization of Detection Results

Bounding box color and class correspondence: Car, Van, Truck, Pedestrian, Person_sitting, Cyclist, Tram, Misc.

A-1 Image-based Detection Results

Figure A-1: Visualization of image-based detection results

A-2 Lidar-camera Fusion Detection Results

Figure A-2: Visualization of Lidar-based detection results

Appendix B

Theories

B-1 Image Features

B-1-1 Histogram of Oriented Gradients (HOG)

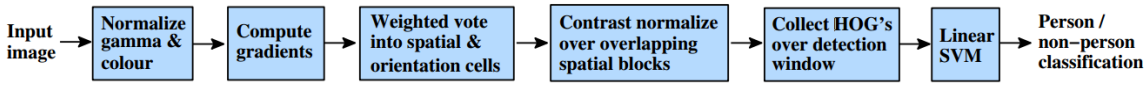


Figure B-1: HOG feature extraction scheme[16]

Dalal and Triggs create the HOG, which is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid [16]. The image is first divided into small spatial regions ('*cells*'). The feature extraction process is shown in Figure B-1. Color normalization is applied to the image to make the descriptor invariant to the color. Then Gaussian smoothing is applied to the image and the gradients are calculated by applying a 1D filtering mask $[-1, 0, 1]$ to the image. Each pixel calculates a weighted vote for an edge orientation histogram channel based on the orientation of the gradient element centred on it, and the votes are accumulated into orientation bins over the '*cells*' [16]. After getting the histograms of oriented gradients, local contrast normalization is applied to avoid the effect of the illumination and foreground-background contrast. After all the '*cells*' are normalized, the feature vector of all the regions are concatenated into a single one, which is the feature vector of the image.

The basic idea of HOG is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions [16].

B-1-2 Deformable Part Model (DPM)

After the success of HOG, Felzenszwalb et al. (2010) [18] use HOG as a building block of their DPM feature. They find that there are still a lot of problems that HOG cannot solve,

such as the occlusion, deformation, photometric variation, viewpoint variation, intra-class variability, etc. They are inspired by the pictorial structures [66], which model the object as a constellation of parts as shown in Figure B-2. So the DPM captures local appearance properties of an object while the deformable configuration is characterized by spring-like connections between certain pairs of parts [18].

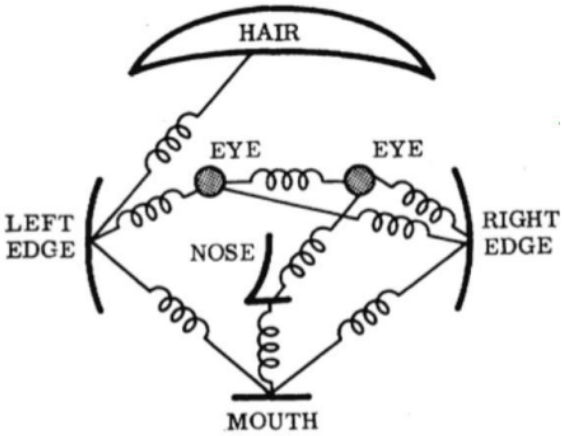


Figure B-2: Pictorial structure [17]

In Dalal's work, only one filter is applied to the whole object. Felzenszwalb et al. (2010) [18] enrich the Dalal-Triggs model using a star-structured part-based model defined by a 'root' filter (analogous to the Dalal-Triggs filter) plus a set of part filters and deformation models [18] as shown in up-right corner of Figure B-3. The part filters capture features at twice the spatial resolution relative to the features captured by the root filter [18] in order to model visual appearance at multiple scales. In the final step, a latent SVM is trained. In the latent SVM, each example x is scored according to the following function [18]:

$$f_\beta(x) = \max_{z \in Z(x)} \beta \cdot \Phi(x, z) \tag{B-1}$$

β is the concatenation of the root filter, the part filters and deformation cost weights, z is a specification of the object configuration, and $\Phi(x, z)$ is a concatenation of sub-windows from a feature pyramid and part deformation features [18]. A score function is defined for each hypothesis. It is composed of three parts: the global object model (from the root filter), the object part model (from the part filters) and the deformation model. The matching process is shown in Figure B-3. A root location where the score is high corresponds to the detection result. According to the connection between the score function and the latent Support Vector Machine (SVM) score function B-1,the model parameters are learned with the latent SVM framework.

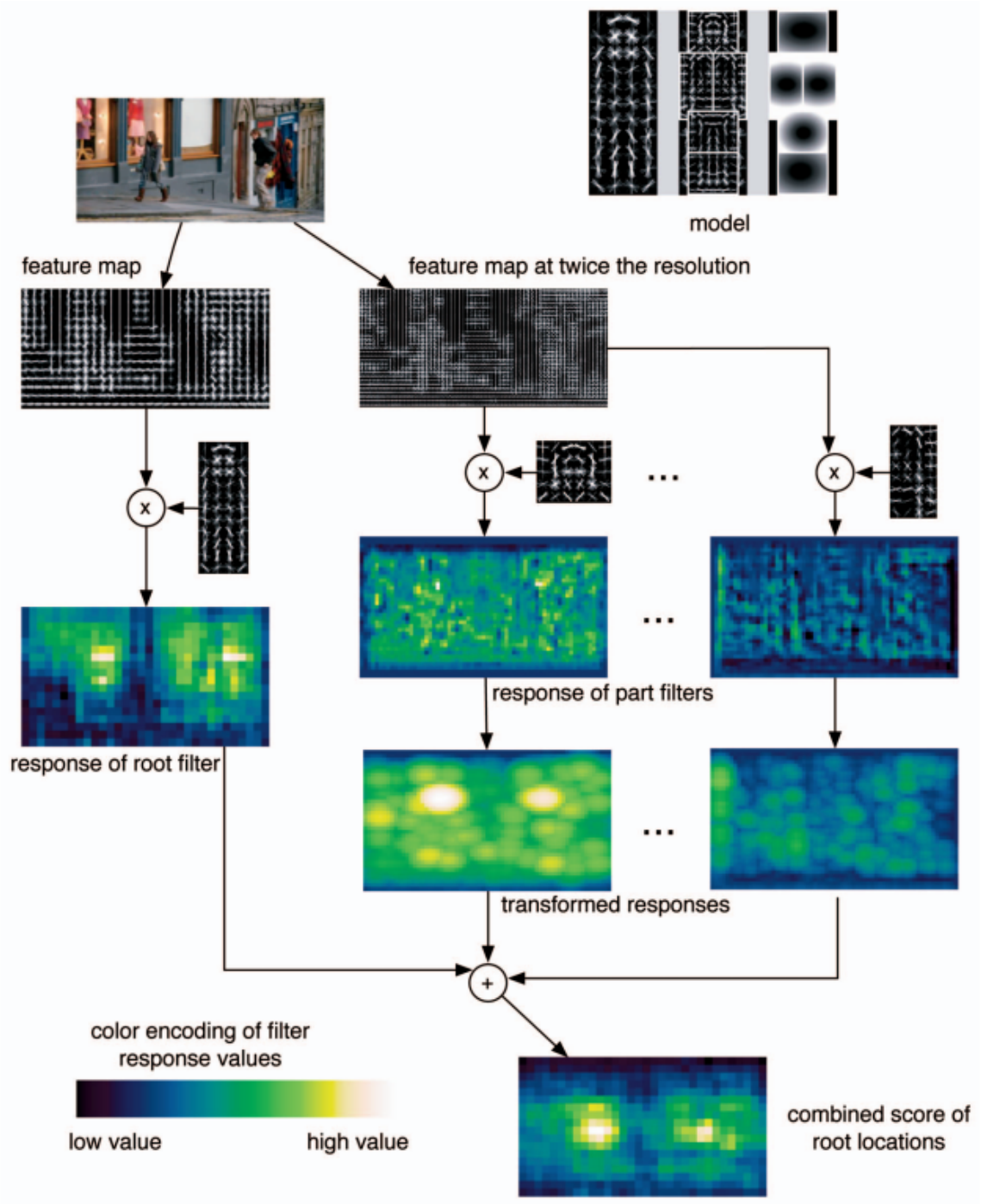


Figure B-3: Matching process of DPM [18]

Bibliography

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3354–3361, IEEE, 2012.

[2] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, vol. 1, no. 2, p. 4, 2017.

[3] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *IEEE CVPR*, vol. 1, p. 3, 2017.

[4] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[7] S. Song and J. Xiao, "Sliding shapes for 3d object detection in depth images," in *European conference on computer vision*, pp. 634–651, Springer, 2014.

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

[9] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," *arXiv preprint arXiv:1711.06396*, 2017.

[10] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," *arXiv preprint arXiv:1711.08488*, 2017.

[11] G. Zhao, X. Xiao, and J. Yuan, "Fusion of velodyne and camera data for scene parsing," in *Information Fusion (FUSION), 2012 15th International Conference on*, pp. 1172–1179, IEEE, 2012.

[12] T.-D. Vu, O. Aycard, and F. Tango, "Object perception for intelligent vehicle applications: A multi-sensor fusion approach," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pp. 774–780, IEEE, 2014.

[13] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, "Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving," *arXiv preprint arXiv:1612.01051*, 2016.

[14] I. Bogoslavskyi and C. Stachniss, "Efficient online segmentation for sparse 3d laser scans," *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, vol. 85, no. 1, pp. 41–52, 2017.

[15] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," *arXiv preprint arXiv:1710.07368*, 2017.

[16] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.

[17] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Transactions on computers*, vol. 100, no. 1, pp. 67–92, 1973.

[18] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[19] C. Thorpe, M. Herbert, T. Kanade, and S. Shafter, "Toward autonomous driving: the cmu navlab. ii. architecture and systems," *IEEE expert*, vol. 6, no. 4, pp. 44–52, 1991.

[20] T. Luettel, M. Himmelsbach, and H.-J. Wuensche, "Autonomous ground vehiclesconcepts and a path to the future," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1831–1839, 2012.

[21] Wikipedia contributors, "Camera — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Camera&oldid=838183109, 2018. [Online; accessed 30-April-2018].

[22] R. Girshick, "Fast r-cnn," *arXiv preprint arXiv:1504.08083*, 2015.

[23] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, pp. 2980–2988, IEEE, 2017.

[24] J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with cuda," in *ACM SIGGRAPH 2008 classes*, p. 16, ACM, 2008.

[25] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," *arXiv preprint arXiv:1608.07916*, 2016.

[26] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, pp. 5105–5114, 2017.

[27] F. Garcia, A. De la Escalera, and J. M. Armingol, "Joint probabilistic data association fusion approach for pedestrian detection," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pp. 1344–1349, IEEE, 2013.

[28] J. Wang, Z. Wei, T. Zhang, and W. Zeng, "Deeply-fused nets," *arXiv preprint arXiv:1605.07716*, 2016.

[29] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *null*, p. 734, IEEE, 2003.

[30] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[31] D. Park, D. Ramanan, and C. Fowlkes, "Multiresolution models for object detection," in *European conference on computer vision*, pp. 241–254, Springer, 2010.

[32] J. Yan, X. Zhang, Z. Lei, S. Liao, and S. Z. Li, "Robust multi-resolution pedestrian detection in traffic scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3033–3040, 2013.

[33] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[34] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International journal of computer vision*, vol. 59, no. 2, pp. 167–181, 2004.

[35] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[36] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[37] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[38] C. Premebida, J. Carreira, J. Batista, and U. Nunes, "Pedestrian detection combining rgb and dense lidar data," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 4112–4117, IEEE, 2014.

[39] A. González, D. Vázquez, A. M. Lóopez, and J. Amores, "On-board object detection: Multicue, multimodal, and multiview random forest of local experts," *IEEE transactions on cybernetics*, 2017.

[40] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, "Joint 3d proposal generation and object detection from view aggregation," *arXiv preprint arXiv:1712.02294*, 2017.

[41] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pp. 127–136, IEEE, 2011.

[42] D. Z. Wang and I. Posner, "Voting for voting in online point cloud object detection.," in *Robotics: Science and Systems*, 2015.

[43] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks," *arXiv preprint arXiv:1609.06666*, 2016.

[44] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," *arXiv preprint arXiv:1612.03144*, 2016.

[45] O. H. Jafari, D. Mitzel, and B. Leibe, "Real-time rgb-d based people detection and tracking for mobile robots and head-worn cameras," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 5636–5643, IEEE, 2014.

[46] R. Schnabel, R. Wahl, and R. Klein, "Efficient ransac for point-cloud shape detection," in *Computer graphics forum*, vol. 26, pp. 214–226, Wiley Online Library, 2007.

[47] G. Shafer, *A mathematical theory of evidence*, vol. 42. Princeton university press, 1976.

[48] R. O. Chavez-Garcia, T.-D. Vu, O. Aycard, and F. Tango, "Fusion framework for moving-object classification," in *Information Fusion (FUSION), 2013 16th International Conference on*, pp. 1159–1166, IEEE, 2013.

[49] B. Li, "3d fully convolutional network for vehicle detection in point cloud," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 1513–1518, IEEE, 2017.

[50] Y. Choe, S. Ahn, and M. J. Chung, "Fast point cloud segmentation for an intelligent vehicle using sweeping 2d laser scanners," in *Ubiquitous Robots and Ambient Intelligence (URAI), 2012 9th International Conference on*, pp. 38–43, IEEE, 2012.

[51] B. Douillard, J. Underwood, V. Vlaskine, A. Quadros, and S. Singh, "A pipeline for the segmentation and classification of 3d point clouds," in *Experimental robotics*, pp. 585–600, Springer, 2014.

[52] J. Behley, V. Steinhage, and A. B. Cremers, "Laser-based segment classification using a mixture of bag-of-words," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 4195–4200, IEEE, 2013.

[53] M. Himmelsbach, F. V. Hundelshausen, and H.-J. Wuensche, "Fast segmentation of 3d point clouds for ground vehicles," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pp. 560–565, IEEE, 2010.

[54] F. Moosmann, *Interlacing self-localization, moving object tracking and mapping for 3d range sensors*, vol. 24. KIT Scientific Publishing, 2013.

[55] A. Savitzky and M. J. Golay, "Smoothing and differentiation of data by simplified least squares procedures.," *Analytical chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964.

[56] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2018.

[57] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 1529–1537, 2015.

[58] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[59] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

[60] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-net: Point cloud upsampling network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2790–2799, 2018.

[61] D. Ruta and B. Gabrys, "An overview of classifier fusion methods," *Computing and Information systems*, vol. 7, no. 1, pp. 1–10, 2000.

[62] P. Smets and R. Kruse, "The transferable belief model for belief representation," in *Uncertainty Management in Information Systems*, pp. 343–368, Springer, 1997.

[63] X. Du, M. H. Ang Jr, S. Karaman, and D. Rus, "A general pipeline for 3d detection of vehicles," *arXiv preprint arXiv:1803.00387*, 2018.

[64] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz, "Splatnet: Sparse lattice networks for point cloud processing," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2530–2539, 2018.

[65] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *arXiv preprint arXiv:1801.07829*, 2018.

[66] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *International journal of computer vision*, vol. 61, no. 1, pp. 55–79, 2005.

Glossary

List of Acronyms

ADAS Advanced Driver-Assistance Systems

LKA Lane Keeping Assistance

ACC Adaptive Cruise Control

VRU Vulnerable Road Users

CV Computer Vision

SSD Single-Shot-Detector

GPU Graphics Processing Unit

NN Neural Network

R-CNN Region-based Convolutional Neural Networks

FPS Frames Per Second

YOLO You Only Look Once

IOU Intersection Over Union

NMS Non-maximum suppression

ROI Region of Interests

MLP Multi-layer Perceptron

AP Average Precision

TP True Positives

FP False Positives

FN False Negatives

SGD Stochastic Gradient Descent

CNN Convolutional Neural Networks

HOG Histogram of Oriented Gradients

DPM Deformable Part Model

FC Fully Connected Layer

RPN Region Proposal Network

TSDF Truncated Signed Distance Function

SVM Support Vector Machine

VFE Voxel Feature Encoding

FPN Feature Pyramid Networks

LBP Local Binary Pattern

RANSAC Random Sample Consensus

ML Machine Learning

MLS Moving Least Squares

DS Dempster-Shafer

BBA Basic Belief Assignment

CRF Conditional Random Field

RNN Recurrent Neural Network

DBSCAN Density-Based Spatial Clustering of Applications with Noise

List of Symbols

α	Ground removal angle in depth clustering
β	Clustering angle in depth clustering
θ	Threshold for the clustering angle