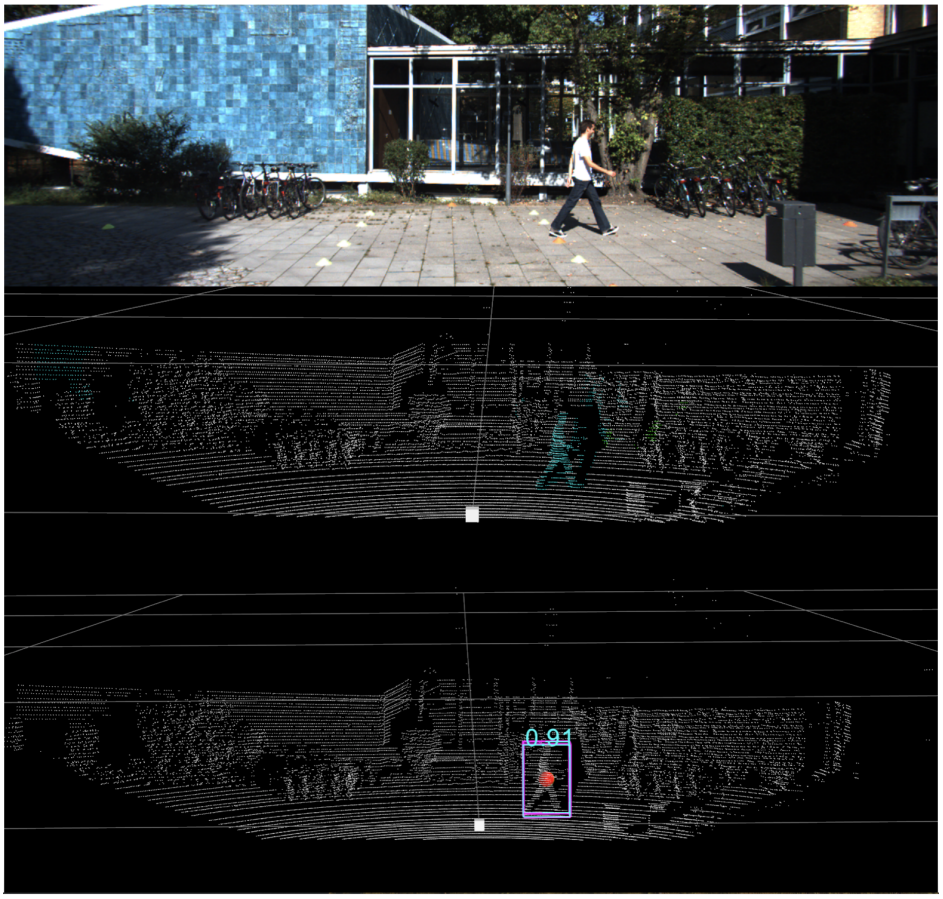
CHALMERS
UNIVERSITY OF TECHNOLOGY



Object detection using deep learning and camera-lidar fusion methods

Master's thesis in Electrical Engineering

YAXI XIE
YULING ZHANG

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021

Object detection using deep learning and camera-lidar fusion methods

YAXI XIE
YULING ZHANG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2021

Object detection using deep learning and camera-lidar fusion methods
YAXI XIE
YULING ZHANG

Supervisor: Lars Hammarstrand, Department of Electrical Engineering
Supervisor: Karl-Magnus Dahlén, Xymbiotec System AB
Supervisor: Christopher Lindberg, Xymbiotec System AB
Supervisor: Alexander Laas, Xymbiotec System AB
Examiner: Lars Hammarstrand, Department of Electrical Engineering

Master's Thesis 2021
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Object detection using deep learning and camera-lidar fusion methods
YAXI XIE
YULING ZHANG
Master's thesis in Communication Engineering
Chalmers University of Technology

Abstract

In order for self-driving cars to make more reasonable decisions regarding safe path to navigate, it needs to rely on a better understanding of the surrounding environment. The perception collects measurements from the environment using sensors, such as cameras, lidars and radars. Based on the collected data, the location and class of objects are predicted by object detection algorithms.

There are different detection methods based on single or multiple sensors. To fully use the characteristics from various sensors, combining the information from multiple sensors may be a potential prospect, so-called sensor fusion. Sensor fusion can achieve more reliable results by the complementary information from different sensors. It is certified that trained deep neural networks have the strength to achieve accurate object detection. The main objective of this thesis is to investigate how the lidar-only deep model CenterPoint can be improved by also considering camera information. One of the common ways to extract object classification from a camera is semantic segmentation, partitioning the pixels with semantic labels. The semantic segmentation scores for relevant objects should help object detection. Hence, this thesis focuses on the following research questions: 1) Can the CenterPoint algorithm be improved by including semantic information from a camera? If so, by how much? 2) Are there situations where fusing with the segmentation information degrade the result? 3) What are the reasons causing the differences?

We propose a fusion strategy called Painted CenterPoint, inspired by the PointPainting fusion algorithm. After projecting lidar point clouds on images, points are painted with the corresponding segmentation scores. Then employ the CenterPoint on painted point clouds to achieve the final detection results. The segmentation methods would differ how PointPainting benefits the lidar detector in different metrics and scenarios, so we introduce three segmentation methods: DeepLabV3, DeepLabV3+ and Hierarchical Multi-scale Attention (HMA). We train the fusion models on a simple KITTI training set to detect and classify Vehicle, Pedestrian and Cyclist classes. Then the model is evaluated on KITTI metrics.

The final results indicate that CenterPoint can be improved by the "paint" strategy. In conclusion, Painted CenterPoint with DeepLabV3 gives the most unstable results. And Painted CenterPoint with HMA gives the highest improvements and precision on KITTI. The performance of DeepLabV3+ is somewhere in between.

Keywords: Deep Learning, Image Segmentation, Object Detection, Sensor Fusion.

Acknowledgements

First of all, we feel appreciated the help and valuable suggestions given by Professor Lars Hammarstrand from Chalmers. Thanks to his patience and kindness, we received a lot of meaningful guide and advice. He gave us insightful comments, helping us solve the problems and go on the right track.

We would also express our gratitude to our supervisors Karl-Magnus Dahlén, Christopher Lindberg and Alexander Laas from Xymbiotec System AB company. They gave us a great chance to research this state-of-the-art topic and support us the high-performance hardware with GPU in this thesis's project, which greatly decreased our training time. They always encouraged us and gave us opportune feedbacks. This thesis would not complete smoothly without their supports. We do feel motivated and confident when we were doing this thesis with their help.

Last but not least, we would like to thank our opponents, Lingyun Deng and Xi Chen. They gave us many valuable comments for our thesis. None mentioned none forgot.

Yaxi Xie & Yuling Zhang, Gothenburg, October 2021

Contents

List of Figures

List of Tables

Abbreviations

AD Autonomous Driving

ADAS Advanced Driver Assistance Systems

ANNs Artificial Neural Networks

AOS Average Orientation Similarity

AP Average Precision

APH Average Precision Weighted by Heading

ASPP Atrous Spatial Pyramid Pooling

AUC Area Under Curve

b-box Bounding Box

BEV Bird's Eye View

CaDNN Categorical Depth Distribution Network

CNNs Convolutional Neural Networks

CP CenterPoint

DCNNs Deep Convolutional Neural Networks

DNNs Deep Neural Networks

DLA Deep Layer Aggregation

Fast R-CNN Fast Region with CNN features

FPN Feature Pyramid Networks

HMA Hierarchical Multi-scale Attention

HOG Histogram of Oriented Gradients

IOU Intersection Over Union

mAP mean Average Precision

NDS NuScenes Detection Score

NNs Neural Networks

NMS Non Maximum Suppression

P-R Precision-Recall

PCP Painted CenterPoint

PVRCNN Point Voxel RCNN

ROI Region Of Interest

RPN Region Proposal Network

ResNets Residual Networks

SECOND Sparsely Embedded Convolutional Detection

SOTA State-Of-The-Art

STD Sparse-to-Dense

std standard deviation

V3 DeepLabV3

V3+ DeepLabV3+

YOLO You Only Look Once

1

Introduction

1.1 Background

The perception system plays a vital role in Autonomous Driving (AD) and Advanced Driver Assistance Systems (ADAS) and it generally uses various types of sensors to collect data. Typical sensors are cameras, automatic radars, and lidar. Object detection tries to determine the class and location of the objects in the surrounding environment based on the measurements from these sensors. Since the measurements are often with lots of noise, the performance of the object detection algorithm used becomes extremely important for the decision-making system to make a reasonable and safe decision on the predicted path.

The measurements from lidar or radars are in form of 3D point clouds that is a set of data points generated by scanners. When scanners shoot lasers onto the surface of objects, the surface reflects lasers and sends them back to scanners. The points show the positions of objects reflection like part of vehicles, people or trees. These points compose the point clouds. To help perception systems understand the environment, in general, point clouds are clustered and each cluster represents measurements of one object. Clustering methods are often heuristic methods that need to be hand-crafted to handle many different classes of objects, which may be troublesome and not that practical. Recent studies have proved that a better way is to train a deep learning detector that can learn to localize and possibly classify the objects based on point clouds. With the development of deep learning, the heuristic clustering methods are gradually replaced by deep learning methods.

While it is obvious that the information provided by a single type of sensor is finite so that the detection precision is limited as well. If more types of sensors are added to the detection system providing additional external information, the detection accuracy can be further improved consequently. To effectively combine the information from different sensors, sensor fusion methods can be used. The complementary properties between different sensors may benefit the object detection systems and enhance the precision of the detection results.

Since cameras and lidar are the most commonly used sensors in object detection, this thesis then focuses on the detection performance of these two sensors. The purpose of this study is to explore how to fuse the information from cameras and

lidar to obtain the best performance of clustered pixels and detection. Therefore, a fusion strategy is built based on deep neural networks to realize object detection. The final results show detected objects with the corresponding classes, locations, and confident scores.

1.2 Motivation and Purpose

According to the benchmarks provided by KITTI [1] and nuScenes [2], many lidar-based object detection algorithms are at higher ranking places, which is surprisingly even better than camera-lidar fusion methods, that considering both lidar and cameras information. Does this indicate that lidar are perfectly enough to complete object detection tasks? The answer should certainly be no. Although point clouds provide a relatively accurate range view and depth information, they lack shape and texture information. Moreover, even if they can provide pixel-level texture and structure information, but cameras always struggle with depth ambiguity. Therefore, sensor fusion is still a meaningful task to effectively utilize the advantages from these two sensors and learn from each other to offset their drawbacks [3].

There are many popular and high-performance lidar-based detectors. In this thesis, the designed fusion strategy is based on a State-Of-The-Art (SOTA) object detection method CenterPoint which only uses lidar point clouds [4]. It is an excellent method based on a novel idea of centre-based representation. There are 3 out of the top 4 entries in the recent NeurIPS 2020 nuScenes 3D Detection challenge [5] applied this algorithm. Since this algorithm has caused new fashion and trends in the field of object detection, it is worth studying if introducing cameras information would further improve the precision. As mentioned, this thesis focuses on camera-lidar fusion. Semantic segmentation extracts the image information for relevant objects at pixel level, where each pixel on images is labelled with its corresponding semantic label, e.g., car, road, building, etc. This is a common way to capture object classes from camera images and provides camera information to fusion strategy in a simple and effective way. To sum up, here lists the main study objectives and questions in this thesis:

1. Can the lidar-only CenterPoint algorithm be improved by including semantic information from camera images? If so, by how much?

2. Are there situations where fusing with the image segmentation information degrade the result?

3. What are the reasons causing the difference?

1.3 Challenges and limitations

There have been many studies with respect to the fusion of cameras and lidar in recent years, but the results are not satisfying. One of the challenges is synchronization. For example, the lidar and cameras used on the nuScenes capture the

information at different frequencies, which requires extra attention [3].

Another challenge is viewpoint misalignment. It is known that most SOTA methods, such as Sparse-to-Dense (STD) [6] and PointPainting [3], both use convolution in the Bird's Eye View (BEV). This is because of the advantages in BEV including minimal occlusions and lack of scale ambiguity. Additionally, it does not suffer the depth-blurring effect, which usually occurs when applying 2D convolution in the other 2D views. So lidar-based detection methods are often in BEV while camera-based methods are in the camera view. When applying the camera-lidar fusion method, then the question is how to precisely match the information from these two different views. If convolution is still used in BEV to obtain the best use of lidar point clouds as mentioned, camera images should therefore be converted to the same view as point clouds. However, it is difficult to get camera images in BEV or convert the images to BEV. So the viewpoint misalignment is now a big challenge when using the camera-lidar fusion method in object detection [3].

Therefore, how to effectively deal with the above two problems would directly affect the detection results. Due to the limited time and resources in this thesis, we can only realize and train a reasonable model on the simple KITTI dataset. Our purpose is to study if and how camera information can benefit the CenterPoint algorithm and to show the potential of sensor fusion object detection methods in some scenarios, instead of providing a model comparable to the benchmarks. Then, the final achieved results may not be with as high precision and capability as ranking on benchmarks.

1.4 Contributions and main results

This thesis proposes a deep learning fusion strategy called Painted CenterPoint inspired by the PointPainting fusion algorithm [3]. After projecting lidar point clouds on images, points are painted with the corresponding segmentation scores. Then the CenterPoint [4] works as a detector in the fusion strategy on the painted point clouds to achieve final detection results. The segmentation methods would differ how PointPainting benefits the CenterPoint detector in different metrics and scenarios, so we introduce three segmentation methods: DeepLabV3 [7], DeepLabV3+ [8] and Hierarchical Multi-scale Attention (HMA) [9]. The fusion models are trained on a KITTI training set to detect and classify three classes: Vehicle, Pedestrian and Cyclist. Finally, the model is evaluated based on KITTI evaluation metrics.

The final results show successfully improved performance of the SOTA object detector: CenterPoint in Vehicle, Pedestrian and Cyclist classes and show the potential of sensor fusion in improving the detection precisions. We also found that the performance may also deteriorate after fusing and that semantic segmentation methods have different degrees of accuracy improvements. We study and discuss the reasons causing the difference in performance based on the three segmentation methods and give the outstanding strategy. Overall, the main contributions are:

1. Designed and realized a deep learning sensor fusion object detection strategy

based on PointPainting.

2. Successfully improved the performance of SOTA object detector: CenterPoint in Vehicle, Pedestrian and Cyclist classes.

3. Studied and discussed the reason for different fusion performances based on 3 segmentation methods.

4. Studied and discussed the key characteristics causing different performances in various scenarios.

1.5 Outline

The thesis is structured as follows. Chapter 2 covers the description of the necessary background to understand the method and idea used in the project. We introduce the idea of neural networks, object detection and segmentation. We show the related work of some existing object detection methods based on cameras and lidar. In Chapter 3, we describe the fusion method PointPainting, in addition to the segmentation methods and 3D detector used in our strategy. In Chapter 4, we show the designed methods in more detail. In Chapter 5, the settings and results of the experiment are illustrated. In Chapter 6, we show the discussion and analysis of experiment results and suggest the future work for further study, followed by the conclusion in Chapter 7.

2

Background

This chapter gives the background of the methods applied in this project, including deep learning, object detection, the data implemented, and the object detection algorithms based on different kinds of sensors.

2.1 Deep learning

Deep learning is a branch of machine learning that using Artificial Neural Networks (ANNs) to achieve the "learning" process [10]. In recent years, deep learning has rapidly developed and is widely used in many different aspects, such as autonomous driving, object detection, artificial intelligence, and so on. Its architecture is the main force to promote its development. There are various deep learning architectures, for instance, Neural Networks (NNs), Convolutional Neural Networks (CNNs) and Residual Networks (ResNets), etc. This section briefly introduces some common architectures.

2.1.1 Neural Network

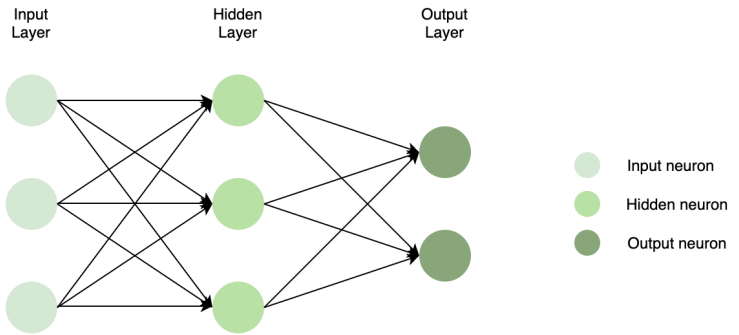


Figure 2.1: Example of the typical structure of Neural Networks. The arrows represent connections. In this example, there are 3 input units, 3 hidden units and 2 output units. The number of neurons can be modified depending on how complex the problem is.

ANNs are simply called NNs, which simulate the human brain to complete computation and learning processes. The entire NN systems are built in layers. Figure

2.1shows a generic structure of NNs: an input layer, a hidden layer and an output layer. Each layer is composed of neurons, which are the connection units in NNs. The neurons between two adjacent layers are generally fully connected. Each connection is assigned with a random weight, which represents how important the connection is. These weighted connections are summed up with an additional bias [11]. The input of NNs can be images, point clouds or documents, etc. The hidden layer always locates between the input and output layers. It generally contains activation functions that helping to pass the information and produce the result [11]. Through non-linear activation functions, NNs are capable of better approximate general target functions and handle more complex tasks. Because non-linear activation functions can help the network learn the non-linearity of target functions and it is impossible for linear activation functions. Therefore, non-linear activation functions, like Sigmoid, Tanh, ReLU functions, are always applied in NNs. The outputs from the hidden layer go through the final layer and the Softmax function is often applied here as a classifier. The results from the hidden layer include class scores that show the confidence of which category the detected object may belong to, which is named as confidence score. After normalized by Softmax function, these scores distribute in the interval of $[0, 1]$ and are summed up to 1. Then the scores can be understood as probabilities. Only if the class with the highest probability will be output as the final classification result.

2.1.2 Deep Neural Network

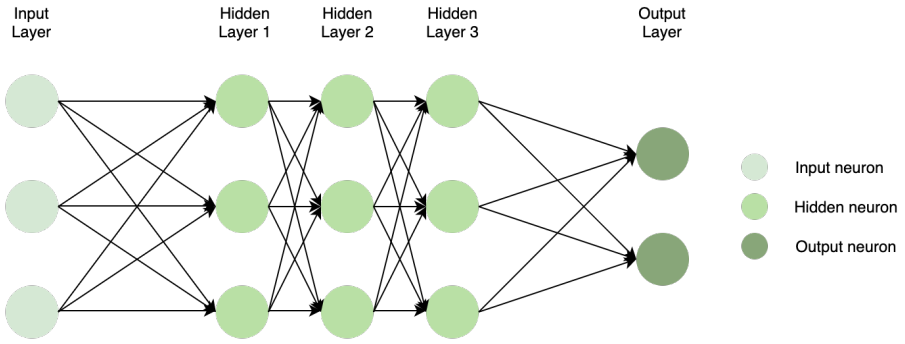


Figure 2.2: The generic structure of Deep Neural Network. This example gives 3 hidden layers. The number of hidden layers depends on the complexity of the learning target.

As the learning tasks become more and more complex, NNs are incapable to learn complicated features. Therefore, a general idea to breakout this limitation is adding more hidden layers so that the model can extract and learn much deeper and complicated features from the inputs. The NNs with more than two hidden layers are called Deep Neural Networks (DNNs), which are the subclass of NNs. The additional layers provide a higher capability to extract more features. DNNs are generally feed-forward networks that data flowing from the input to the output layer in a single direction. They initially assign a random weight to each connection which works similarly to NNs. Through the learning process, the weights will be changed during

each training iteration. If the network cannot identify features based on current weights, then the weights would be adjusted furthermore by a certain algorithm aiming to assign larger weights to particular parameters and emphasize their effect [10]. According to the complexity of problems, the activation functions in each hidden layer can be different. With the help of the additional hidden layers and arbitrarily combined activation functions, DNNs are therefore capable to extract features in detail and solve more complex tasks. This is the reason why DNNs are widely applied in many SOTA methods.

2.1.3 Convolutional Neural Network

CNNs are also the most commonly used neural network architecture in deep learning. Different from the other architectures, the hidden layers in CNNs generally involve convolutional layers, pooling layers and fully connected layers. CNNs have the capability to solve even more complex tasks and a larger number of inputs by means of kernels, as shown in Figure 2.3.

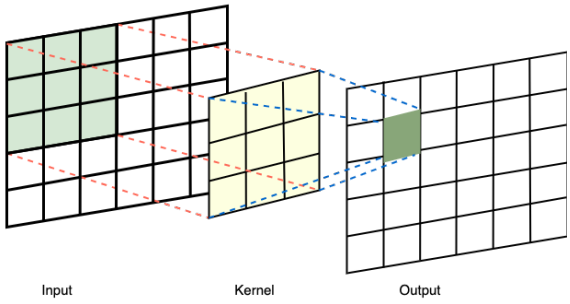


Figure 2.3: The working process of the convolutional kernel. In this example, the kernel is shown as yellow with size 3×3 which should be smaller than the size of the input.

Kernels can be formulated as "filters". By convolving the input data with kernels, different features can be extracted and the original relative position between objects can still be retained. After convolution, data pass from convolutional layers to the pooling layers. Pooling layers downsample these refined features to make concise feature maps. This step is able to reduce the number of learning parameters and avoid the overfitting problem [12], that the statistic model exactly fits the training data, therefore, fails to fit the new data. In the final step, the fully connected layers summarize all the extracted features and integrate them as one value via convolution.

Overall, kernels can well preserve spatially related information and help to extract various information, such as longitudinal information, edge information of objects, and so on. By applying kernels, CNNs are able to enhance the robustness of translation and scale. Additionally, the number of parameters that need to be learning is also decreased, which greatly simplifies the difficulty of training. Consequently, CNN is the most representative model in deep learning and plays an important role in object detection.

2.1.4 Applications

The development of deep learning brings remarkable improvements to many of the methods. The flexible network architecture of deep learning is one of the reasons why it has gradually become an essential tool in computer vision. The deep learning network can be designed with fewer hidden layers to efficiently solve simple problems. In the contrast, more hidden layers are needed in more complex tasks and non-linear activation functions can be arbitrarily combined in this case. Here list several common tasks in computer vision implementing deep learning:

- **object recognition:** it usually has images as inputs and the aim is to recognize all the objects in the input images. All the objects are annotated with their classes and bounded by either 2D or 3D bounding boxes in the output results.
- **object detection:** this task focuses on detecting only the target objects instead of all the objects. Thus, only the targets are annotated with classes and bounded by the bounding boxes in this task.
- **image segmentation:** partitions the inputs into segments. This task is at the pixel level, that is to say, each pixel will be labelled with a class at the end. And each segment composed of these pixels demonstrates an object.
- **object tracking:** this is a process to continuously locate and trace the moving objects in time series. Object tracking is crucial for autonomous driving.

2.2 Object detection

Object detection with the deep learning method can classify and localize objects from the input resources [13]. The traditional object detection process can be generally divided into three stages: region proposal, feature extraction, and object classification [14]:

- **Region proposal:** raises proposals to show where objects would possibly be. One of the typical methods is Faster R-CNN [15], which shifts multi-scale windows to transverse each part of the input and generate anchors, which is a set of predefined boxes with different sizes trying to capture objects in the input. Then, they go through Region Proposal Network (RPN), which proposes multiple likely objects that might be identified in the input resources. Anchors will finally be output as proposals only if they overlap objects over a certain threshold.
- **Feature extraction:** the key to distinguish objects is extracting their features, which can help discard redundant information and results in a reduction of data dimension. The most commonly used feature extractors are Haar-like [16] and Histogram of Oriented Gradients (HOG) [17].
- **Object classification:** a classifier is needed to predict the corresponding

category of each localized object. The related class information is annotated to the object as the final result of object detection, which makes the visual recognition more informative.

Objects can be at any place with various sizes or shapes, thus, there would be a huge amount of proposals that need to be generated, which consequently would increase the computation complexity. In other words, the traditional region proposal method is not efficient. This is the reason why people come up with the proposal-free idea to realize object detection. The most representative one is the centre-based method, which uses a single point with its bounding box information to represent each detected object [18]. More details of this method are demonstrated in the following chapters.

2.3 Image segmentation

The image segmentation process divides an input image into several segments or objects [19]. Image segmentation aims to simplify the image representation because the segments are more meaningful compared to the entire image. At the end of the entire process, each pixel is assigned a class label. For pixels with the same labels, they share certain characteristics [19]. From this, it could help object detection to better analyse that information.

Image segmentation can be classified into three groups regarding the results: semantic segmentation, instance segmentation, and panoptic segmentation. The instance segmentation identifies which pixels belong to that instance. In other words, instance segmentation focuses on differentiation between instances of the same class. For example, every person shown in an image is segmented as a single object distinguished from other persons. Typical instance segmentation methods are Mask R-CNN [20], and Cascade Mask R-CNN [21]. The semantic segmentation partitions the pixels into different classes and is unable to distinguish the individuals in the same class. For instance, all the people in an image are partitioned as an overall one object and all the cars shown in the same image are segmented as another object. SegNet [22] and DeepLab series [23][24][7][8] are the most classical semantic segmentation algorithms. The panoptic segmentation combines the idea of the semantic and instance segmentation, in this case, every pixel is assigned with its related class label and the objects in the same class are distinguished from each other. Panoptic FCN [25] has outstanding performance among the other panoptic segmentation methods.

2.4 Data

2.4.1 Sensor

Sensors in object detection tasks are used to collect data from the environment. These data are important in different object detection application tasks. For example, in autonomous driving, vehicles make correct decisions regarding the safe

paths to navigate based on the collected data by sensors. The most commonly used sensors are cameras, lidar and radars.

- **Cameras:** save the captured information, such as objects' colours, shapes and texture, as 2D images or videos. However, the video image quality is affected by the camera itself, as well the ambient light illumination and weather conditions. Then, to a great degree, the environmental condition determines the performance of camera-only object detection. Another problem is that even in the same condition, cameras have a shorter detection range than radars and lidar. And due to the perspective projection of cameras, the depth information, that the distance between cameras and objects, would be distorted.

- **Lidar:** is the shorthand of 'light detection and ranging'. It works by sending out laser beams that bounced back from the objects and lidar will collect the returned beams in the format of 3D point clouds. Lidar can work properly in a dark environment and detect objects further than 200 meters. Whereas, it lacks texture information and weather also affects its performance.

- **Radar:** is the abbreviation of 'radio detection and ranging'. They also collect data in the format of 3D point clouds, but radars use radio waves instead. This is the reason why they can detect farther objects than lidar [26] and work properly even in bad weather conditions, like rainy or snowy days. Nevertheless, radars generate sparser point clouds with lower resolution compared with lidar. This brings a challenge for object detection since it is hard to cluster the sparser point clouds and detect them as different objects.

2.4.2 Dataset

Dataset is defined as a collection of data samples, such as images, point clouds, or including both of them [27]. Besides that, datasets also provide annotations or ground truth labels, that are represented as bounding boxes, or 3D cuboids, etc. Generally, data annotation is a crucial step in a deep learning task. By "learning" the data with annotations in the training process, deep learning models are able to correctly identify and interpret objects. Consequently, the quality of the dataset is crucial for deep learning that partly determining the quality of output models. When choosing an appropriate dataset, other elements should also be considered, for instance, the sensors, fields, and objectives of the study. KITTI [1][28] and nuScenes [2] are the most common datasets in object detection and also applied in multi-object tracking. Apart from those, Waymo [29] and many dedicated datasets are extensively used in object tracking as well. Image segmentation generally uses COCO [30] and CityScapes [31]. With the development of computer vision, there are more and more datasets are generated with various properties, enabling to meet of different requirements of studies.

2.5 Related work

As mentioned above, there are various sensors used in object detection. According to the sensors implemented, object detection algorithms can be simply classified into camera-based, lidar-based, radar-based, and multi-sensor-based methods. This section will illustrate the principle of these methods. The corresponding algorithms in each case will also be briefly introduced.

2.5.1 Camera-based object detection algorithms

Cameras are kind of low cost and common sensors. Many algorithms therefore only implement cameras to achieve object detection. Typical camera-based object detection algorithms are Fast Region with CNN features (Fast R-CNN) and You Only Look Once (YOLO). Fast R-CNN [32] is an upgrade of the R-CNN method. R-CNN inputs every proposal into CNN independently and extracts the features of each proposal. If there are thousands of proposals going to the CNN one by one, it leads to thousands of repeated calculations. While Fast R-CNN inputs only the image into CNN capturing the features and then maps the proposals onto a certain convolutional layer and get the feature of each proposal, which greatly speeds up the training and evaluation processes. YOLO [33] does not localize the objects in the entire image. Instead, it divides the image into grids. Only if grids have high probabilities contain the target objects, then these grids will be further looked at.

Cameras bring the benefits of classifying objects because it contains colour and texture information. However, as illustrated before, ambient light illumination, weather and detection range are the vital challenges for camera-based object detection algorithms.

2.5.2 Lidar-based object detection algorithms

Different from cameras, lidar is capable to collect 3D information of objects. It also has the advantage of detecting objects in long ranges and providing stable point clouds even in low illumination conditions. But compared to cameras, point clouds from lidar lack the colour and texture information and the sparse point clouds also brings challenges to lidar-based object detection algorithms.

According to different 3D data representation in algorithms, lidar-based object detection algorithms generally have two categories: point-based and voxel-based.

- **Point-based:** The point-based methods feed the original point clouds into their models to extract features, which have the advantage to keep the accurate location information of each point.

  PointRCNN [34] is the first two-stage object detection algorithm that solely using lidar. In the first stage, after semantic segmentation, each point is labelled as either foreground or background. If the points correctly detect the object, these points are considered as foreground, otherwise, as background. Then, a small amount of proposals are generated based on the segmentation

results. Second stage transforms points of proposals to canonical coordinate in order to better learn local spatial features [34]. The final proposals are refined with higher confidence prediction by two-stage network architecture.

Although point-based methods retain accurate point positions and provide high object detection precision, they have drawbacks of complex computation because they consider all the original points. Therefore, voxel-based algorithms are raised up to tackle the problem in point-based methods, in the meantime, realizing accurate object detection only using point clouds.

- **Voxel-based:** The voxel-based methods divide point clouds into regular 3D grids [35] that each grid considered as a voxel. The size of voxels and the maximum number of points in each voxel are different regarding different algorithms. Each point is assigned an index to a specific voxel. If the voxel has already reached the maximum number of points, then the left points will be discarded [36]. Voxel-based methods then calculate in voxel-level rather than pointwise as in point-based methods, which greatly simplifies the complexity of calculation. Whereas, the drawback in voxel-based methods is also apparent that, to some extent, they delete some points which may contain important information.

  VoxelNet is a voxel-based end-to-end algorithm [36]. The first step of its network is to partition the point clouds into 3D voxels. If the voxel contains more than T number of points, then it randomly samples T points retaining in the voxel and other points are removed. Through this, the imbalance number of points between voxels could be relieved. Then, CNN extracts features in each voxel. The RPN is finally applied to generate the bounding boxes.

  PointPillars algorithm raises another kind of data representation: pillars. In contrast to the voxels, points are divided by some vertical columns, and the spatial along the z-direction is infinite [37]. The entire PointPillars method can be divided into three steps. Firstly, PointNet [38] is used to extract features from input point clouds. The points with retained features form pseudo-2D images, which can be then processed by the image object detection methods. In the second step, these pseudo-2D images are passed into the CNNs to extract features. In the end, the detection head uses the features on pseudo-2D images to predict 3D bounding boxes.

  Besides VoxelNet and PointPillars, there are other voxel-based object detection methods only based on point clouds. For example, Sparsely Embedded Convolutional Detection (SECOND) implements the sparse 3D convolution to increase the efficiency of CNN and introduces the angle loss to improve the direction estimation. Part-A^2 Net [39] converts point clouds into voxels as well and divide the entire object detection process into two parts: part-aware and part-aggregation. Part-aware is used to predict the intra-object part location and generate proposals. Part-aggregation refines the proposals and locations to final detected targets.

Since both point-based and voxel-based methods have their own advantages and drawbacks. Point Voxel RCNN (PVRCNN) [34] proposes a new network structure combining the properties of these two categories. It firstly implements the voxel-based idea to efficiently encode features in multi-scales that generating proposals with high quality. The point-based idea is applied afterwards to find local detailed information. Consequently, PVRCNN takes the benefits from both point-based and voxel-based methods so that it is one of the SOTA lidar-only object detection methods.

2.5.3 Multi-sensor-based object detection algorithms

Since different sensors have different properties, taking advantage of the complementary properties from different sensors is a possible method to achieve more accurate object detection. The most commonly used sensor fusion methods are the camera-lidar method and the camera-radar method.

- **Camera-lidar sensor fusion:** camera-lidar is the most common sensor fusion combination since fusion strategy can better offset the drawbacks in these two types of sensors. Whereas, other new challenges are generated. Viewpoint misalignment is one of the most important problems to be solved. Many excellent lidar-based detection algorithms apply convolution in BEV because detection in the BEV has advantages, such as minimal occlusions, less depth-blurring effect and more information than front-view. Even if the detectors work in the other view, it is simple to convert generated lidar detection results to BEV. Whereas, cameras are generally in front or back view, and it is hard to get the BEV from cameras. Therefore, the core of the current research is how to use cameras view to consolidate the BEV.

  Some effective ideas have emerged to solve this problem. For example, object-centric fusion, pioneered by MV3D [40] that using images and point clouds data from BEV as well as the front view to extract features, and then merged features together to the same dimension by Region Of Interest (ROI) pooling. The object-centric methods have the common drawback of being slow and complicated. Additionally, Roddick et al. [41] proposed explicitly transforming to convert the image into a BEV representation and fusion at there. Subsequent work then builds fusion based on this idea, but the performance falls short of SOTA [42] and requires several expensive steps of processing to build the pseudo-point clouds.

  There is a new sensor fusion method, PointPainting [3], which is straightforward, highly operable, and effective. It has been verified that the algorithm achieved large improvements on three different SOTA methods, Point-RCNN [34], VoxelNet [36] and PointPillars [37] on both KITTI and nuScenes datasets. In this thesis, we use the camera-lidar sensor fusion strategy based on this PointPainting algorithm.

- **Camera-radar sensor fusion:** since lidar is more sensitive to weather, which results in limitations of the camera-lidar fusion method. Instead, radars use

radio frequencies to detect targets rather than lasers. Therefore, it has a higher capability to detect objects in a longer range. Additionally, it observes the range rate of target by Doppler Effect, which would also benefit other aspects such as object tracking task. The most popular camera-radar fusion object detection algorithm is CenterFusion **CenterFusion**. It uses the centre point of objects to firstly identify objects on images. The radar data are associated with the corresponding centre points on images by the frustum method, which is a novel data association method raised in CenterFusion. The radar-based feature map is generated by the associated data and then complement the extracted features from images. In the final step, these features are used to regress to object properties and output detection results. However, radars provide even sparser measurements than lidar.

3

Theory

This chapter introduces related theories for better understanding the designed sensor fusion object detection algorithm. As mentioned, this thesis proposes a deep learning fusion strategy called Painted CenterPoint (PCP), inspired by PointPainting fusion algorithm [3]. The idea is that it employs an image segmentation network to calculate segmentation scores for each pixel. By projecting point clouds into an image, the pixel position corresponding to each lidar point can be obtained. Then it appends the segmentation score of this pixel to this point. Through projecting and appending, painted point clouds are generated. Finally, a lidar-based 3D detection network utilizes these painted point clouds, which provides final detection results.

The segmentation methods would differ how PointPainting benefits the CenterPoint detector in different metrics and scenarios, so we introduce three segmentation methods: DeepLabV3 [7], DeepLabV3+ [8] and Hierarchical Multi-scale Attention (HMA) [9].

As mentioned in Chapter 1, we use an excellent lidar-based 3D detection network called CenterPoint [4]. Since this algorithm has caused new fashion and trends in the field of object detection, it is worth studying if introducing camera information would further improve the precision.

In this chapter, we will describe the technical details about the methods: PointPainting algorithm, three image segmentation methods and CenterPoint.

3.1 PointPainting fusion

PointPainting algorithm is consisted of three steps: 1) Image Segmentation: an image-based segmentation network that computes pixel-wise segmentation scores 2) Fusion (Painting): original lidar points are painted with segmentation scores. 3) Lidar Detector: a lidar detection network that employs on painted point clouds to achieve detection results [3]. The overall structure of PointPainting is demonstrated in Figure 3.1, and Algorithm 1 presents the pseudocode. We give more explanation of each step in Figure 3.1 as follows.

In advance of introducing the method itself, it would be better to make explicit of denotations that will be used later. The input lidar point clouds is denoted as

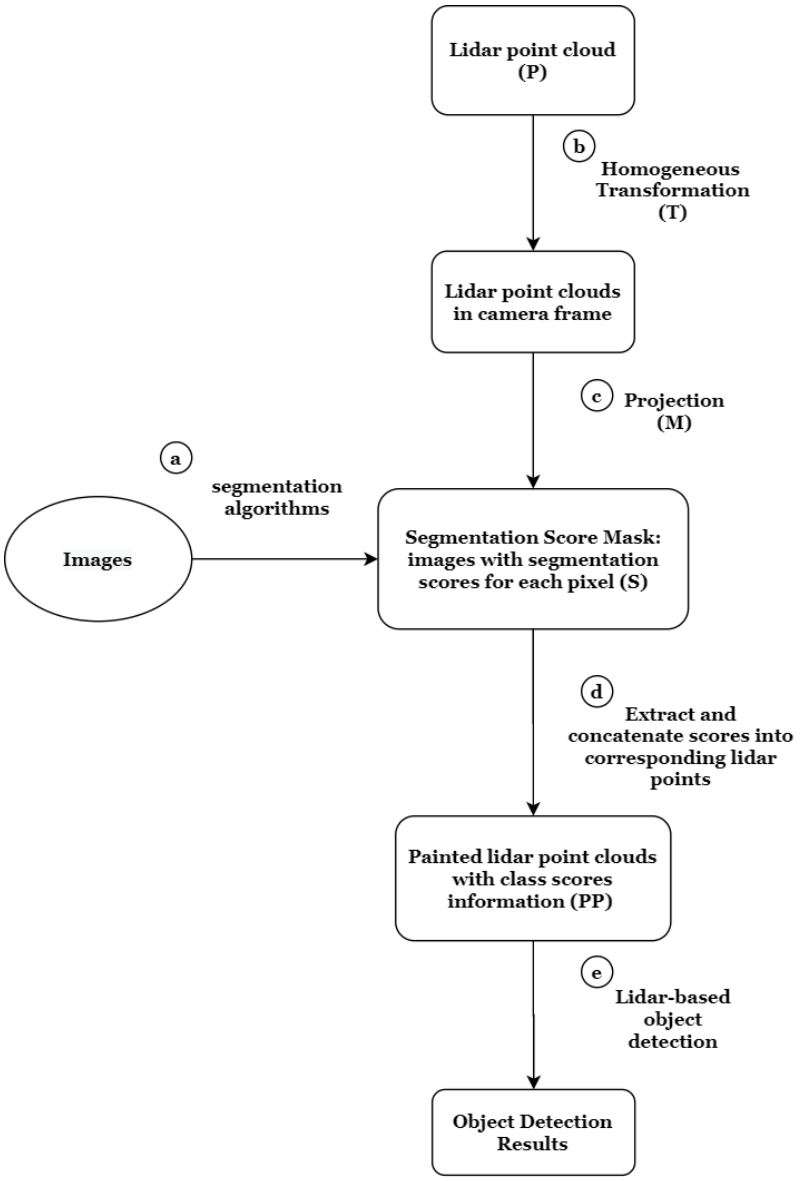


Figure 3.1: The PointPainting architecture

$P \in \mathbb{R}^{N \times D}$, where N is the number of lidar points, and $D = 4$ is the number of dimensions. Then we have each lidar point as $p = [x, y, z, r]^{\mathrm{T}} \in P$ where x, y, z are 3D coordinates and r is the reflected intensity. And the corresponding pixel position of each lidar point is $p_{image} = [u, v]^{\mathrm{T}}$. Then the painted lidar points is $pp = [x, y, z, r, s_{Veh}, s_{Ped}, s_{Cyc}, s_{Back}]$.

a. Image Segmentation Network

Firstly, PointPainting uses an image segmentation network to obtain segmentation scores (class scores) at pixel level. The input is an image $I \in \mathbb{R}^{W \times H}$, where W and H are the width and height of the image respectively. After going through the segmentation network, it gives pixel-wise class scores which forms a segmentation mask $S \in \mathbb{R}^{W \times H \times C}$ where C is the number of classes. The segmentation scores

Algorithm 1 The PointPainting Algorithm (Details of the Algorithm and parameters are described in the text)

Input:
Lidar point cloud $PointsList\ P \in \mathbb{R}^{N \times D}$ with N points.
Segmentation scores mask $ScoreMask\ S \in \mathbb{R}^{W \times H, C}$ with C classes.
Segmentation scores for each pixel $PixelScore\ s \in S$.
Homogenous transformation matrix $T \in \mathbb{R}^{4 \times 4}$.
Camera matrix $M \in \mathbb{R}^{3 \times 4}$.

Output:
Painted lidar points $PaintedPointsList\ PP \in \mathbb{R}^{N \times D+C}$

for each $p \in P$ **do**
$p' =$ **Transform** (T, p);
$p_{image} =$ **Project** (M, p');
$s = S[p_{image}[0], p_{image}[1], :]$
$pp = [p; s]$
$PaintedPointsList.ADD(pp)$
end for

show how likely this pixel belongs to a certain class so the segmentation mask contains summary classification information of an image. We use three segmentation networks and the principles of them will be explained in the next section.

b. Homogeneous transformation

In order to add the segmentation scores to lidar points, it is necessary to first find the corresponding pixel positions of the lidar points. PointPainting gets pixel positions by projecting points into images. However, since lidar points are in the lidar coordinate frame, they should first be transformed to the camera coordinate frame. In this thesis, we use a homogeneous transformation matrix $T \in \mathbb{R}^{4 \times 4}$ given by the KITTI dataset to convert lidar points p to the camera frame and get p' (see *Transform* in Algorithm 1).

c. Projection

After transforming the lidar point to camera coordinate frame, we use the camera matrix $M \in \mathbb{R}^{3 \times 4}$ to project points into the image and get the pixel position $p_{image} \in \mathbb{R}^2$ of the lidar point in the image (see *Project* in Algorithm 1).

d. Concatenating scores into lidar points

After step b and c in Figure 3.1, it is able to obtain the corresponding pixel positions for each lidar point. Then in step d, we extract the segmentation scores by indexing pixel positions in the segmentation mask S, which is shown as $s = S[p_{image}[0], p_{image}[1], :]$ in Algorithm 1. That is to say, the first two channels of S are the x, y pixel positions of the projected points and the rest channels are segmenta-

tion scores $s \in S$ for each classes and $s \in \mathbb{R}^C$ where C is the number of categories. Then we concatenate the segmentation scores s to original points p by simply adding more channels and get the painted point: $pp = [p; s]$. Finally, we add this painted point into the $PaintedPointsList$.

e. Lidar-based detection

Till now, we get painted lidar point clouds including image segmentation information and next step, we employ lidar-based detector CenterPoint on these painted points to achieve object detection results. More explanation will be given in Chapter 4, but what should be mentioned in advance is that when using PointPainting, we do not need to make many modifications to the original lidar-based detection network for new input adapting. The only requirement is enlarging input dimensions to fit newly-added channels for storing segmentation scores.

3.2 Image segmentation

As mentioned, the segmentation methods would differ how PointPainting benefits the CenterPoint detector, so we introduce three segmentation methods. In this section, we are about to explain the principles of three segmentation networks applied: DeepLabV3 [7], DeepLabV3+ [8] and Hierarchical Multi-scale Attention (HMA) [9].

3.2.1 DeepLabV3

Prior to introduce DeepLabV3, the two common challenges should be mentioned when applying Deep Convolutional Neural Networks (DCNNs) on image segmentation tasks. First, the reduced feature resolution which is caused by consecutive operations of striding and pooling always leads to loss detailed spatial information. Second, an image usually contains multiple scaled objects [7], which results in difficulty in segmenting objects with various scales.

To handle these problems, DeepLabV3 utilizes atrous convolution [43] in parallel or cascade to capture multi-scale context by adopting multiple atrous rates [7]. The idea of atrous convolution is that it inserts holes between filter weights. That is to say it extracts denser feature maps by removing downsampling operations from the last few layers and upsampling the corresponding filter kernels. By using atrous convolution, it is able to control how densely to compute feature responses.

Besides introducing atrous convolution, DeepLabV3 also proposes to augment the previously proposed Atrous Spatial Pyramid Pooling (ASPP) [44] module, which probes convolutional features at multiple scales, with image-level features encoding global context and further boost performance [7]. ASPP is capable of effectively capturing multi-scale information by changing the atrous rates. Experimental results demonstrate that the proposed DeepLabV3 model significantly improves over previous DeepLab series [23] [24].

3.2.2 DeepLabV3+

DeepLabV3+ further upgrades the performance of DeepLabV3. It is built on an encoder-decoder structure, in which the encoder part applies DeepLabV3. By arbitrarily controlling the extracted resolution of encoder features, DeepLabV3+ enables the trade-off between precision and complexity. Additionally, DeepLabV3+ adds a simple yet effective decoder module. The encoded features are firstly upsampled and then concatenated with the corresponding low-level features from the network backbone. Followed by upsampling, several 3×3 convolutions are then applied to refine the features.

Compared with DeepLabV3, DeepLabV3+ compensates for missing detailed information around object boundaries by adopting this encoder-decoder structure. The experimental evaluation demonstrates that DeepLabV3+ is a faster and stronger network and defeats the top-performing model on PASCAL VOC 2012 test[45].

3.2.3 Hierarchical Multi-scale Attention

The multi-scale inference is generally used after images with multiple scales passed through a network. Average-pooling and max-pooling are the most common inference methods to combine multi-scale predictions. Average-pooling assigns the same weights to multi-scale predictions and gets the average of them as the output. And max-pooling gets the maximum of multi-scale predictions as the output. These two pooling methods are too simple and sub-optimal in real traffic scenarios. In theory, it would be better to handle objects in different sizes at different resolutions. For example, in the first row of Figure 3.2, the thin posts are inconsistently segmented at lower resolution (0.5x), but better predicted at higher resolution (2.0x). In the second row, the large road / divider region is better segmented in the scaled down (0.5x) image [9]. So the best idea is to give predictions with pertinence based on different scales and types of objects, rather than ambiguously take the maximum or average.

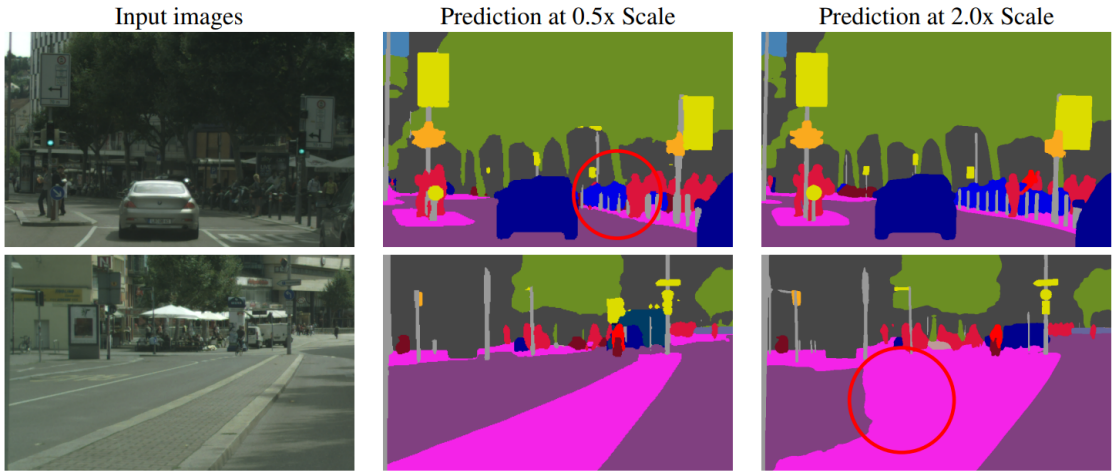


Figure 3.2: Two failures modes for semantic segmentation because of inference scales [9].

Hierarchical multi-scale attention (HMA) [9] gives a fairly good solution. HMA is a hierarchical attention-based approach that combines multi-scale predictions at a pixel level. During training, the network learns to predict relative attention between adjacent-scale-pairs images. And during inference, the learned attention is used to better combine N scales of predictions hierarchically. That is to say, the lower scale attention determines the contribution of the next higher scale [9].

Experimental evaluation demonstrates that HMA gives higher accuracy (51.6 IOU) than the baseline averaging approach (49.4 IOU) [9], where Intersection Over Union (IOU) is an evaluation metric. This hierarchical attention mechanism can improve classification confusion and segmentation accuracy, fine details of objects and be more efficient in memory and computation.

3.3 3D object detection

As mentioned, an excellent lidar-based 3D detection algorithm is applied in this thesis, which is called CenterPoint [4]. And we want to study if introducing camera information to this method would further improve the detection performance. Since CenterPoint is a center-based detection network, we first elaborate the idea of center-based detection methods. Then we explain the principle of CenterNet [18], which promotes the launch of CenterPoint and finally decscibe CenterPoint.

Actually, most commonly used 3D detection algorithms are anchor-based, which is a set of predefined boxes with different sizes trying to capture objects in the input. So these anchor-based algorithms often propose many 3D candidate boxes. In contrast, center-based algorithms do not demand candidate boxes, instead, they directly detect the likely object centers. Thus, the computation based on points is much simpler than boxes. The reason is that if the candidate is in box representation, we need to calculate IOU to decide if the candidate box should be reserved or not. Calculating the overlap or intersection of boxes requires much computation. Whereas, if the point representation is applied, we only need to calculate the distance between the ground truth point and the proposal, which is much easier. So center-based algorithms can give results in shorter time than anchor-based. So when facing up problems like corners, center-based algorithms will fit the new 3D bounding boxes much faster than anchor-based algorithms.

3.3.1 CenterNet

As mentioned, most 2D object detection methods are based on candidate boxes, such as RCNN series [32] [15] [46], YOLO [33], SSD [47] and so on. However, CenterNet [18] does not need candidate boxes. It directly detects the implicit object centers instead.

CenterNet uses a single point to represent the center of object, see Figure 3.3. The other desired properties of an object, like size and orientation, can be regressed from image features at the center. Therefore, the focus of CenterNet proposes likely centers. From this aspect, the detection task is converted to a standard keypoint

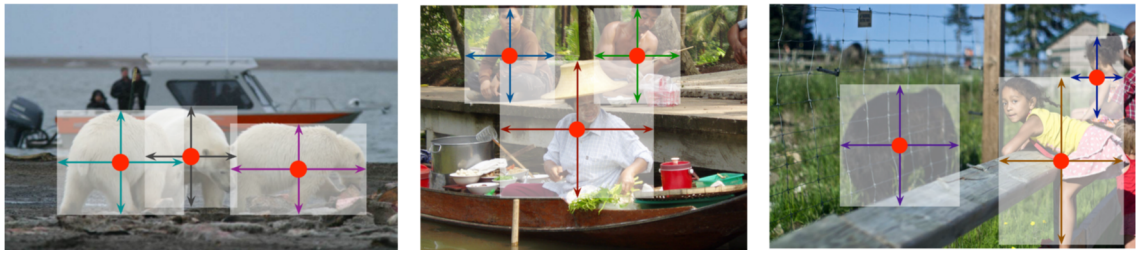


Figure 3.3: Centre-based representation.[18]

estimation problem [48]. The principle of CenterNet is as follow.

a. Predicting heatmaps

Above all, CenterNet passes the images into a keypoint detection network. Generally, the keypoint detection network is either a fully convolutional encoder-decoder network ResNet [49], or Deep Layer Aggregation (DLA) [50]. It outputs a predicted keypoint heatmap $\hat{Y} \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times C}$ where W and H are the width and height of input images, R is the stride and C is the number of object classes. So we will output one heatmap for each class.

b. Training

CenterNet trains the keypoint detection network using ground truth keypoints: $p \in \mathbb{R}^2$ of class c. The ground truth keypoints are centers of 2D ground truth bounding boxes, which is applied to obtain ground truth heatmaps and the predicted heatmap $\hat{Y}$ can therefore compare with it. To get the ground truth heatmap, CenterNet firstly downsamples ground truth keypoints by the stride factor of R and gets $\tilde{p} = \lfloor \frac{p}{R} \rfloor$. Then it converts keypoints into a heatmap $Y \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times C}$ by using Gaussian kernals. The objective of training is a penalty-reduced pixel-wise logistic regression with focal loss [18].

c. Inference

During inference, peak values in the predicted heatmap are compared and determine which of them are the most possible object centers. For each heatmap given by step **a**, CenterNet first extracts peaks and goes through all these peak values. It reserves the peaks larger than or equal to its 8 neighbors and keeps top n peaks as the detected center points $\hat{p} = (\hat{x}_i, \hat{y}_i)$ where $(\hat{x}_i, \hat{y}_i)$ is the 2D coordinates. Finally, CenterNet outputs the keypoint value $\hat{Y}_{x_i, y_i, c}$ as the confidence score for that class. So this is how CenterNet detects centers and class scores of objects through heatmaps.

d. Predicting other properties

If user also wants to output the b-boxes around objects. CenterNet can add regression heads —simple 2D CNNs. For example, if a regression head is added to CenterNet for getting b-boxes sizes, then this head utilizes image features to regress to b-boxes sizes $\hat{S} \in [0, 1]^{\frac{W}{R} \times \frac{H}{R} \times 2}$. This supervision is only done at predicted keypoint locations $\hat{p}$. Furthermore, CenterNet also uses these regression heads realizing

optimization. For example, to recover the discretization error caused by the stride R, a local offset prediction $\hat{O} \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$ can be added for each predicted center. The offset is also achieved by using and training a regression head. Each property has individual regression head, which can be simply trained with L1 Loss —the least absolute errors between the prediction and ground truth. Finally, through the sizes and offsets, CenterNet plots a 2D bounding box at location:

$$(\hat{x}_i + \hat{o}_{x_i} - \frac{\hat{w}_i}{2}, \hat{y}_i + \hat{o}_{y_i} - \frac{\hat{h}_i}{2}, \hat{x}_i + \hat{o}_{x_i} + \frac{\hat{w}_i}{2}, \hat{y}_i + \hat{o}_{y_i} + \frac{\hat{h}_i}{2}) \tag{3.1}$$

where $\hat{O}_{x_i,y_i} = (\hat{o}_{x_i}, \hat{o}_{y_i})$ is the offset and $\hat{S}_{x_i,y_i} = (\frac{\hat{w}_i}{2}, \frac{\hat{h}_i}{2})$ is the size.

3.3.2 CenterPoint

After the above foreshadowing, it should be uncomplicated to understand the principle of CenterPoint [4]. Its architecture is shown in Figure 3.4. CenterPoint is conceptually quite similar to CenterNet but develops the idea of CenterNet on 3D lidar point clouds. The main difference is that it uses a 3D Backbone to encode 3D features into BEV feature maps. Then the following steps are all the same as in CenterNet. It also predicts centers by a keypoint detection network, which is called Center Heatmap Head in CenterPoint and the other properties are predicted by training Regression Heads. Consequently, the ideas of 3D Backbone and Detection Heads are dicussed as follow.

3D Backbone

CenterPoint uses VoxelNet as 3D Backbone to extract BEV features map $M \in \mathbb{R}^{W \times H \times C}$ from lidar point clouds. Implementing the idea of CenterNet in BEV is better than using it in camera-view. The reason is that, in BEV, depth information is the absolute distance without distortion. However, due to perspective effect, depth information is usually distorted in camera-view.

Detection Heads

Next CenterPoint uses Center Heatmap head and Regression Heads to predict centers and other attributes, including 3D sizes, 3D orientations, and offsets.

- **Center Heatmap head:** The Center heatmap head is a keypoint estimation network to predict object centers by using heatmaps. CenterPoint projects centers of 3D ground truth b-boxes to BEV and regards them as ground truth keypoints. This is the only difference between CenterPoint and CenterNet. Besides that, all the steps are the same as in CenterNet.
- **Regression head:** To get other attributes, CenterPoint adds separate Regression heads. These heads regress additional properties, such like an offset $\hat{O} \in \mathbb{R}^2$, sizes of 3D b-boxes $\hat{S} \in \mathbb{R}^3$, and orientation angles $(sin(\alpha), cos(\alpha)) \in \mathbb{R}^2$ at each object center.

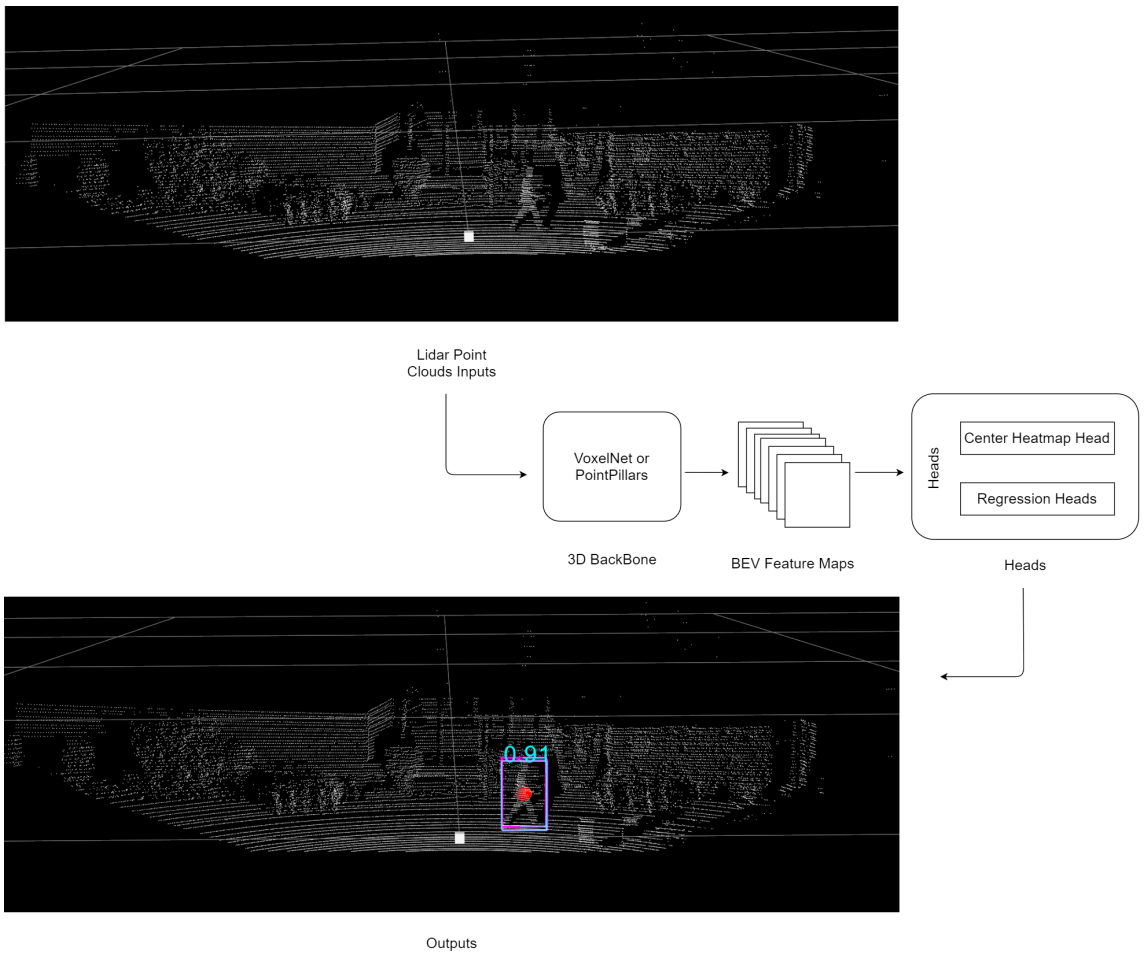


Figure 3.4: CenterPoint structure. The input of CenterPoint is point clouds. The Center Heatmap head outputs predicted centers and classes of objects. The regression head yields the 3D sizes and orientation information of 3D bounding boxes. At the end, the results of CenterPoint shown as lidar point clouds with 3D b-boxes and confidence scores for classes. The colour of b-box corresponds to class: Vehicle in green, Pedestrian in blue and Cyclist in yellow.

4

Method

4.1 Data

As mentioned in Section 1.3, the purpose of this thesis is to study if and how camera information can benefit the CenterPoint algorithm and to show the potential of sensor fusion object detection methods in some scenarios, instead of providing a model with comparable precision to the benchmarks. According to this principle, in this thesis, we train and evaluate our models on a very commonly-used and simple dataset —the KITTI dataset [1].

KITTI collects data from cities and countrysides in Germany. The data collection platform recorded 6 hours of traffic scenarios at 10-100 Hz using a variety of sensor modalities such as high-resolution grayscale and color stereo cameras, one 64-beam Velodyne 3D laser scanner and a high-precision GPS/IMU inertial navigation system [1]. All the data is collected in fine weather as well as daytime, which means that the images are almost not distorted by weather or light illumination. In this thesis, we use the measurements from two color stereo cameras and the Velodyne 3D laser scanner. There are 7481 images and point clouds samples available. We use 3712 of them for training, 1528 for validation, and 3769 for evaluation. This thesis mainly focuses on 3 most representative classes: Vehicle, Pedestrian and Cyclist.

Additionally, as mentioned in Section 1.3, synchronization between cameras and lidar is a big challenge when studying the fusion of cameras and lidar. KITTI is able to solve this problem perfectly because it provides calibrated, synchronized and timestamped data, as well as the rectified and raw image sequences [1]. First of all, Kitti mounts a reed contact at the bottom of the lidar scanner, and then it continuously rotate and receive response measurements. But it will only trigger the cameras when the scanner also facing forward as cameras. This minimizes differences in range and image observations caused by dynamic objects [1]. Moreover, it also gets images and lidar points at the same timestamps. KITTI itself also provides fairly good calibration matrice between each cameras. It also provides the transformation and projection matrice from lidar coordinate frame to camera coordinate frame. As mentioned in Section 3.1 Step **b** that, in order to add the segmentation scores to points, the first step is to find the corresponding pixel positions of the lidar. The transformation and projection matrice can help us to achieve this. To sum up, considering the objectives, KITTI dataset is therefore implemented to train and

evaluate results in this thesis.

4.2 Training

Since the fusion model is built based on PointPainting, the training process therefore follows the same structure in Figure 3.1 going through the steps **a** to **e**. In this section, we give more technical details involved in these steps.

4.2.1 Getting Segmentation Scores

As mentioned in Section 3.1 Step **a**, PointPainting uses an image segmentation network to get segmentation scores at pixel level and all the scores for images forms the segmentation masks. We get segmentation masks by passing through images into a pre-trained image segmentation network. Here we use use three pre-trained networks: DeepLabV3, DeepLabV3+ and HMA from MMSegmentation network architecture —an open-source semantic segmentation toolbox based on PyTorch. The processes of getting segmentation scores are almost the same among these three methods, so we introduce one of them, DeepLabV3, as a representative and the other two are similar.

DeepLabV3: The first step is image pre-processing. The inputs are mini-batches of 3-channel RGB images with shape $(N, 3, H, W)$, where N is the number of input images, H and W are height and width of images. The input values are converted from range of $[0, 255]$ to $[0, 1]$ and then be normalized.

After pre-processing, images are passed through a DeepLabV3 pre-trained model—the DeeplabV3-ResNet101 [51]. Softmax function is applied in the output layer to normalize prediction probabilities of each class. Since DeeplabV3-ResNet101 is trained for 21 classes, the outputs are segmentation masks of shape (C, H, W), where $C = 21$. Each pixel contains the probabilities of class prediction for 21 classes. The maximum one indicates that the pixel most likely belongs to this class.

Finally, we do some post-precessing of the segmentation masks. DeepLabV3 has 21 classes, but in this thesis, we are only interested in the class of Vehicle, Pedestrian and Cyclist. Among the 21 channels, the channel number 7, 15 and 2 respectively denote the class of Vehicle, Pedestrian and Cyclist. We extract pixels related to these 3 classes and all the other uninterested-classified pixels are regarded as Background with a prediction probability of 0. For simplicity, the channel number is reassigned as 0, 1, 2, 3 for Background, Vehicle, Pedestrian and Cyclist respectively.

DeeplabV3-ResNet101 is trained on a subset of the COCO train2017 dataset [52], on the 20 categories present in the Pascal VOC dataset [45]. The results of pre-trained model evaluated on the COCO val2017 dataset has 67.4 Mean IOU [51].

DeepLabV3+: The process of achieving segmentation masks is the same as DeepLabV3 including the steps of pre-processing, passing images through the pre-trained model and post-processing. The only difference is the applied pre-trained

model. Here we use the DeeplabV3Plus-ResNet101 [53], which is trained on Cityscapes datasets [54].

HMA: The steps of getting HMA segmentation masks are similar as mentioned above, except for the pre-trained model. Here we use the pre-trained model as mentioned in [9]. This model is firstly trained on the larger Mapillary dataset [55] and then trained on Cityscapes [54]. For Cityscapes, the model used 3,500 finely labelled images. And for the other 20,000 coarsely labelled images, HMA uses an auto-labelling strategy to classify these coarsely labelled images. So the final pre-trained model is trained on fine-labelled images in addition to the auto-labelled coarse images. Then, by passing through the images into this model, the segmentation scores are obtained.

4.2.2 Concatenating Scores into Points

As mentioned in Section 3.1 Steps **b**, **c** and **d**, the corresponding pixel positions for lidar points should be found so that we can add the segmentation scores to points. Since we do this by transforming and projecting lidar points into images, it is necessary to apply transformation and projection matrice provided by KITTI, as mentioned in Section 4.1.

KITTI has four types of camera information, which are grayscale and color images from left and right stereo cameras: $i \in \{0, 1, 2, 3\}$. We denote left-grayscale, right-grayscale, left-colour and right-colour image with the indexes of $i \in \{0, 1, 2, 3\}$, respectively. In this work, we solely employ two colour images with index: $i = 2$ or 3. Then the aim comes to project a 3D lidar point $p = [x, y, z, r]^{\mathrm{T}}$ to an image pixel $p_{image} = [u, v, 1]^{\mathrm{T}}$ on image with $i = 2$ or 3.

First, the point should be transformed from lidar coordinate frame to camera coordinate frame. KITTI provides a 3D rigid body transformations matrix $\mathbf{T}_{\mathbf{lidar}}^{\mathbf{cam}} \in \mathbb{R}^{4\times4}$ for each captured image, which can be considered as the homogenous transformation matrix. So the transform formula is shown in Equation 4.1:

$$p' = \mathbf{T}_{\mathbf{lidar}}^{\mathbf{cam}} p \tag{4.1}$$

Notice that this transformation matrix only transforms points to the reference camera coordinate system —the left-grayscale image of index $i = 0$. The rectifying rotation matrix $R^{(0)}$ is therefore needed to transform from camera frame $i = 0$ to $i = 2$ or 3. The homogeneous transformation matrix T mentioned in Algorithm 1 should be $\mathbf{R}^{(\mathbf{0})}\mathbf{T}_{\mathbf{lidar}}^{\mathbf{cam}}$ in this case. Consequently, the camera-coordinated lidar points can be transformed from camera frame $i = 0$ by Equation 4.2:

$$p'' = \mathbf{R}^{(\mathbf{0})}\mathbf{T}_{\mathbf{lidar}}^{\mathbf{cam}} p \tag{4.2}$$

Projecting lidar points to the images, we directly utilize the projection matrix $\mathbf{P}^{(2)}$ and $\mathbf{P}^{(3)}$ for left-colour and right-colour camera frames from KITTI. So the cam-

era/projection matrix M mentioned in Algorithm 1 can be written as $\mathbf{P^{(i)}}$. Then, the projected lidar point —the corresponding image pixel p_{image} in the image can be achieved by Equation 4.3:

$$p_{image} = \mathbf{P^{(i)}} p'' = \mathbf{P^{(i)}} \mathbf{R^{(0)}} \mathbf{T^{cam}_{lidar}} p \tag{4.3}$$

Finally, the segmentation scores can be extracted at pixel position p_{image} for each lidar point and concatenate the scores into each point $pp = [p, s]^{\mathrm{T}}$. Then by concatenating the pixel segmentation scores $s = S[p_{image}[0], [p_{image}[1], :]^{\mathrm{T}}$ with the original lidar vector $p = [x, y, z, r]^{\mathrm{T}}$, the painted lidar points $pp = [x, y, z, r, s_{Veh}, s_{Ped}, s_{Cyc}, s_{Back}] \in \mathbb{R}^{4+C}$ can be obtained.

4.2.3 Training CenterPoint Detector

As mentioned in Section 3.1 Steps **e**, we employ the lidar-based detector: CenterPoint on these painted lidar point clouds. The network structure, training parameter settings and training process are introduced in this section.

4.2.3.1 CenterPoint network structure

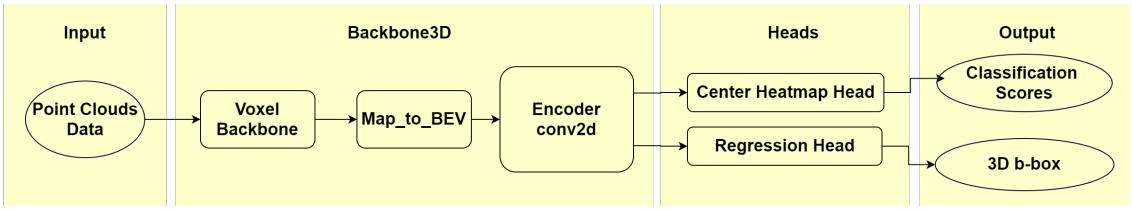


Figure 4.1: The network architecture of CenterPoint.

The main structure of CenterPoint is shown in Figure 4.1 which is similar to Figure 3.4 but refined with more details. As mentioned in Section 3.3.2, Centerpoint detector consists two parts: Backbone3D and Detection Heads. It uses VoxelNet as 3D Backbone to extract BEV features maps from point clouds and then use Detection Heads to predict centers and other properties.

In this thesis, we use VoxelNet as 3D Backbone. VoxelNet is consisted of 6 sparse convolutional layers including input and output layers. And each sparse convolutional layer is composed of either SubMconv3d, SparseConv3d or both of them. Batch Normalization and Relu layers are also applied in each sparse convolutional layer. Then the 3D feature maps from VoxelNet are encoded into BEV through the "Map_to_BEV" and "Encoder conv2d" modules. The "Encoder conv2d" module is a 2D Convolutional encoder including six 2D Convolutional layers and one 2D Convolutional Transpose output layer and each two layers are separated by Batch-Normalization and Relu layers. The output of 3D Backbone is the extracted BEV features maps.

Based on these BEV features, we use the Center Heatmap head and the Regression head as Detection Heads in CenterPoint to predict centers and other properties. The

Center Heatmap head is a keypoint estimation network to predict object centers by using heatmaps, so the outputs are 2D Gaussian heatmaps with peaks at center locations of any detected objects. The training target is ground-truth heatmaps produced by 2D Gaussian based on ground-truth labels and b-boxes extracted from the 3D backbone. Besides the object centers, we can also get other attributes of objects like 3D sizes, 3D orientations, offsets and even velocities by the Regression heads, which is a simple 2D CNN. The Regression heads use L1 Loss—the least absolute errors between the prediction and ground truth. Finally, we can plot the centers, classes with confidence scores and 3D b-boxes of detected objects.

4.2.3.2 Training parameter settings

During model training process, the information for applied hyperparameters and optimizer are concluded as follows. One cycle learning rate scheduler is used with an initial learning rate of 0.003 and learning rate decay of 0.1. Here, Adam method is utilized as optimizer with weight decay is 0.01 and the momentum is 0.9. The batch size for each GPU is set as 2. And we have attempted different training epochs to get a reasonably well-performed model.

4.2.3.3 The training process

In this thesis, the sensor fusion model, Painted CenterPoint (PCP), is trained on a pre-trained model of CenterPoint (CP). As can be seen in Figure 4.2, the first two learnable layers for CP is shown on the left with 4 neurons. The corresponding network structure for PCP is shown on the right. The difference is that PCP has 8 input neurons where the additional neurons store the segmentation scores for 4 classes, including Background, Vehicle, Pedestrian and Cyclist. The neurons in the first learnable layer are connected with 16 neurons in the next layer, after training, there will be 64 learnable weights that shown as pink arrows in Figure 4.2. And the weights generated by the additional 4 neurons are shown in black. At the beginning of training process, the first 4 input neurons of PCP has the same values as CP. In another word, the weights in pre-trained model can therefore be applied to initialize PCP. The weights for the last 4 neurons are initialized as Gaussian random variables with zero mean and a fairly small standard deviation (std). Since the input lidar points are painted with image information, PCP network is about to learn the information both from lidar and image. Therefore, after a number of epochs, the sensor fusion model can be obtained.

4.3 Evaluation

During evaluation, Average Precision (AP) is one of the metrics to compare the performance of different models, which is obtained for each class (Vehicle, Pedestrian and Cyclist) with 3 difficulty levels (Easy, Moderate and Hard). The definition of difficulty levels is related to how much objects are occluded and truncated. The more occluded, the more difficult to correctly detect.

IOU is a way to evaluate the accuracy of object localization and judges how similar

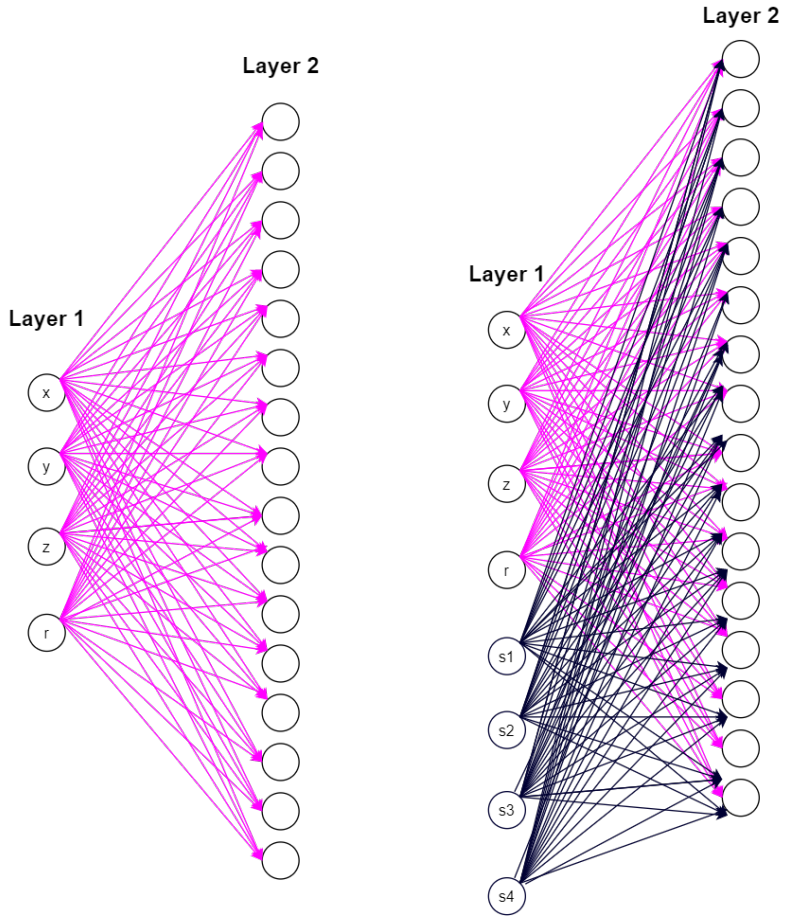


Figure 4.2: The left figure is the first and second learnable layers of CenterPoint. The first four neurons represent the coordinates x, y, z and intensity r of a lidar point. And weights related to these four neurons are pink. These pink branches contains the pre-trained weights of CenterPoint. The right figure shows the first and second learnable layers of Painted CenterPoint (our fusion model). The last four neurons represent the additional channels for segmentation scores $s1, s2, s3, s4$. The initialization weights related to these four neurons are black.

the predicted b-box to ground truth. It is calculated as the formula 4.3. According to the convention of Computer Vision Law [56], IOU is usually judged by threshold 0.5, or even higher in order to obtain accurate predicted b-boxes. In this thesis, the thresholds for Vehicle, Pedestrian and Cyclist are respectively set as 0.7, 0.5, 0.5. It means that, for instance, an object is detected as Vehicle only if the IOU between its predicted bounding box and ground truth is greater or equals to 0.7.

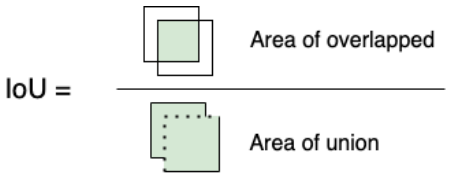


Figure 4.3: IoU calculation formula.

KITTI provides 4 evaluation metrics: 2D Bounding Box (b-box), BEV, 3D b-box and Average Orientation Similarity (AOS).

- **2D b-boxes:** the predicted 3D b-boxes are firstly projected to camera view. Then this metric calculates the IOU between mapped predicted b-boxes and ground-truth 2D b-boxes in 2D.
- **BEV:** the predicted 3D b-boxes and ground-truth 2D b-boxes are mapped to BEV. Then the IoU is calculated in BEV.
- **3D b-box:** this metric directly calculates the IoU between predicted 3D b-boxes and ground-truth 3D b-boxes in 3D space.
- **AOS:** this metric measures the similarity of direction between the predicted b-boxes and ground-truth b-boxes.

The last metric, AOS, is mostly used to evaluate performance of object tracking, therefore, only the first three metrics are mainly discussed in this thesis. The KITTI evaluation metrics above are given in the form of the average precision (AP) of detection results. Additionally, to better compare the performance between different strategies, other metrics are also introduced into this thesis:

- **mean Average Precision (mAP):** since the evaluation metrics will give us AP for each class of a certain difficulty level, it is hard to conclude how each class performs intuitively. In order to comprehensively compare the performance among classes, the mean Average Precision (mAP) for each class is used, which is obtained by $mAP = \frac{\sum_{i=1}^{C} AP_i}{C}$, where $C = 3$ represents for tasks of 3 difficulty levels: easy, moderate and hard. And AP_i is the AP for each difficulty level of a certain class. Then we get mAP for each class and compare them.
- **P-R curve:** due to the classes of objects may imbalanced distribute in the input resources, Precision-Recall (P-R) curve is a great tool to measure the correct prediction without effect of class imbalance. Precision is also called Positive Predictive value calculated as Equation 4.4. The recall in the Equation 4.5, reflects the sensitivity of the model [57]. According to their definitions, precision considers the the relevancy of the results, instead, recall highlights how many objects are correctly detected [57].

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{4.4}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{4.5}$$

  Generally, higher precision and recall values are expected. For instance, if the model has low precision but with high recall values, it means that there are many objects are detected and returned with predicted classes. However, these predicted classes are mostly incorrect regarding ground truth. Therefore,

a desired model must have high precision and high recall, which means there are many of the objects are correctly detected with the same class as ground truth. That is to say, the larger Area Under Curve (AUC), the better the model is.

Consequently, P-R curves would be another intuitive method to check the model's performance. In this thesis, we set 40 thresholds, and for each threshold, the corresponding precision and recall are calculated. After plotting all these 40 results in (P, R) pairs, the P-R curve is achieved. Further discussion will be given in Chapter 6.

4.4 Implementation framework

This thesis is completed by Python, Pytorch and the machine learning library from TensorFlow. The results are saved in log files and written into the TensorBoardX to intuitively check the continuous changes of related metrics during the training process. The training and evaluation are performed on the GeForce RTX 3090 model of GPU to accelerate the training process. Moreover, we train and evaluate the outputs based on CenterPoint model and three Painted CenterPoint models with either DeepLabV3, DeepLabV3+ or HMA segmentation methods. We will show and discuss the results in Chapter 5 and 6.

5

Experiment Results

In this chapter, we show output results obtained through experiments. The results, which is evaluated on KITTI, will be firstly compared with other models shown in baseline Table 5.2. It is followed by the intuitive histogram and P-R curves. Then, to further investigate the performance of CenterPoint and all three Painted CenterPoint methods, the mAP (%) results are computed with different aspects of classes, difficulties and metrics. The visualization results in different scenes will be demonstrated as well. Additionally, we also append the concrete AP results in each different aspect for more details in Tabel A.1, which can be used for reference.

5.1 Comparison with baselines

	Metric name and value	Global rank
CenterPoint[nuScenes]	NDS=71%	No.5
	mAP=67%	No.5
CenterPoint[Waymo-all]	APH/L2=71.93%	No.1
CenterPoint[Waymo-cyc]	APH/L2=71.28%	No.1
CenterPoint[Waymo-ped]	APH/L2=71.52%	No.1

Table 5.1: The official CenterPoint results on nuScenes and Waymo datasets[58].

The CenterPoint has very well performance on nuScenes and Waymo datasets as shown in Table 5.1. Waymo has its own evaluation metrics. Average Precision Weighted by Heading (APH) is similar as AP but it also considers orientation when computing the average precision. Here the detection results are measured in APH and L2 (Hard) difficulty. CenterPoint ranks global top 1 in 3D Object Detection task, outperforming all published methods on Waymo datasets. Additionally, NuScenes Detection Score (NDS) is created by nuScenes team, which comprehensively considers the performance of model in translation, velocity, scale and attribute estimation. And CenterPoint also ranks top 5 on nuScenes dataset [58]. Considering our limited computer resources and time consumption, Waymo and nuScenes are too large to be implemented. KITTI is simple and has good calibration capabilities so we use KITTI in this thesis. Whereas, there are no KITTI pre-trained models are provided by CenterPoint team. Therefore, we should train a reasonably comparable model by ourselves.

mAP(%)	**Modality**	**Vehicle(0.7)**	**Pedestrian(0.5)**	**Cyclist(0.5)**
PointPillar	Lidar[voxel-based]	77.28	52.29	62.68
SECOND	Lidar[voxel-based]	78.62	52.98	67.15
PointRCNN	Lidar[point-based]	78.70	54.41	72.11
Part-A^2-Free	Lidar[voxel-based]	78.72	65.99	74.29
Part-A^2-Anchor	Lidar[voxel-based]	79.40	60.05	69.90
PV-RCNN	Lidar[point&voxel-based]	83.61	57.90	70.47
CaDDN	Images	21.38	13.02	9.76
CenterPoint	Lidar[voxel-based]	76.9766	48.7677	60.9019
CenterPoint + DeepLabV3	Lidar&Images	76.9073	47.1765	59.1901
CenterPoint + DeepLabV3+	Lidar&Images	77.0933	49.5723	61.2767
CenterPoint + HMA	Lidar&Images	76.8278	51.3317	66.1074

Table 5.2: Comparison of our results against the baselines provided by OpenPCDet [59]. The results are all evaluated on KITTI with 3D b-box metrics and moderate difficulty. The IOU thresholds for Vehicle, Pedestrian and Cyclist are respectively 0.7, 0.5 and 0.5. Note that the Part-A^2 has with or without anchors application.

In this thesis, OpenPCDet is utilized as the framework of CenterPoint, which is an open source lidar detector framework giving a commonly used platform that can compile a lot of different algorithms [59]. The results of this thesis are therefore able to compare with baselines provided by OpenPCDet, as seen in Table 5.2. Each baseline method within that Table is trained over 7000 epochs by 8 GTX 1080Ti GPUs and the final result is evaluated on a fully-trained model. As introduced in Section 4.2.3, this thesis applies VoxelNet as 3D backbone. CP model is trained and evaluated with 500 epochs on a GeForce RTX 3090 GPU. Since we are limited by the training resources and time, we cannot achieve the perfectly fully-trained model as baselines. In this thesis, the model at epoch 500 is considered a reasonable fully-trained model. This model is found by plotting the training and validation loss with iteration steps. Moreover, PCP mdoels uses it as pre-trained model and continues training for additional 50 epochs.

5.2 Main evaluation results

Figure 5.1 gives comprehensive performance for CenterPoint and all three Painted CenterPoint methods, where the mAP results are calculated across all the classes, difficulties and metrics. According to the figure below, all the semantic segmentation methods applied in this thesis are able to further improve the accuracy of CenterPoint. Especially, when CenterPoint considers the semantic segmentation information provided by HMA, it has the highest mAP among all the methods shown in this figure.

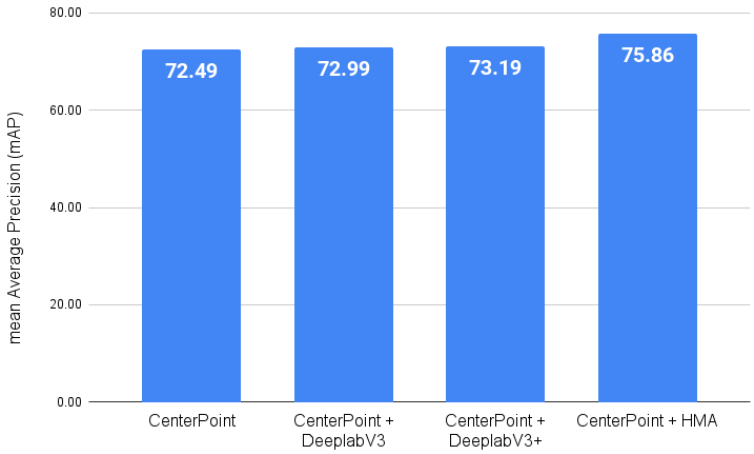


Figure 5.1: The mAP (%) results for CP and PCP methods across all the classes, difficulties and metrics.

5.2.1 P-R curves

In order to well investigate the properties of CenterPoint and Painted CenterPoint with three distinct semantic segmentation algorithms, P-R curve is a great tool that apparently indicating their capabilities. Figure 5.2, 5.3 and 5.4 intuitively illustrate the performance of our models. We solely present the P-R curves for different classes in specific metrics and difficulties to make the analysis in the next chapter be more targeted.

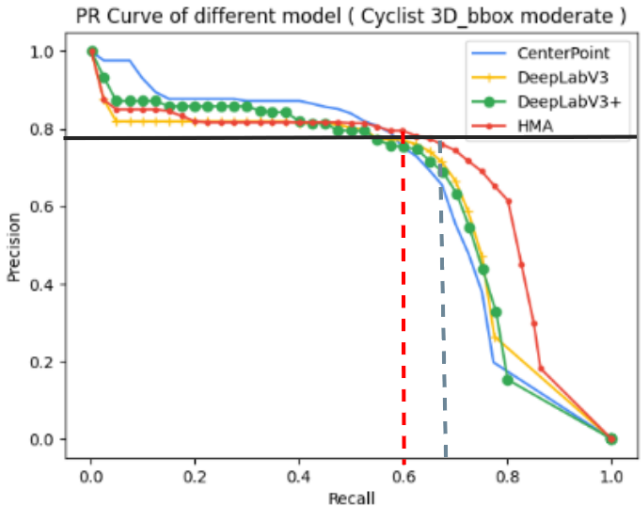


Figure 5.2: P-R curves of CenterPoint and three Painted CenterPoint models for Cyclist detection with 3D b-box evaluation metric and Moderate difficulty.

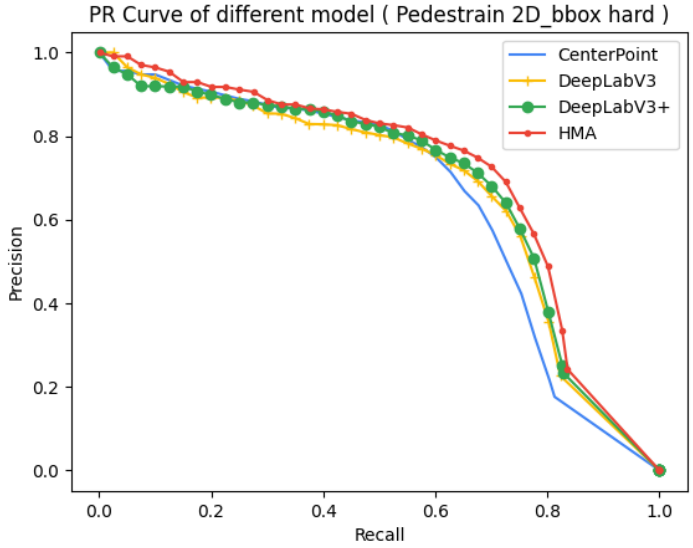


Figure 5.3: P-R curves of CenterPoint and three Painted CenterPoint models for Pedestrian detection with 2D b-box evaluation metric and Hard difficulty.

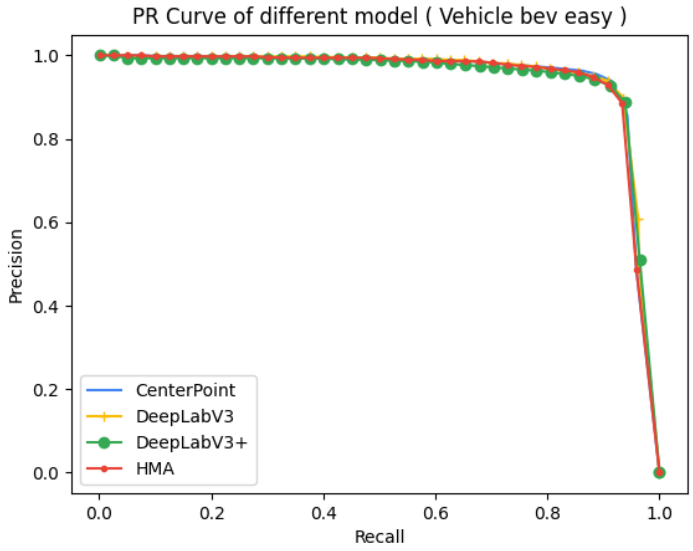


Figure 5.4: P-R curves of CenterPoint and three Painted CenterPoint models for Vehicle detection with BEV evaluation metric and Easy difficulty.

5.2.2 mAP results from different aspects

Table 5.3, 5.4 and 5.5 collect the mAP results of CenterPoint and Painted CenterPoint methods from aspects of classes, difficulties and evaluation metrics. Hence, it is possible to assess their performance from various aspects and reasonable conclusions can also be drawn.

- **mAP for classes**

	Vehicle(0.7)		**Pedestrian(0.5)**		**Cyclist(0.5)**	
CenterPoint	88.07		57.11		72.28	
CP + DeepLabV3	**89.24**	**+1.17**	56.83	-0.28	72.90	+0.62
CP + DeepLabV3+	88.71	+0.64	58.46	+1.35	72.42	+0.13
CP + HMA	89.15	+1.08	**61.27**	**+4.16**	**77.14**	**+4.86**

Table 5.3: The mAP for CenterPoint and Painted CenterPoint applied with different semantic segmentation methods across all difficulties and metrics. The green font represents the increase against CP and the black font indicates that the differences with CP is very small.

- **mAP for difficulties**

	Easy		**Moderate**		**Hard**	
CenterPoint	79.44		70.65		67.37	
CP + DeepLabV3	79.56	+0.11	71.16	+0.50	68.26	+0.89
CP + DeepLabV3+	80.00	+0.56	71.12	+0.47	68.46	+1.09
CP + HMA	**81.88**	**+2.44**	**74.43**	**+3.78**	**71.26**	**+3.89**

Table 5.4: The mAP for CenterPoint and Painted CenterPoint implemented with different semantic segmentation methods across all classes and metrics.

- **mAP for metrics**

	2D b-box		**BEV**		**3D b-box**	
CenterPoint	78.49		69.95		64.42	
CP + DeepLabV3	80.17	+1.68	69.89	-0.07	63.18	-1.24
CP + DeepLabV3+	80.02	+1.53	70.10	+0.15	64.39	-0.03
CP + HMA	**82.82**	**+4.34**	**72.70**	**+2.74**	**66.40**	**+1.98**

Table 5.5: The mAP for CenterPoint and Painted CenterPoint implemented with different semantic segmentation methods across all classes and difficulties. The red font represents the decrease against CP.

5.3 Visualization results

This thesis selects three completely different scenarios from KITTI, so as to check whether the conclusion drawn from above tables would be supported in real scenarios or not. For each scene, the original image would be listed in the first place. It is then followed by lidar point clouds which shows the detected objects in 3D bounding boxes with class information and related confidence scores. Output 3D bounding boxes in different colours corresponding to the class: *Vehicle* in green, *Pedestrian* in blue and *Cyclist* in yellow. The pink boxes are the ground truth boxes provided by KITTI. The red point in the middle of each box is the predicted center point.

SCENE 1

Figure 5.5: Scene 1: Camera front view.

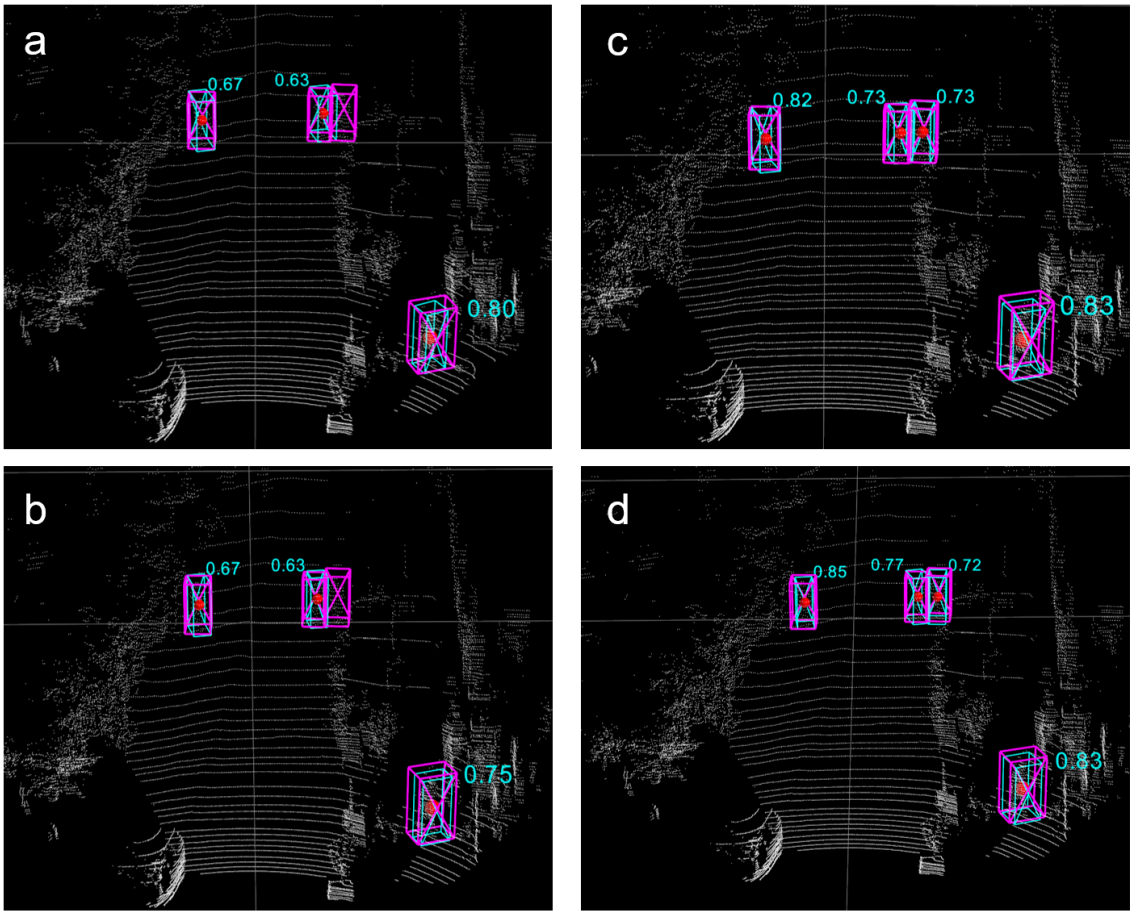


Figure 5.6: Scene 1: 3D bounding boxes and confidence scores of different models: a) CP b) PCP with DeepLabV3 c) PCP with DeepLabV3+ d) PCP with HMA

SCENE 2

Figure 5.7: Scene 2: Camera front view.

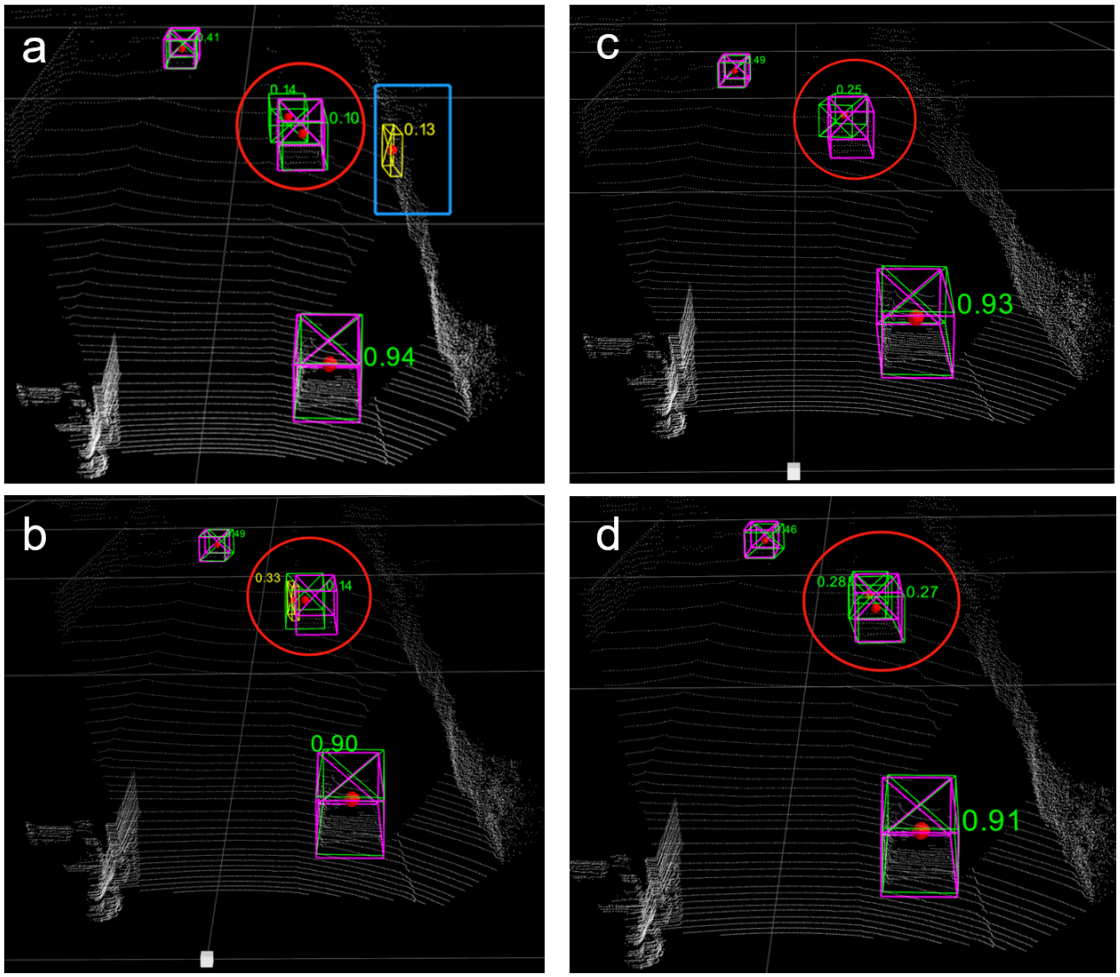


Figure 5.8: Scene 2: 3D bounding boxes and confidence scores of different models: a) CP b) PCP with DeepLabV3 c) PCP with DeepLabV3+ d) PCP with HMA

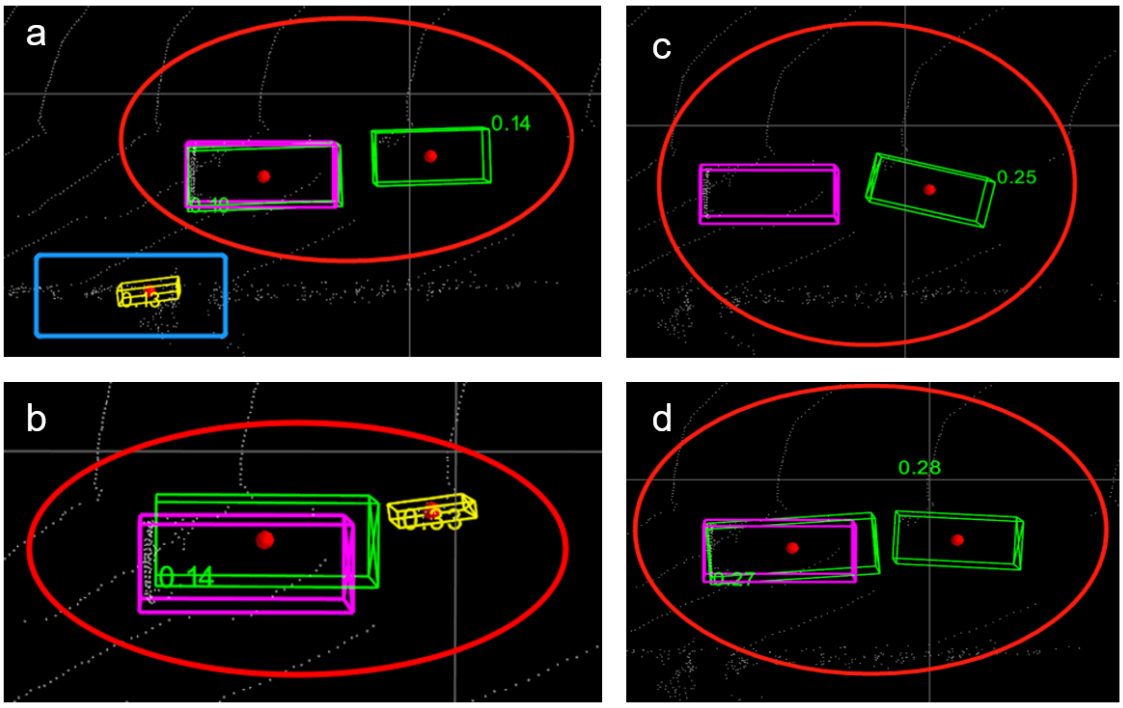


Figure 5.9: Scene 2: Top view: a) CP b) PCP with DeepLabV3 c) PCP with DeepLabV3+ d) PCP with HMA

SCENE 3

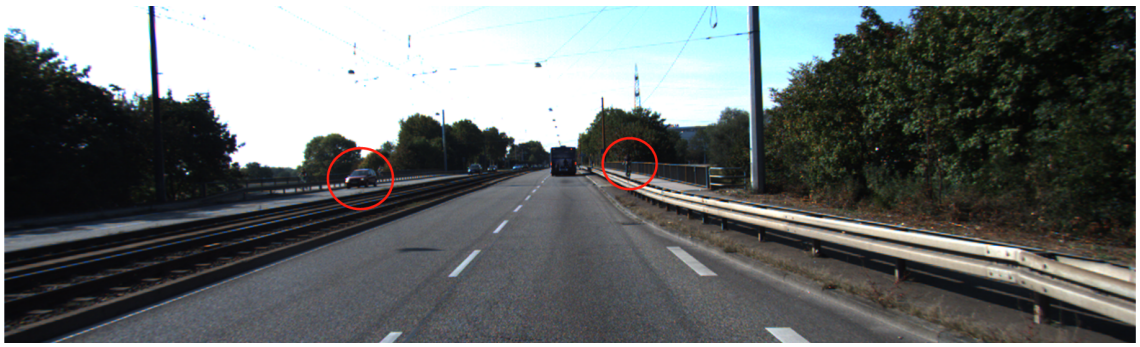


Figure 5.10: Scene 3: Camera front view.

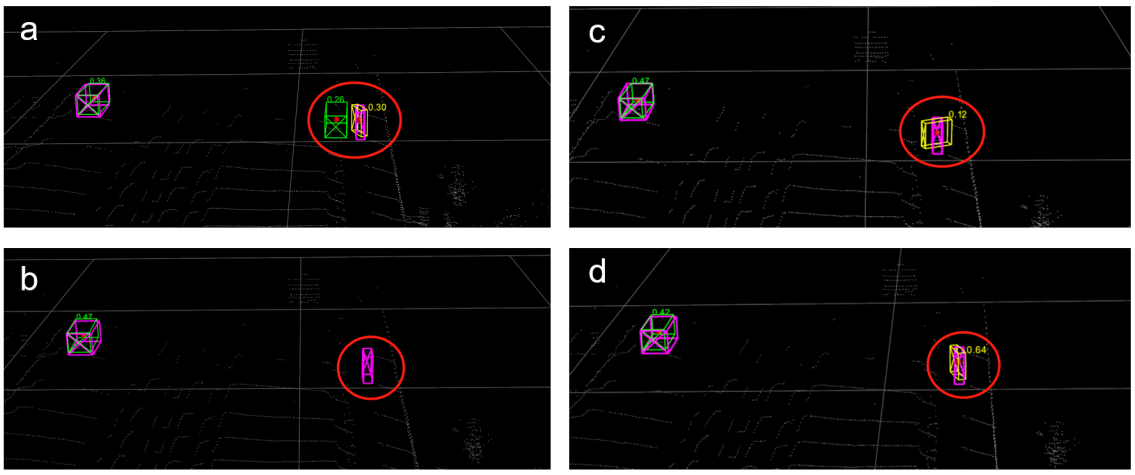


Figure 5.11: Scene 3: 3D bounding boxes and confidence scores of different models: a) CP b) PCP with DeepLabV3 c) PCP with DeepLabV3+ d) PCP with HMA

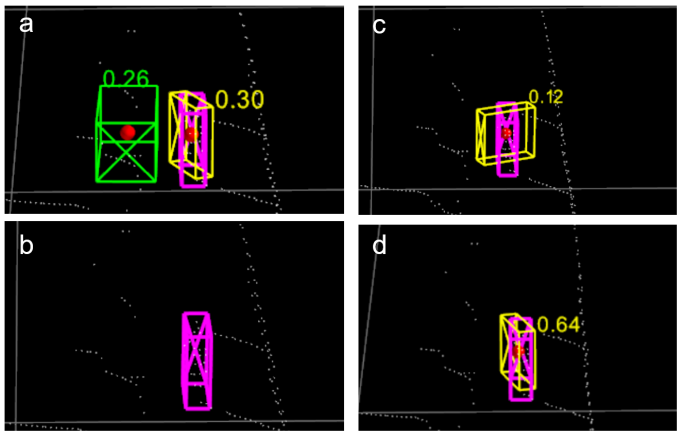


Figure 5.12: Scene 3: Zoom in the detection of the cyclist: a) CP b) PCP with DeepLabV3 c) PCP with DeepLabV3+ d) PCP with HMA

Figure 5.13: Scene 3: Semantic segmentation results with DeepLabV3.

Figure 5.14: Scene 3: Semantic segmentation results with DeepLabV3+.

Figure 5.15: Scene 3: Semantic segmentation results with HMA.

6

Discussion and future work

This chapter is going to firstly discuss the data shown in Table 5.2. Then we analyse the obtained results from aspects of segmentation methods, classes, difficulties and metrics. It is then followed by the visualization results as shown in Section 5.3. At the end, we suggest some possible future works.

6.1 Analysis with baselines

As mentioned, due to the limited time and resources, this thesis is solely able to obtain the comparable results. Although, the precision of CP shown in Table 5.2 is slightly inferior to the other methods, they are still at the same accurate level. Different segmentation method can improve the precision of CP with different extents. In Table 5.2, the segmentation information produced by DeepLabV3+ increases the precision of CP in all three classes: Vehicle, Pedestrian and Cyclist. Furthermore, HMA gives the biggest growth in Pedestrian and Cyclist classes, which are respectively 3 and 6 percentage points compared to CP model. DeepLabV3 decreases the accuracy in Pedestrian and Cyclist classes about 1 percentage. The reason is that V3 has poor segmentation quality, in another word, the segmentation scores provided by V3 is less accurate than V3+ and HMA. These results reflect that segmentation quality is crucial to improve the precision of lidar-only object detection method.

6.2 Analysis of main evaluation results

In this sections, we further discuss their performance from different aspects: classes, difficult levels and metrics.

6.2.1 Results for different classes

Table 5.3 shows the mAP of CP and PCP models in 3 classes: Vehicle, Pedestrian and Cyclist. As can be seen, the PCP models generally improve object detection precision of CP. Even though, Vehicle detection by CP has already been at a high accuracy level, nevertheless, the three segmentation methods are still able to further increase it for 1.17, 0.64 and 1.08 mAP percentage points. The detection of pedestrians and cyclists are harder than vehicles because of their various poses and shapes. And this leads to PCP models sometimes cannot further raise the detection

accuracy compared to CP. For instance, PCP model with V3 has similar Pedestrian detection accuracy to CP. The same condition also happens to PCP model with V3+ in Cyclist class. Opposite to the other segmentation methods, HMA still gives the greatest growth in Pedestrian and Cyclist classes with respect to 4.16 and 4.86 percentage points. This can be explained by the properties of HMA. Since it applies multi-scale strategy to hierarchy segment objects, it therefore has capability to segment small objects with larger scales and finally provide higher segmentation scores to promote better object detection results.

Additionally, we can better prove and analysis these results according to visualization results. In Scene 1 as shown in Figure 5.5, we mainly tests the ability to detect pedestrians in a crowded street. The ability of detecting these two persons standing together is different with different models.

As the results shown in Figure 5.6, pink boxes represent the ground truth and blue boxes show the detection of pedestrians. It can be concluded that CP performs fairly well in detecting the nearby person, which gives about 0.8 confidence score, but the precision of detection becomes worse when going farther. As can be seen in Figure 5.5, there are two pedestrians standing together in the middle. Whereas, CP fails to detect the right person, see Figure 5.6 a. The reason relates to sparse point clouds, that is to say, the two person are standing too close to be distinguished between each other only by lidar points. After adding the segmentation scores from DeepLabV3, the person on the right is still unable to be detected, shown in Figure 5.6 b. This visualization result approves the conclusion from Table 5.3, where V3 is powerless to improve performance of CP in Pedestrian detection. Nevertheless, DeepLabV3+ and HMA methods not only successfully distinguish these two pedestrians, but also give rather higher confidence scores than CP. It could be noticed that V3+ does not perfectly estimate the orientation of the left most pedestian, but the 3D b-box provides by HMA fits the ground truth well. In conclusion, the visualization results of this scene show that the precision of Pedestrian is improved even in a fairly complicated and crowded street scene based on our PCP models. HMA outperforms among the other segmentation methods, which is the same conclusion as in Table 5.3. Furthermore, this scenario also indicates that the segmentation quality is crucial in sensor fusion strategy.

6.2.2 Results for different difficulty levels

As mentioned in Chapter 4, the difficulty is defined by the level of occlusion and truncation between objects. The more occlusion between objects, the harder to be correctly detected. Regarding Table 5.4, the PCP with either V3 or V3+ does not have apparently improvement in Easy and Moderate difficulties. However, CP could benefits more from these two segmentation methods in Hard difficulty. This makes sense because it is hard to distinct and predict labels of highly occluded object only by point clouds. The limitation can be relieved by additional color and texture information. That is to say, in Easy and Moderate cases, the added image information may not help too much when point clouds can provide enough information to give good detection result while it would provide big improvements

when point clouds cannot work in Hard difficulty. HMA method still gives the greatest growth, especially in difficult tasks. As mentioned in section 3.2.3, HMA is an attention-based method, which means that it aims to deal with specific failures with certain scales. Furthermore, it has the ability to predict relative attention through its network. These properties can explain the outstanding performance of PCP with HMA shown in this table.

Additionally, the conclusion can be proven in the following scenario shown in Figure 5.7, which is able to investigate the capability of the models detecting occluded objects.

Figure 5.8 and Figure 5.9 give the visualization results, especially, the Figure 5.9 shown from top view. As mentioned before, lidar has shortcomings to detect occluded objects. In Figure 5.9 a, the results of CP model shows that the confidence scores are fairly low for the two vehicles in the red circle. And there is an object surrounded by a blue box in Figure 5.7 with rather smaller size which is unclear. CP incorrectly detects this object as Cyclist class. This false detection can then be avoided after considering the segmentation information.

See Figure 5.9, HMA shows its potential to detect objects in more difficult tasks. After adding segmentation scores onto the point clouds, HMA improves the confidence scores for both the front and occluded vehicles compared with CP. DeepLabV3 still falsely detects the occluded object as a cyclist, instead of a vehicle. And DeepLabV3+ fails to detect these two occluded vehicles as well.

6.2.3 Results for different metrics

As introduced in Chapter 4, the 3 evaluation metrics on KITTI is distinguished by the calculation dimension of IOU. According to Table 5.5, PCP models are able to improve the accuracy of CP in 2D b-box metric. The reason is that image segmentation methods capture features in 2D image view, which is the same view as 2D b-box. Moreover, HMA is the most robust segmentation method here, therefore, it has the most improvements as before. And for 3D b-box metric, the object detection accuracy drops 1.24 percentage points after considering segmentation information from DeepLabV3. In another word, here DeepLabV3 deteriorates the precision of object detection. One possible reason is that images lack the depth information, which would not benefit to estimate depth and build 3D b-box. Additionally, DeepLabV3 provides poor segmentation scores, that further disturbing original lidar points. This reduction can be better analyzed later with visualization results.

What should be noticed is that although V3 decreases the precision in 3D b-box metric, however, it can improve the precision in 2D b-box metric. Why it has these performances? This may be interpreted with the painting step in PointPainting. As introduced in Section 3.1 Step **c. Projection**, only if the lidar points mapped to the image can be further added with segmentation scores, which are finally used for 2D object detection. However, for those points that projected out of the image range, they are assigned as Gaussian random variables with zero mean and a small standard deviation value. This step approximately introduces random disturbance

to the original lidar points, which effects the final 3D detection results. Since the random disturbance are only added to the points that out of image range, thus, there is no effect on 2D detection, but destroying results in 3D detection.

Here also shows visualization results to better analysis the data. This scene mainly demonstrates detection of a vehicle and a cyclist on a relatively empty highway (see the object in two red circles in the Figure 5.10). Figure 5.11 shows the detection results based on the 4 models. And Figure 5.12 shows the zoom-in detection of Cyclist.

As can be seen, all the four models have similar vehicle detection performance, however, they present different capability to detect cyclists. CP successfully detects the cyclist but also false detects a vehicle next to it, shown as a green b-box in Figure 5.12 a. This false detection can be prevented by introducing image information. In Figure 5.12 c and d, DeepLabV3+ and HMA are able to both detect the cyclist and cancel this false detection.

According to Figure 5.12 d, for PCP with HMA, the confidence score for the cyclist is increased from 0.3 to 0.64. This proves the conclusion drawn from Table 5.5, PCP with HMA gives the most precise results in 3D b-box detection metric among the other methods in this thesis. However, in Figure 5.12 b, V3 fails to detect the cyclist, which is even worse than CP. Why introducing image information by V3 will further degrade the results? We think it is because the segmentation quality provided by V3 is not as good as V3+ and HMA.

The segmentation results are introduced to demonstrate the reason shown in Figure 5.14 and 5.15. V3+ and HMA correctly segment the pixels with respect to Cyclist class in red. Whereas, in Figure 5.13, V3 segments the cyclist pixels in green, indicating that V3 considers it as a tree. This is definitely wrong. Because trees are regarded as background in the painting step, the pixel corresponding to the cyclist is therefore painted as background, and lidar points consequently interfere with the wrong additional image information. This explains why CP can detect the cyclist (see Figure 5.11 a), in contrast, PCP with V3 loses the detection (see Figure 5.11 b). Furthermore, the noticeable decrease shown in Table 5.5 is proofed in this scenario as well.

Thus, it can be concluded that the PCP highly depends on the quality and strength of segmentation methods. If the segmentation method has poor segmentation quality and unstable performance, such like DeepLabV3, the additional segmentation information may introduce errors to lidar points. Consequently, the detection performance may even be deteriorated after adding image information.

6.3 Analysis of P-R curves

In this section, we analysis the performances of CP and PCP models based on P-R curves.

In Figure 5.2, the red dashed line shows that the 4 models having almost the same precision when recall equals to 0.6. According to the definition of recall [60], the larger values of it means that the more labels of samples are correctly predicted. As can be seen in Figure 5.2, the blue dashed line with larger recall value indicates different performance from different models. PCP with HMA still keeps as good precision as recall equals to 0.6. In contrast, CP, DeepLabV3 and DeepLabV3+ have lower precision when recall goes up, which means more ground truth objects can be detected but most of their labels are falsely predicted.

Similar performance can be seen in Figure 5.3 that the P-R curves of Pedestrian class in Hard and 2D b-box metrics. The model applied with HMA still has the best capability among all the others. Because in P-R curve, the better model is the one has the largest area under the curve, which is just PCP with HMA. It is noticeable that the precision of CP is greatly improved by segmentation information, especially at larger recall. This can be explained that the additional image information allowing lidar point clouds to be robust and supporting more labels of objects correctly predicted. Due to the simple and common shapes of vehicles, all the models have nearly same performance for the Vehicle detection in Easy and BEV metrics, as shown in Figure 5.4. Then, the same conclusion can be drawn from P-R curves as Figure 5.1. Within the scope of this thesis, PCP models can improve the accuracy of lidar-only object detection method. Regarding the relative well segmentation quality, PCP with HMA is the most robust and accurate model compared to the other three methods.

6.4 Future work

Although the current PCP models have improved detection results, there are still possible improvements, optimizations and expanded tasks that can be done in the future.

- **Change 3D backbone of CP:** this thesis solely applies VoxelNet as the 3D backbone of CP. Point-based methods keep information and original position for all lidar points without discard so that they usually have higher precision than Voxel-based methods, but the price is high computational complexity. Therefore, it can be expected that if the 3D backbone is changed to point-based methods, the detection precision can be improved a lot.

- **Fuse CP with other segmentation methods:** here applies three basic and popular segmentation methods. The follow-up experiments can try the other SOTA segmentation methods or even the instance segmentation methods.

- **Implement on the other datasets:** because the time and computer memory are limited. CP model is trained on a simple KITTI dataset with only 3712 samples, which is not enough for obtaining ideal results. To achieve higher precision and a more robust model, one of the best attempts is to train on other datasets with diversity, like the nuScnenes dataset. That kind of experiments may give more convincing results of whether the sensor fusion model is robust

enough to provide highly accurate results in various scenes.

- **Extend to object tracking problem:** because CP is the centre-based algorithm, it has advantages to direct regress from the features at the centre location, therefore, it is able to detect objects in a shorter time, which can greatly benefit the object tracking task. Thus, it would be interesting to further test our PCP models on tracking tasks. And to see if it would still improve the performance as in the object tracking.

7

Conclusion

This thesis proposes a deep learning sensor fusion object detection method, which is named as Painted CenterPoint. It successfully improves the performance of the SOTA lidar object detection method, CenterPoint, by introducing camera information which is done by painting/adding the segmentation scores onto lidar point clouds. This thesis demonstrates that it is potential to improve lidar-based object detection methods also considering segmentation information, like the PCP method. The final evaluation results have shown that our PCP models based on different segmentation methods can may improve or degrade the performance of CP in different cases.

According to the results regarding different classes, it can be concluded that colour and texture information are crucial to improve object detection performance. Even though the Vehicle detection has already been at a high level, adding camera information is still able to slightly improve its precision. While for classes with smaller size of objects, like Pedestrian and Cyclist, it may challenge the capability of lidar-only based object detection methods, like CP, so adding camera information would benefit more for these two classes.

PCP is also able to improve the performance of CP in different difficulties. By adding V3 and V3+ camera information, the precision of CP can be slightly increased in specific difficult levels. HMA gives the greatest improvements in all difficulties, especially in Hard tasks.

For different metrics, PCP based on these three segmentation methods are all superior to CP for the 2D b-box metric. However, for BEV and 3D b-box metrics, PCP with HMA still improves performance while PCP with V3 and V3+ may not make a big improvement and even degrade the performance.

Within the scope of this thesis, the additional information from cameras can enhance the performance but may also deteriorate the performance. It is strongly related to the quality of segmentation. HMA has the capability to predict and provide the most precise segmentation scores, compared with DeepLabV3 and V3+ models. Consequently, Painted CenterPoint (PCP) with HMA provides the best detection results among all the cases. Thus, we verify that the segmentation methods would differ how PointPainting benefits the CenterPoint detector in different metrics and difficulty levels (scenarios).

Above all, according to the results obtained in this thesis, additional camera information may improve or deteriorate the performance of lidar object detection methods, that related to the quality and capability of segmentation methods. If a method only provides poor segmentation quality, such like DeepLabV3, inaccurate information is added on CenterPoint, which sometimes interferes the detection performance. Therefore, it is necessary to circumvent these possible errors introduced by segmentation methods when implementing sensor fusion strategy.

Bibliography

[1] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," 2013.

[2] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "Nuscenes: A multimodal dataset for autonomous driving," *arXiv:1903.11027*, 2020.

[3] Sourabh Vora, Alex H. Lang, Bassam Helou, Oscar Beijbom, "Pointpainting: Sequential fusion for 3d object detection," *arXiv:1911.10150*, 2019.

[4] Tianwei Yin, Xingyi Zhou, Philipp Krähenbühl, "Center-based 3d object detection and tracking.," https://github.com/tianweiy/CenterPoint, 2020.

[5] "Neurips 2020 nuscenes 3d detection challenge," https://www.nuscenes.org/object-detection?externalData=all&mapData=all&modalities=Any.

[6] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia., "Std: Sparse-to-dense 3d object detector for point cloud," *arXiv:1907.10471*, 2019.

[7] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv:1706.05587*, 2017.

[8] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," *arXiv:1802.02611*, 2018.

[9] Andrew Tao, Karan Sapra, Bryan Catanzaro, "Hierarchical multi-scale attention for semantic segmentation," *arXiv:2005.10821*, 2020.

[10] "Deep learning," https://en.wikipedia.org/wiki/Deep_learning.

[11] "Artificial neural network," https://en.wikipedia.org/wiki/Artificial_neural_network.

[12] "Convolutional neural network," https://en.wikipedia.org/wiki/Convolutional_neural_network.

[13] "Object detection," https://en.wikipedia.org/wiki/Object_detection.

[14] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, Xindong Wu, "Object detection with deep learning: A review," *arXiv:1807.05511*, 2019.

[15] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *arXiv:1506.01497*, 2016.

[16] "Joint haar-like features for face detection," https://www.researchgate.net/publication/4193945_Joint_Haar-like_features_for_face_detection.

[17] "Histogram of oriented gradients," https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients.

[18] X. Zhou, D. Wang, P. Krähenbühl, "Objects as points," *arXiv:1904.07850*, 2019.
[19] "Image segmentation," https://en.wikipedia.org/wiki/Image_segmentation.
[20] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He, "Detectron," https://github.com/facebookresearch/detectron, 2018.
[21] N. V. Zhaowei Cai, "Cascade r-cnn: Delving into high quality object detection," *arXiv:1712.00726*, 2019.
[22] R. C. Vijay Badrinarayanan Alex Kendall, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *arXiv:1511.00561*, 2015.
[23] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," *arXiv: 1412.7062*, 2016.
[24] ——, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *arXiv:1606.00915*, 2017.
[25] Y. Li, H. Zhao, X. Qi, L. Wang, Z. Li, J. Sun, and J. Jia, "Fully convolutional networks for panoptic segmentation," *arXiv:2012.00720*, 2021.
[26] "Radar," https://en.wikipedia.org/wiki/Radar.
[27] "Data set," https://en.wikipedia.org/wiki/Data_set.
[28] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," 2012.
[29] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, S. Zhao, S. Cheng, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," 2020. arXiv: 1912.04838 [cs.CV].
[30] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015. arXiv: 1405.0312 [cs.CV].
[31] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," 2016. arXiv: 1604.01685 [cs.CV].
[32] Ross Girshick, "Fast r-cnn," *arXiv:1504.08083*, 2015.
[33] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, "You only look once: Unified, real-time object detection," *arXiv:1506.02640*, 2015.
[34] Shaoshuai Shi, Xiaogang Wang, Hongsheng Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," *arXiv:1812.04244*, 2019.
[35] "Voxel," https://en.wikipedia.org/wiki/Voxel.
[36] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," *arXiv:1711.06396*, 2018.
[37] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, Oscar Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," *arXiv:1812.05784*, 2018.
[38] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," 2017. arXiv: 1612.00593 [cs.CV].

[39] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network," *arXiv:1907.03670*, 2020.

[40] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," *arXiv:1611.07759*, 2017.

[41] R. C. Thomas Roddick Alex Kendall, "Orthographic feature transform for monocular 3d object detection," *arXiv:1811.08188*, 2018.

[42] Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving," *arXiv:1906.06310*, 2020.

[43] Alessandro Giusti, Dan C. Cireşan, Jonathan Masci, Luca M. Gambardella, Jürgen Schmidhuber, "Fast image scanning with deep max-pooling convolutional neural networks," *arXiv:1302.1700*, 2013.

[44] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, Alan L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *arXiv:1606.00915*, 2017.

[45] "Pascal voc 2012 test," https://paperswithcode.com/sota/semantic-segmentation-on-pascal-voc-2012.

[46] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, "Mask r-cnn," *arXiv:1703.06870*, 2018.

[47] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg, "Ssd: Single shot multibox detector," *arXiv:1512.02325*, 2016.

[48] Xingyi Zhou, Arjun Karpur, Linjie Luo, Qixing Huang, "Starmap for category-agnostic keypoint and viewpoint estimation," *arXiv:1803.09331*, 2018.

[49] S. K.He X.Zhang and J.Sun, "Deep residual learning for image recognition," *arXiv:1512.03385*, 2015.

[50] E. F.Yu D.Wang and T.Darrell, "Deep layer aggregation," *arXiv:1707.06484*, 2017.

[51] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," 2013.

[52] "Coco2017," https://paperswithcode.com/dataset/coco.

[53] "Deeplabv3plus-resnet101," https://github.com/VainF/DeepLabV3Plus-Pytorch.

[54] "Cityscapes," https://paperswithcode.com/dataset/cityscapes.

[55] "Mapillary," https://www.mapillary.com/datasets.

[56] "Jaccard index," https://en.wikipedia.org/wiki/Jaccard_index.

[57] "Precision and recall," https://en.wikipedia.org/wiki/Precision_and_recall.

[58] "Centerpoint 3d object detection and tracking ranks," https://paperswithcode.com/paper/center-based-3d-object-detection-and-tracking.

[59] "Openpcdet," https://github.com/open-mmlab/OpenPCDet.

[60] "Precision and recall," https://en.wikipedia.org/wiki/Precision_and_recall.

A

Appendix 1

A.1 Overall results on KITTI

		$V_e(0.7)$	$V_m(0.7)$	$V_h(0.7)$	$P_e(0.5)$	$P_m(0.5)$	$P_h(0.5)$	$C_e(0.5)$	$C_m(0.5)$	$C_h(0.5)$
2D bbox	CenterPoint	95.3951	91.1254	88.8159	70.5053	66.0364	62.8913	89.3989	73.0215	69.1764
	CP+DeepLabV3	97.5752	**92.153**	**91.3741**	70.2661	67.2461	64.2701	90.1692	76.1313	72.3455
		+2.1801	**+1.0276**	**+2.5582**	-0.2392	+1.2097	+1.3788	+0.7703	+3.1098	+3.1691
	CP+DeepLabV3+	95.8068	92.0668	91.2366	72.0775	68.0289	65.8595	90.9653	73.9359	70.1837
		+0.4117	+0.9414	+2.4207	+1.5722	+1.9925	+2.9682	+1.5664	+0.9144	+1.0073
	CP+HMA	**97.9142**	92.059	90.9101	**75.031**	**71.437**	**68.3502**	**92.2981**	**80.9215**	**76.4768**
		+2.5191	+0.9336	+2.0942	**+4.5257**	**+5.4006**	**+5.4589**	**+2.8992**	**+7.9**	**+7.3004**
BEV	CenterPoint	92.2144	87.7183	85.3917	56.1032	52.0919	48.1709	80.5586	65.5971	61.7329
	CP+DeepLabV3	**93.9148**	**87.9175**	85.4275	54.4492	51.1673	46.9592	81.8648	65.3718	61.9097
		+1.7004	**+0.1992**	+0.0358	-1.654	-0.9246	-1.2117	+1.3062	-0.2253	+0.1768
	CP+DeepLabV3+	91.8296	87.8779	**85.4361**	56.5913	52.1467	48.9079	82.1789	64.4485	61.5006
		-0.3848	+0.1596	**+0.0444**	+0.4881	+0.0548	+0.737	+1.6203	-1.1486	-0.2323
	CP+HMA	92.2271	87.6575	85.1121	**59.2568**	**55.2226**	**52.0668**	**83.6074**	**71.6038**	**67.5063**
		+0.0127	-0.0608	-0.279	**+3.1536**	**+3.1307**	**+3.8959**	**+3.0488**	**+6.0067**	**+5.7734**
3D bbox	CenterPoint	87.5996	78.2554	75.6438	50.1229	46.8903	42.9283	**79.1182**	61.62	57.6034
	CP+DeepLabV3	88.0413	78.2587	75.487	49.2178	45.2663	41.6259	74.5992	59.3089	56.7886
		0.4417	+0.0033	-0.1568	-0.9051	-1.624	-1.3024	-4.519	-2.3111	-0.8148
	CP+ DeepLabV3+	87.1103	**78.5342**	**75.87**	52.2817	47.1795	43.7192	76.6941	60.813	57.3327
		-0.4893	+0.2788	+0.2262	+2.1588	+0.2892	+0.7909	-2.4241	-0.807	-0.2707
	CP+HMA	**89.4324**	78.4115	75.6069	**53.306**	**49.5683**	**45.8425**	77.3186	**66.123**	**61.9527**
		+1.8328	+0.1561	-0.0369	**+3.1831**	**+2.678**	**+2.9142**	-1.7996	**4.503**	**+4.3493**
AOS	CenterPoint	95.33	90.85	88.48	67.71	62.56	59.33	89.26	72.06	68.27
	CP+DeepLabV3	97.55	**92.01**	**91.16**	67.27	63.9	60.35	89.76	75.13	71.44
		+2.22	**+1.16**	**+2.68**	-0.44	+1.34	+1.02	+0.50	+3.07	+3.17
	CP+DeepLabV3+	95.79	91.92	91	68.7	64.26	61.74	90.02	72.21	68.71
		+0.46	+1.07	+2.52	+0.99	+1.70	+2.41	+0.76	+0.15	+0.44
	CP+HMA	**97.89**	91.9	90.68	**72.23**	**68.12**	**64.85**	**92.07**	**80.13**	**75.73**
		+2.56	+1.05	+2.20	**+4.52**	**+5.56**	**+5.52**	**+2.81**	**+8.07**	**+7.46**

Table A.1: The detail results obtained in this thesis are shown according to different classes, difficulty levels, and metrics. The data in this table are generated with 40 Recall positions. The results with green font indicates that the improved Average Precision comparing to original CenterPoint. While the red font means the reduced AP. The bold results are the highest precision in that difficulty of class for the specific metric.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY