Doctor of Philosophy

LiDAR-Camera-Based High-Performance 3-D Object Detection for Autonomous Driving

The Graduate School

of the University of Ulsan

Department of Electrical, Electronic and Computer Engineering

Li-Hua Wen

LiDAR-Camera-Based High-Performance 3-D Object Detection for Autonomous Driving

Supervisor: Kang-Hyun Jo

A Dissertation

Submitted to
the Graduate School of the University of Ulsan
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

by

Li-Hua Wen

Department of Electrical, Electronic and Computer Engineering
University of Ulsan

June, 2021

LiDAR-Camera-Based High-Performance 3-D Object Detection for Autonomous Driving

This certifies that the dissertation

of Li-Hua Wen is approved:

Committee Chair Prof. Hee-Jun Kang

Committee Member and Supervisor Prof. Kang-Hyun Jo

Committee Member Prof. Young-Soo Suh

Committee Member Prof. Mun-Ho Jeong

Committee Member Prof. Hyun-Deok Kang

Department of Electrical, Electronic and Computer Engineering

University of Ulsan, South Korea

June, 2021

"Those who can imagine anything, can create the impossible."

Alan Turing

Acknowledgements

I would like to express my gratitude for my supervisor, Professor Kang-Hyun Jo, who gave me an opportunity to study under his guidance, nurture, encourage, and give abundant support during my life as a graduate student at the University of Ulsan. I would also thank the members of the thesis committee Prof. Hee-Jun Kang, Prof. Young-Soo Suh, Prof. Mun-Ho Jeong, and Prof. Hyun-Deok Kang for their invaluable comments and suggestion to improve the quality of this thesis.

I am grateful to the members of the Intelligent Systems Laboratory for the meaningful discussion and lesson that enhance my knowledge and broaden my understanding in this research field, computer vision, and machine learning. Much thanks for the Brain Korea Scholarship program that allow me to attend various conferences around the world and expand my horizon as a student and researcher. Much thanks to the China Scholarship Council sponsors me to pursue my dream and get my Ph.D. Last but not least, special thanks to my parents, my family, who fully support me when I study in Korea.

UNIVERSITY OF ULSAN

ABSTRACT

Graduate School of the University of Ulsan

Department of Electrical, Electronic and Computer Engineering

Doctor of Philosophy

LiDAR-Camera-Based High-Performance 3-D Object Detection for Autonomous Driving

by Li-Hua Wen

With the rapid development of autonomous vehicles, three-dimensional (3-D) object detection has become more important, whose purpose is to perceive the size and accurate location of objects in the real world. Currently, an intelligent car is equipped with at least one LiDAR apparatus, one radar and one RGB camera. Note that radar is now widely used in companies, however, only a few researchers use it to validate a new algorithm. Hence, our works focus on LiDAR and camera sensors for 3D object detection. LiDAR is employed to collect the surrounding 3-D data, referred to as a point cloud, and the camera is used to capture a high-resolution RGB image. The two devices provide two important and different types of data. However, it is non-trivial to highly efficiently and quickly extract and fuse the features of the point cloud and RGB image for high-performance 3-D object detection.

The work on this manuscript focus on the tasks of detecting 3-D objects with deep learning methods. First, we revisit the related works for LiDAR-Camera-based 3-D object detection (Chapter 2). Second, three attention mechanisms (Chapter 3) are used to enhance the global and local representative features. Third, we propose to fuse LiDAR and camera features in an early stage to do 3-D object detection (Chapter 4).

In Chapter 2, we first revisit the related works: the networks, the frameworks for object detection, the fusion methods for multi-sensor 3-D object detection, and the related dataset and metrics.

In Chapter 3, this thesis presents a novel one-stage 3D object detection framework based on three-attention mechanisms, called TAO3D, which takes raw point cloud and RGB image as inputs. Three attention mechanisms are used to obtain discriminative features. First, the height attention (HA) mechanism is introduced as an auxiliary attention module before the RGB image is fed into a network. Second, a global feature attention (GFA) mechanism models the long-range dependencies in the channel and spatial dimensions simultaneously at the feature extraction phase. Finally, a region of interest attention (RA) mechanism weights RGB image ROIs and BEV ROIs using two learnable parameters.

In Chapter 4, this thesis first presents an early-fusion method to exploit both LiDAR and camera data for fast 3D object detection with only one backbone, achieving a good balance between accuracy and efficiency. Specifically, it proposes a novel point-wise fusion strategy between point clouds and RGB images. The proposed method directly extracts pointwise features from the raw RGB image based on the raw point cloud first. Then, it fuses the two pointwise features and feeds them into a 3D neural network. The structure has only one backbone to extract features, making the proposed model much faster than state-of-the-art LiDAR and camera fusion methods.

The presented methods achieve a new breakthrough in terms of both accuracy and speed on the KITTI 3-D object detection benchmark suite.

Contents

List of Figures

List of Tables

List of Abbreviations

RGB	**R**ed **G**reen **B**lue
BEV	**B**ird's **E**ye **V**iew
ResNet	**Res**idul **Net**work
ROI	**R**egion **O**f **I**nterest
FPS	**F**rames **P**er **S**econd
3-D	**T**hree- **D**imensional
BN	**B**atch-**N**ormalization
SSD	**S**ingle **S**hot **D**etector
ReLU	**Re**ctified **L**inear **U**nit
VGA	**V**ideo **G**raphic **A**rray
MLP	**M**ulti-**L**ayer **P**erceptron
GPU	**G**raphic **P**rocessing **U**nit
IoU	**I**ntersection **O**ver **U**nion
mAP	**m**ean **A**verage **P**recision
VGG	**V**isual **G**eometry **G**roup
FPN	**F**eature **P**yramid **N**etwork
RPN	**R**egion **P**roposal **N**etwork
NMS	**N**on-**M**aximum **S**uppression
CNN	**C**onvolution **N**eural **N**etwork
LiDAR	**L**ighting **D**etection and **R**anging
Adam	**Ada**ptive **m**oment estimation algorithm

Dedicated to
my family

Chapter 1

Introduction

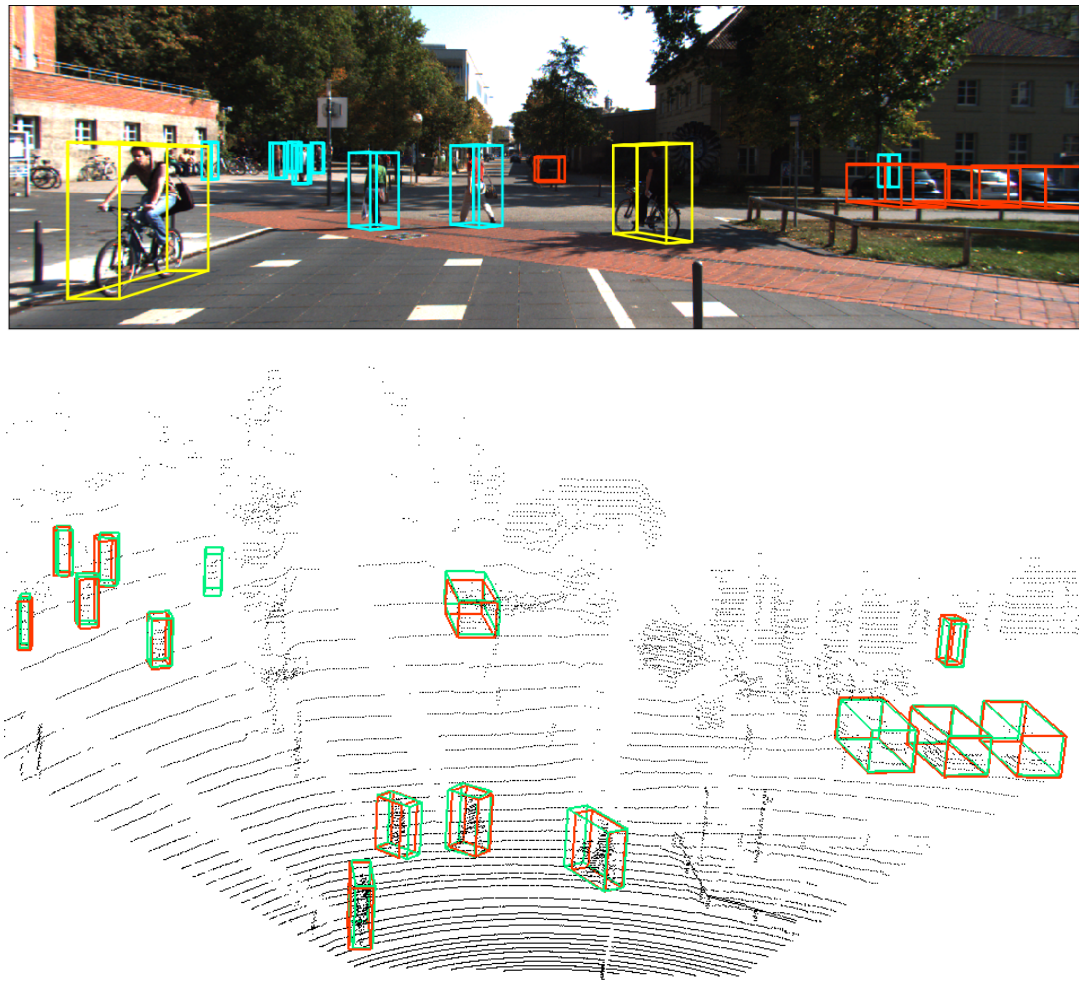


FIGURE 1.1: Visualization of 3-D object detection in an RGB image and a point cloud. In the RGB image, the red, yellow, and cyan colors denote the detection results for cars, cyclists, and pedestrians, respectively. In the point cloud, the red and cyan colors represent the predictions and ground truths, respectively.

1.1 Motivation and Background

With the rapid development of the economy and society, unmanned vehicles and robotics are born in accordance with the trend. Environment perception, especially three-dimensional environment perception, is the most important research topic in intelligent systems. 3-D object detection has lately attracted increasing attention since it can perceive the size and accurate location of objects in the real world. With the rapid development of 3-D acquisition technologies, 3-D sensors are becoming increasingly available and affordable, including various types of 3-D scanners, LiDARs, and RGB-D cameras (such as Kinect, RealSense and Apple depth cameras). Thanks to the release of a large number of LiDAR data, refer as point clouds, LiDAR data is widely adopted to do academic research. For the open-source datasets, please see Table 1.1 for details.

TABLE 1.1: The details of widely used datasets for autonomous vehicles.

Dataset	Year	Scenes	Classes	Frames	3D boxes	Night/Rain	Sensors
KITTI	2012	22	8	15K	200K	Yes/Yes	RGB & LiDAR
SUN RGB-D	2015	47	37	5K	65K	-/-	RGB-D
Cityscapes	2016	-	30	25K	0	No/No	RGB
BDD100K	2017	100K	10	100K	0	Yes/Yes	RGB
ScanNetV2	2018	1.5K	18	-	-	-/-	RGB-D & Mesh
ApolloScape	2018	-	8-35	144K	70K	Yes/No	RGB & LiDAR
KAIST	2018	-	3	8.9K	0	Yes/No	RGB & LiDAR
H3D	2019	160	8	27K	1.1M	No/No	RGB & LiDAR
Argoverse	2019	113	15	22K	993K	Yes/Yes	RGB & LiDAR
Lyft L5	2019	366	9	46K	1.3M	No/No	RGB & LiDAR
A*3D	2019	-	7	39K	230K	Yes/Yes	RGB & LiDAR
nuScenes	2019	1K	23	40K	1.4M	Yes/Yes	RGB & LiDAR
Waymo Open	2020	1K	4	200K	12M	Yes/Yes	RGB & LiDAR

Currently, an intelligent car is equipped with at least one LiDAR apparatus, one radar, and one RGB camera. Note that radar is now widely used in companies, however, only a few researchers use it to validate a new algorithm. To meet the high precision of autonomous driving for 3-D object detection, this thesis focuses on 3-D object detection based on LiDAR and camera. LiDAR is employed to collect the surrounding point clouds, and the camera is used to capture a high-resolution RGB image.

1.2 Problem Description and Objective

The methods proposed in this thesis are trained and evaluated in the famous autonomous driving dataset, KITTI (Geiger, Lenz, and Urtasun, 2012). We know that LiDAR-camera-based methods outperform both LiDAR-based and camera-based methods. However, it is non-trivial to highly efficiently and quickly extract and fuse the features of the point cloud and RGB image. There are some reasons as follows:

1. As shown in Figure 1.2, LiDAR and camera use different coordinate systems. In the coordinate system of LiDAR, the x-axis points forward, the y-axis points to the left of the vehicle, and the z-axis points upward. However, in the camera's coordinate system, the x-axis points to the right of the car, the y-axis points downward, and the z-axis points forward.

2. The point cloud is sparse and irregular and possesses the 3-D information of surrounding objects. However, the RGB image has two-dimensional properties and it has rich textures of objects.

3. LiDAR and camera sensors provide two important and different types of data, usually, two different backbones are utilized to extract features from the two data. It greatly increases computation costs and makes a proposed model runs very slow.

1.3 Contributions

To overcome the above drawbacks, in Chapter 3, this thesis first presents a novel one-stage 3D object detection framework based on three-attention mechanisms, called TAO3D, which takes raw point cloud and RGB image as inputs. Three attention mechanisms are used to obtain discriminative features. First, the height attention (HA) mechanism is introduced as an auxiliary attention module before the RGB image is fed into a network. Second, a global feature attention (GFA) mechanism models the long-range dependencies in the channel and spatial dimensions simultaneously at the feature extraction phase. Finally, a region of interest attention (RA) mechanism weights RGB image ROIs and BEV ROIs using two learnable parameters.

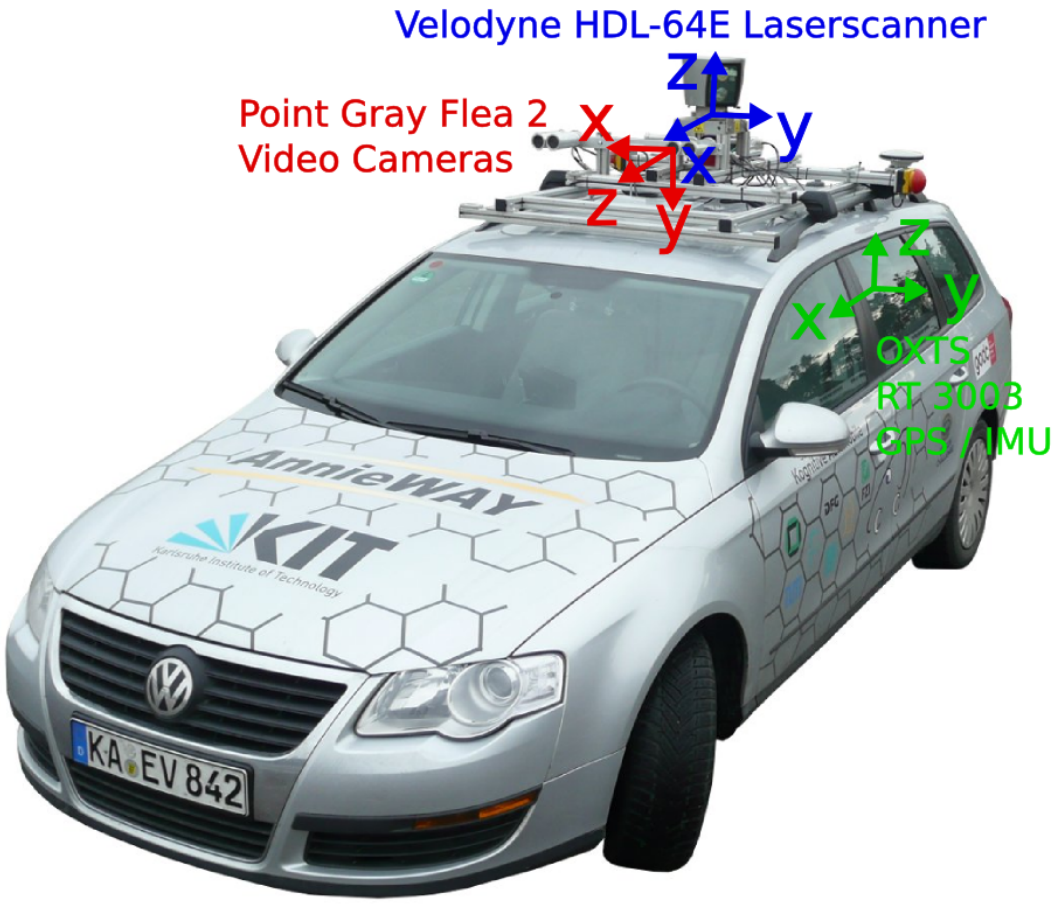


FIGURE 1.2: Details of sensor setup for KITTI vehicle. This figure is cited from KITTI website (Geiger, Lenz, and Urtasun, 2012).

The main contributions of this framework are summarized as follows:

1. It takes the raw LiDAR point clouds and RGB^D images as inputs instead of the RGB image. RGB^D images contain the height information from point clouds.

2. The GFA mechanism captures the feature dependencies in both the channels and spatial dimensions at the feature extraction stage and makes the features much more discriminative.

3. The RA mechanism weights the paired BEV ROIs and RGB image ROIs firstly and then fuses them using the addition operation. This gives more weight to important features.

In Chapter 4, this thesis proposes a novel point-wise fusion strategy between point clouds and RGB images. Firstly, the proposed method directly extracts point-wise features from the raw RGB image based on the raw point cloud. Then, it fuses

the two pointwise features and feeds them into a 3D neural network. The structure has only one backbone to extract features, making the proposed model much faster than state-of-the-art LiDAR and camera fusion methods.

The key contributions of this work are as follows:

- This paper presents an early-fusion method to exploit both LiDAR and camera data for fast multi-class 3D object detection with only one backbone, achieving a good balance between accuracy and efficiency.
- This paper proposes a highly-efficient pointwise feature fusion module, which directly extracts the RGB image point feature based on a point cloud and fuses the extracted RGB image point feature with the corresponding feature of the point cloud.
- This paper also enhances 3D object detection with an RGB^+ image, which preserves the information projected from its corresponding point cloud.

This thesis reports two kinds of framework: the one-stage and the two-stage. Massive experiments show that both the proposed methods outperform state-of-the-art LiDAR-camera-based methods on the KITTI benchmark(Geiger, Lenz, and Urtasun, 2012) both in terms of the speed and accuracy.

Chapter 2

Literature Review and Metrics

The scope in this manuscript is considering four main components: backbone network, object detection framework, fusion method, 3-D object detection. This section discusses some of the publications in these topics that influence the recent research works and consists of four parts. The first one covers a talks about some famous neural networks which usually used as the backbone of 3-D object detection. The second part discusses two main-stream frameworks for object detection. Thirdly, we report several fusion methods for LiDAR-camera-based methods. Finally, the last one presents several publications related to 3-D object detection.

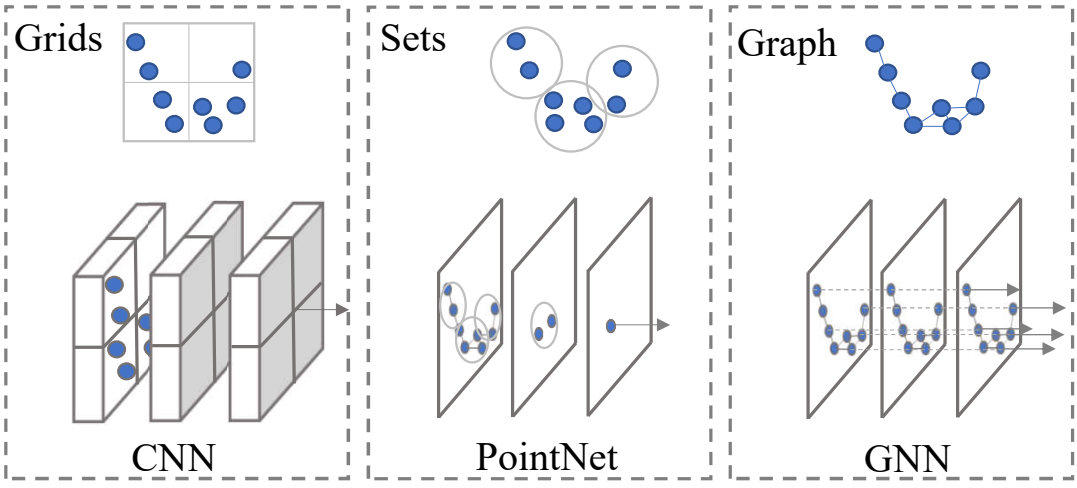


FIGURE 2.1: Three typical methods used to deal with the point cloud. From the left to right are the voxel-grid method, the point-set-based method, and the graph-based method.

Currently, there are three kinds of methods, as shown in 2.1, to extract the features from point clouds. (1) voxel-grid-based methods that either convert point clouds into 2D front view images (Chen et al., 2017), 2D bird's-eye-view (BEV) images (Li, Zhang, and Xia, 2016), or structured voxel-grid representations (Zhou and

Tuzel, 2018; Wen and Jo, 2019). Then, 2D convolutional neural networks (CNNs) or 3-D sparse CNNs are used to learn features from the converted images. (2) Point-set-based methods (Charles et al., 2017; Qi et al., 2017; Yang et al., 2020) make use of several shared multi-layer perceptrons (MLPs) to model each point in a point cloud independently and then generate a global feature using a symmetric aggregation function, such as max pooling. (3) Graph-based methods (Shi and Rajkumar, 2020) consider each point in a point cloud as a vertex of a graph and generate directed edges for the graph based on the neighbors of each point.

2.1 Network as Backbone

2.1.1 Voxel Feature Encoder and VGG network

Voxel-grid-based mthods first voxelize the point cloud into evenly spaced voxel grids and then generate a many-to-one mapping between 3D points and their corresponding voxels. Currently, there are two types of feature encoders: fixed feature encoders (Chen et al., 2017; Yang, Luo, and Urtasun, 2018) and learnable feature encoders Shi et al., 2020; Sindagi, Zhou, and Tuzel, 2019; Lang et al., 2019. Learnable feature encoders, illustrated in Figure 2.2, learn features and weight features throughout the training stage. However, fixed feature encoders encode features into a 2-D pseudo image by fixed means, as shown in Figure 3.2.

The learnable encoder uses an end-to-end learnable method to extract features from the voxel-grid data. Zhou and Tuzel, 2018 first introduced the voxel feature encoder (VFE), which is a learnable encoder, as shown in Figure 2.2. After voxelization, the point-wise feature is transformed through the VFE layer, which is composed of a fully connected network (FCN) into a feature space. Here, information from the point features $\mathbf{f_i} \in \mathbb{R}^m$ can be aggregated to encode the shape of the surface contained within the voxel, where $i \in [1, N]$. Here, N is the number of points in a point cloud, and m is the dimension of a point. The FCN consists of a linear layer followed by a batch normalization layer and an ReLU layer. An element-wise max-pooling technique is used to locally aggregate the transformed features and then output a feature $\vec{\mathbf{f}}$ for $\mathbf{P_f}$. Finally, the max-pooled feature $\vec{\mathbf{f}}$ is concatenated with each point feature $\mathbf{f_i}$ to generate the final 4-D feature $\mathbf{P_{vfe}}$. A 3-D CNN was utilized for

feature extraction. Wen and Jo, 2019 directly reshaped the 4-D feature $\mathbf{P_{vfe}}$ into a 3-D feature, and then a 2-D CNN was used for feature extraction.

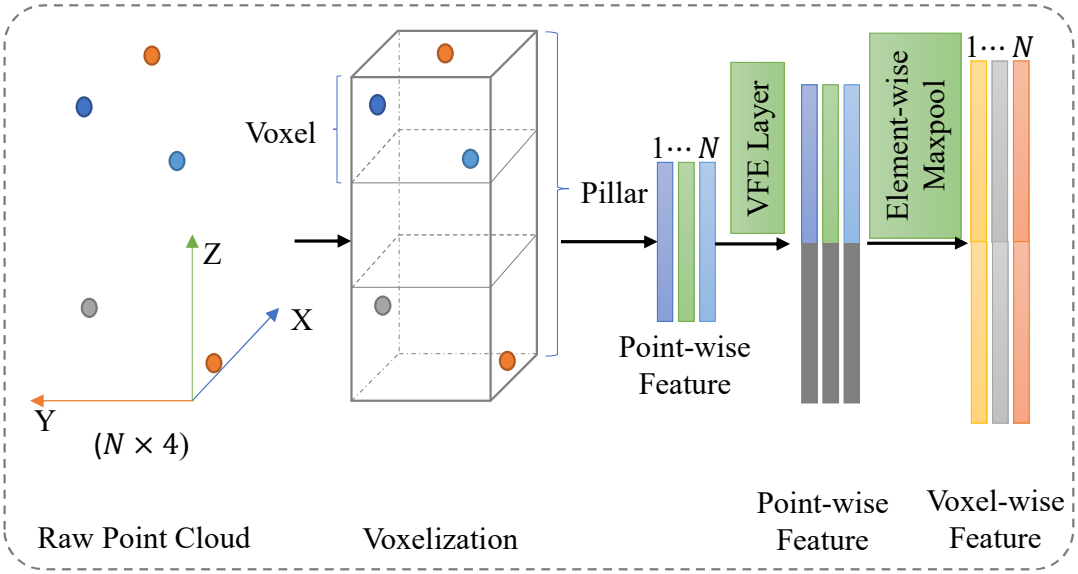


FIGURE 2.2: The process of a voxel-wise feature generation for the voxel feature encoder (VFE). VFE layer adopts a fully connected layer to learn maximum representative pointwise feature and then it is concatenated with each pointwise feature.

VGG network (Simonyan and Zisserman, 2015) is invented by VGG (Visual Geometry Group) from University of Oxford. Though VGG is the 1st runner-up, not the winner of the ILSVRC (ImageNet Large Scale Visual Recognition Competition) 2014 in the classification task. Nevertheless, VGG beats the GoogLeNet (Szegedy et al., 2015) (the winner of ILSVLC 2014 in classification task) and won the localization task in ILSVRC 2014.

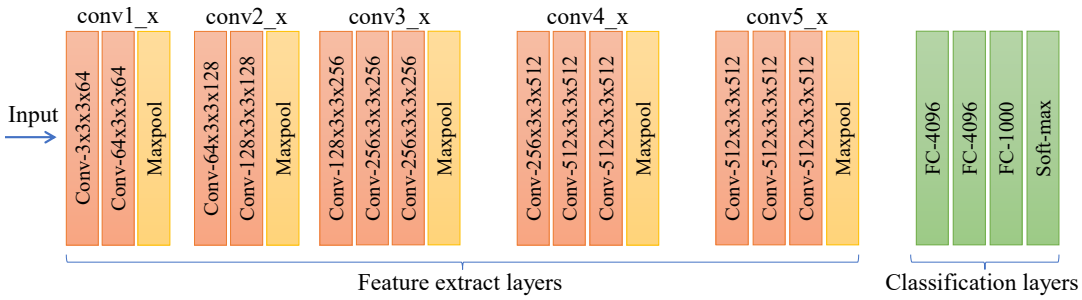
And it is the first year that there are deep learning models obtaining the error rate under 10%. The most important is that there are many other models built on top of VGG network or based on the 3×3 convolution idea of VGG for other purposes or other domains.

Architecture. To reduce the number of parameters, authors propose to use a small respective field to replace large one. Authors conclude:

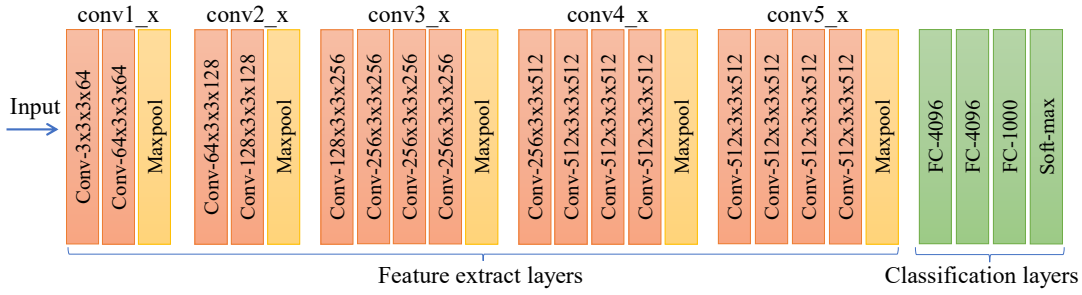
- Incorporate multiple non-linear rectification layers instead of a single rectification layer are more discriminative.
- It helps to decrease the number of parameters while keeping performance. For example, using 2 layers of 3x3 filter is equal to 1 layer of 5x5 filter but using

fewer parameters. The number of a parameter is reduced by 28% ((25–18)/25).

Simonyan and Zisserman, 2015 initialized 6 different ConvNet to see the performance of stacking layers. The difference is the number stacking layer within the same blocks. For example, VGG-16 uses 3 Conv3–256 layers while VGG-19 uses 4 Conv3–256 layers in the third layer of blocks. The two famous configurations of VGG is VGG-16 and VGG-19. The layer structures of VGG-16 and VGG-19 are illustrated in Figure 2.3.



(A) VGG-16 (layers).



(B) VGG-19 (layers).

FIGURE 2.3: Architecture of the VGG network with different layer structures. All convolution layers have stride = 1

Since the plain architecture, VGG-16 is widely used in LiDAR-camera-based 3-D object detection as a 2-D backbone. Chen et al., 2017 and Ku et al., 2018 only adopt the feature extract layers in VGG-16 as a feature encoder and half each channel number of convolutional layer channels.

2.1.2 3-D Sparse Convolutional Network

Graham and Maaten, 2017 proposed the submanifold sparse convolutions, that can be used to build computationally efficient sparse VGG/ResNet/DenseNet-style networks. Yan, Mao, and Li, 2018 proposed a sparse convolution to extract features

from point clouds. To speed up the feature extraction from the structured voxel-grid representations, Shi et al., 2020 first proposed the 3-D sparse convolutional networks, as shown in Figure 2.4. The 3-D sparse network consists of two kinds of convolutional layers: the submanifold convolution and the sparse convolution. In addition to accelerating feature learning, it also enhances the representation of features. The network configuration for the 3-D sparse backbone as shown in Table 2.1.

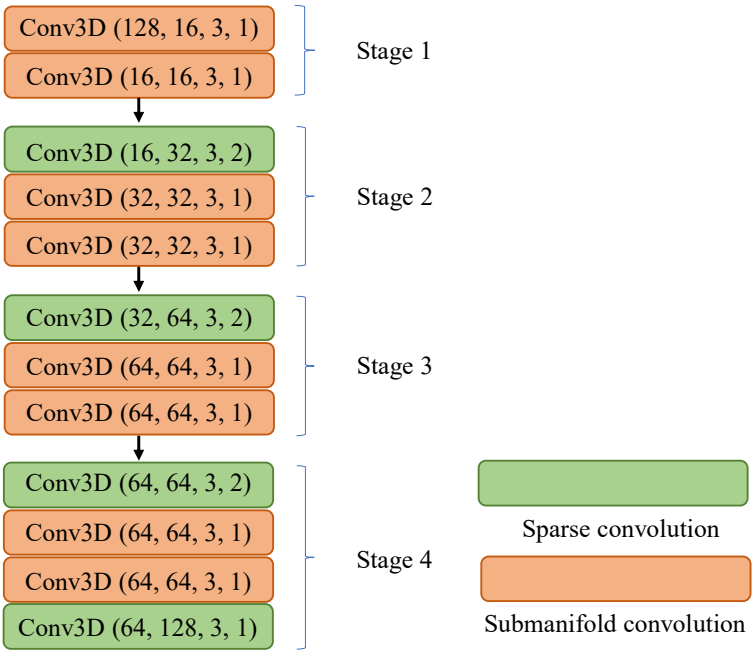


FIGURE 2.4: The 3D backbone architecture. Conv3D (cin, cout, k, s) denotes a convolutional block, where the parameters cin, cout, k, and s represent the input-channel numbers, the output-channel numbers, the kernel size, and the stride, respectively. Each block consist of a 3D convolutional layer followed by a batch normalization layer and a ReLU layer.

2.1.3 PointNet and PointNet++

Due to partial key information loss during the conversion from a point cloud to the voxel-grid representation (Chen et al., 2017; Ku et al., 2018; Lang et al., 2019; Zhou and Tuzel, 2018), Charles et al., 2017; Qi et al., 2017 used several shared multi-layer perceptrons (MLPs) independently model each point in the point cloud. Then, they utilized asymmetric aggregation functions (such as the max pooling) to generate global features. Charles et al., 2017 was the first to adopted shared MLPs to process each point independently. A simple model can be seen in Figure 2.5. The model

TABLE 2.1: The network configuration for the 3-D sparse backbone. The output sizes (width, length, depth, channel) are for the 3-D backbone. The 'layer' includes the information: the convolutional type, filter size, and stride.

Network	Output Size	Stage	Layer
3D Sparse Backbone	(1600, 1408, 41, 16)	Stage1	Sparse_Conv3D, 3, s1 Sub_Conv3D, 3, s1
	(800, 704, 21, 32)	Stage2	Sparse_Conv3D, 3, s2 Sub_Conv3D, 3, s1 Sub_Conv3D, 3, s1
	(400, 352, 11, 64)	Stage3	Sparse_Conv3D, 3, s2 Sub_Conv3D, 3, s1 Sub_Conv3D, 3, s1
	(200, 176, 5, 128)	Stage4	Sparse_Conv3D, 3, s2 Sub_Conv3D, 3, s1 Sub_Conv3D, 3, s1
	(200, 176, 2, 128)		Sub_Conv3D, 3, s1

directly takes point clouds as inputs, applies input transformation and feature transformation, and aggregates point features by max pooling. The input transformation is a data-dependent transformation for automatic alignment, and the feature transformation is an embedding space alignment. The output is classification scores for k classes. Qi et al., 2017 introduced the metric space distances to learn local features with increasing contextual scales and the set learning layers to adaptively aggregate multiple-scale features.

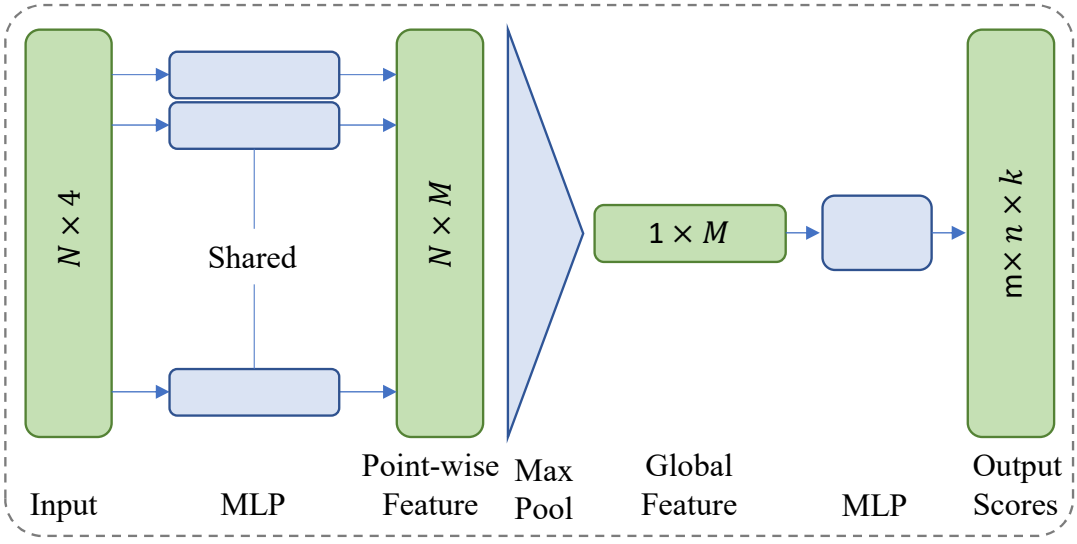


FIGURE 2.5: A simple classification architecture of PointNet (Charles et al., 2017). N is the number of points in the point cloud, M is the dimension of the point-wise feature, and (m, n) and k denote the shape of the output feature map and the classes, respectively.

2.1.4 Graph Convolutional Network

Graph-based methods consider each point in a point cloud as a vertex of a graph and generate directed edges for the graph based on the neighbors of each point. Graph-based methods mainly include four parts: the point cloud input, graph construction, feature learning and pooling, and the key points output, as shown in Figure 2.6. However, since PointNet (Charles et al., 2017) and PointNet++ Qi et al., 2017 treat points independently at a local scale to maintain permutation invariance, they neglect the geometric relationships among points in a point cloud. To tackle this issue, Wang et al., 2018 proposed the EdgeConv technique to aggregate the local geometric structure. They computed a directed graph (vertices and edges) to represent the local structure of a point cloud. Shi and Rajkumar, 2020 introduced the auto-registration mechanism to reduce the translation variance in a graph neural network, which allows a point to align its coordinates according to its features. They also proposed a box merging and scoring method to accurately output predictions.

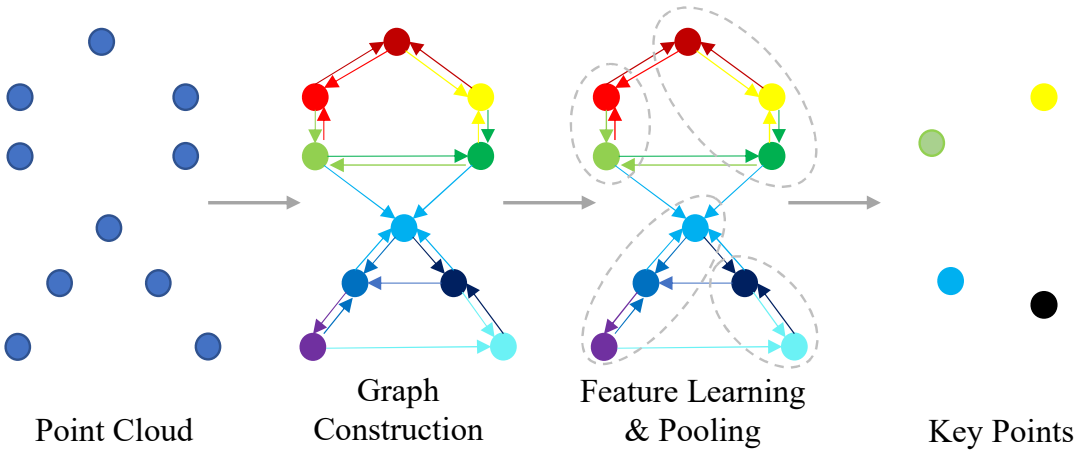


FIGURE 2.6: The process of graph-based methods. It takes a raw point cloud as input and outputs key points after graph construction, feature learning, and feature pooling.

2.2 Frameworks for Object Detection

Currently, there are two frameworks for object detection: one-stage methods (Liu et al., 2016; Redmon et al., 2016; Redmon and Farhadi, 2017) and two-stage methods (Girshick, 2015; Ren et al., 2017). The one-stage method directly outputs predictions, however, the two-stage method first generates proposals in the first stage and then

outputs final predictions based on the proposals in the first stage. Compared to one-stage methods, usually, two-stage methods have higher accuracy. With the development of computer technology, one-stage methods (Lin et al., 2020; Tian et al., 2019) outperform the two-stage methods both accuracy and runtime.

2.2.1 One-stage Framework for Object Detection

Redmon et al., 2016 were the first to propose the one-stage framework for object detection, named YOLO. The framework mainly consists of a backbone, two heavy fully connected layers, a detection head. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance. Although the YOLO model processes images in real-time at 45 FPS, its detection accuracy is lower compared to Fast R-CNN (Girshick, 2015). To solve this issue, Liu et al., 2016 proposed a much fast and higher accuracy object detector, SSD. We can see the differences in Figure 2.7 in more detail. The main differences are as follows:

- In SSD, the authors remove the heavy fully connected layers and employ fully convolutional layers to extract features.
- Compared to the single output in YOLO, SSD outputs multi-scale features for final object predictions.

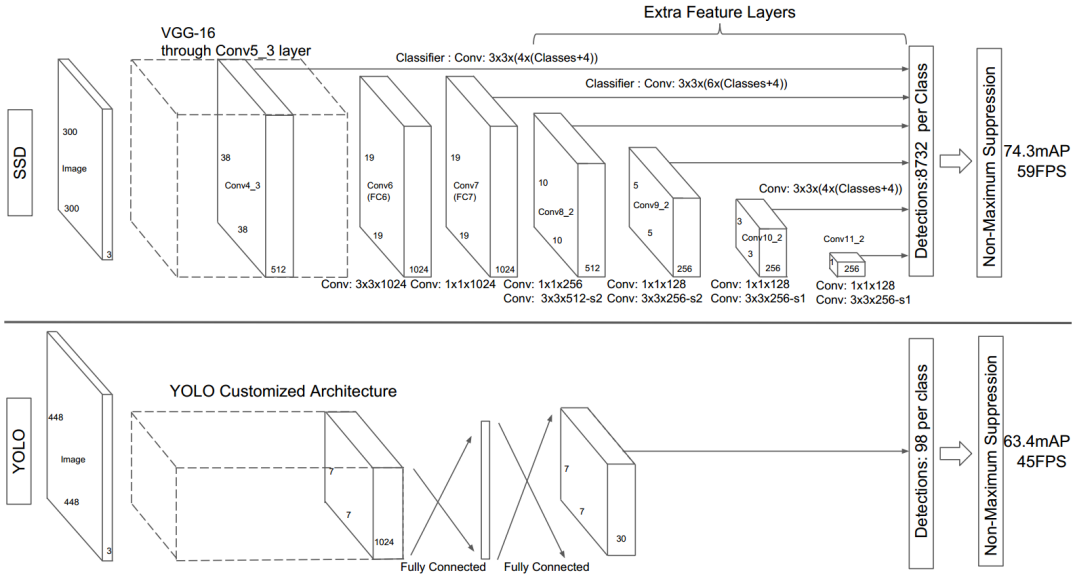


FIGURE 2.7: One-stage architecture comparison, SSD vs YOLO. The Figure is cited from (Liu et al., 2016).

Here, we introduce the famous one-stage method, SSD. It is a single-shot detector proposed by (Liu et al., 2016), for multiple categories that is faster than the previous state-of-the-art single shot detectors (YOLO (Redmon et al., 2016)), and significantly more accurate, in fact as accurate as slower techniques that perform explicit region proposals and pooling (including Faster R-CNN (Ren et al., 2017)). Figure 2.8 illustrates the detailed structure of SSD.

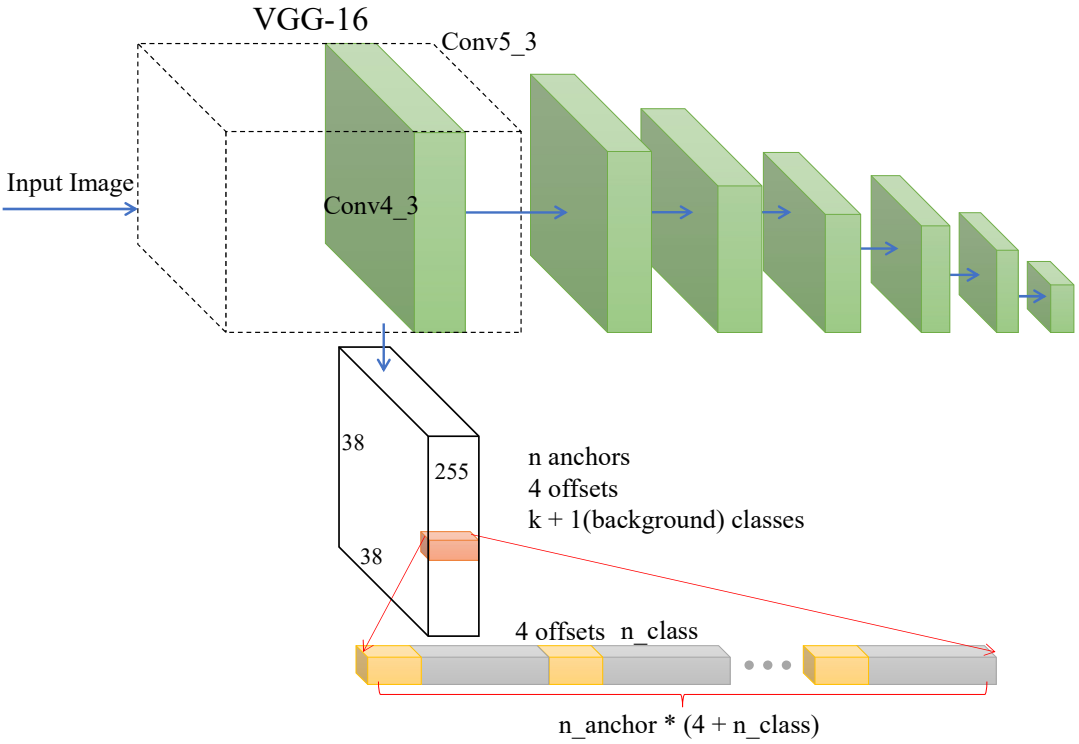


FIGURE 2.8: The details of single shot detector (SSD).

SSD takes the VGG-16 (2.3a) as the backbone and outputs six-resolution feature maps, as shown in Figure 2.7. In each output feature map, the SSD model predicts four bounding box offsets and k+1 object classes (k classes, 1 background) for each anchor box. Anchor boxes are the predefined initial guess of the bounding box for each cell in the feature map. In the original experiments, SSD utilizes four or six anchor boxes for each prediction layer, resulting in a total of 8732 anchor boxes. Finally, a non-maximum suppression (NMS) layer is employed to remove redundant prediction bounding boxes. SSD compares favorably to its state-of-the-art object detector counterparts in terms of both accuracy and speed.

2.2.2 Two-stage Framework for Object Detection

The Region with Convolutional Neural Network (R-CNN) model (Girshick, 2015) is one of the successful deep learning implementations for object detection. This model gathers a set of region proposals (i.e. potential ROIs) from the selective search algorithm (Uijlings et al., 2013). A CNN model then predicts the category of cropped and resized ROIs to determine what kind of object is shown inside. Additionally, it also predicts the bounding box refinement parameters for each cropped image. There are several problems with this scheme. Firstly, the ROIs are obtained from the external region proposal method. Secondly, the computation demanding CNN module extracts the features from each ROIs independently. Thus, the amount of compassion is multiplied by the number of ROIs.

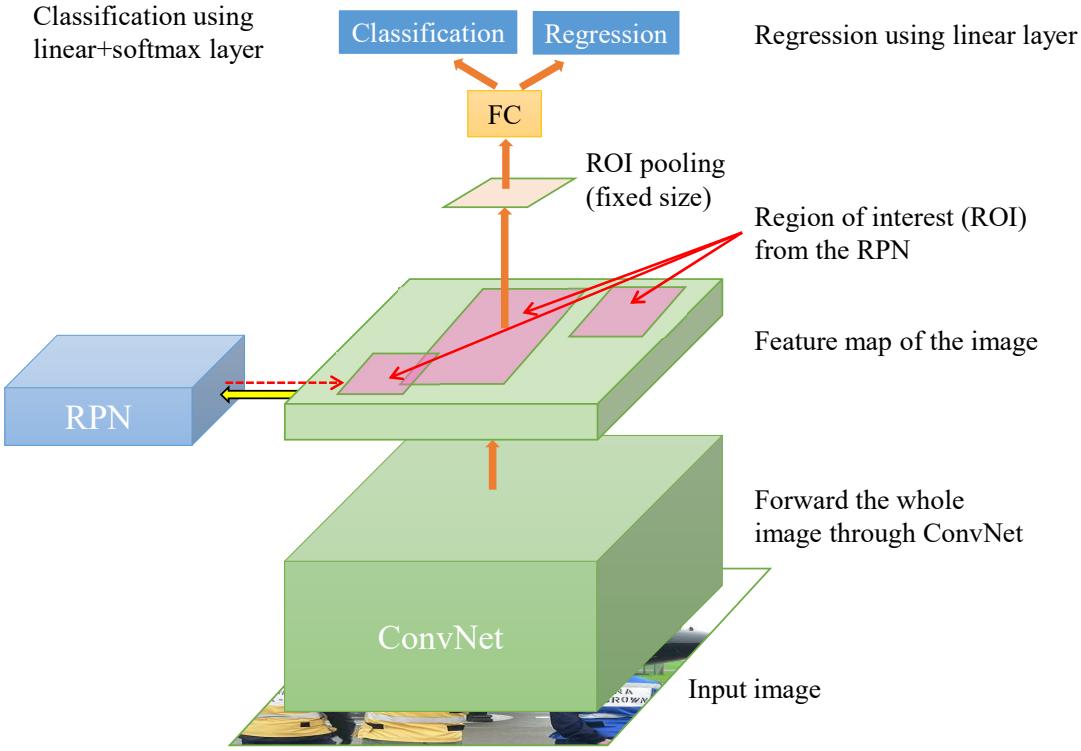


FIGURE 2.9: Two-stage architecture for object detection, Faster R-CNN (Ren et al., 2017).

Several improvements were proposed to alleviate the limitations of the R-CNN model. Instead of cropping the ROIs and passing each crop to CNN independently, the Fast R-CNN (Girshick, 2015) proposes to perform cropping in the feature map generated by feeding the whole image into the CNN model. This allows a more efficient computation but the utilization of an external region proposal method is still becoming a bottleneck.

A region proposal network (RPN) is introduced in the Faster R-CNN model (Ren et al., 2017). This model augments the Fast R-CNN model by changing the external region proposal into an RPN module as illustrated in Figure 2.9. An RPN predicts the bounding box of potential regions from a given feature map. These regions are then cropped and passed into the ROI pooling module which warps all ROIs into the same size. The warped features are then passed into the final classifier which predicts the class and bounding box refinement parameters. As all components are differentiable, end-to-end training on this model is possible.

2.3 Fusion Methods for Multi-Sensor 3-D Object Detection

MV3D (Chen et al., 2017), AVOD (Ku et al., 2018), and MCF3D (Wang et al., 2019) utilize region proposal network (RPN) Ren et al., 2017 to generate 3D proposals based on the fused ROI feature of BEV and camera. MV3D generates 3D proposals from LiDAR features and employs two deep fusion methods shown in Figure 2.10a and Figure 2.10b to fuse the ROI features from multiple modalities. AVOD uses the middle fusion method shown in Figure 2.10c to fuse the ROI features with an element-wise mean operation. MCF3D combines middle fusion method (one layer) with a complementary fusion, such as anchor fusion and proposal fusion. Obviously, the above fusion methods based on ROI features are *local* and inefficient. ContFusion (Liang et al., 2018) proposes a *global* fusion method, as shown in Figure 2.10e, to fuse the BEV feature with different camera feature levels. The *global* method outperformed most of existed fusion method. However, it is only one-way fusion. To address this issue, Wen and Kang-Hyun, 2020 a two-way fusion method, as shown in Figure 2.10f.

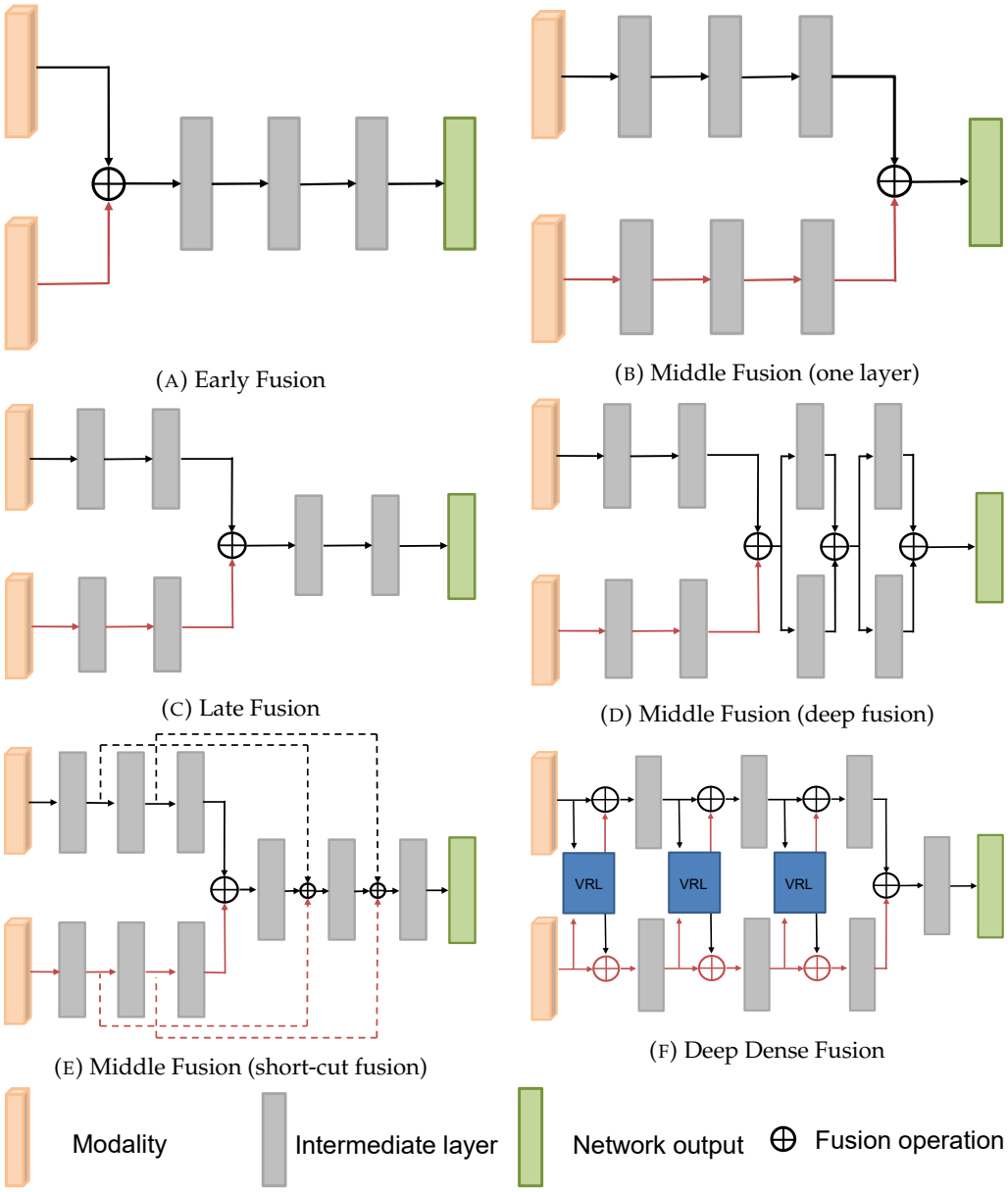


FIGURE 2.10: A comparison of existed fusion methods and the deep dense fusion. Compared with methods (A-E), the deep dense fusion (method F) moves forward to the feature extraction phase and becomes more dense and fully integrates each other's characteristics.

2.4 Classic Multi-Sensor 3-D Object Detection

Currently, there are two classic methods for multi-sensor 3-D object detection: MV3D (Chen et al., 2017) and AVOD (Ku et al., 2018). To have high performance, these two methods all take multiple inputs and adopt two-stage architecture. However, these techniques take much computation costs, and their models run very slowly in the testing stage. We introduce these approaches one by one as follows.

First, we introduce the work proposed by Chen et al., 2017. The detailed architecture is shown in Figure 2.11. The work adopts a two-stage framework to generate 3-D proposals and refine the proposals in the first stage to achieve high performance. Besides that, they also take three different 2-D backbones to extract features from multiple-view images. The proposed model runs at 2 FPS on one NVIDIA GTX 1080Ti GPU. In general, the work is pioneering at the multi-sensor 3-D object detection, however, its performance can be improved in terms of accuracy and speed.

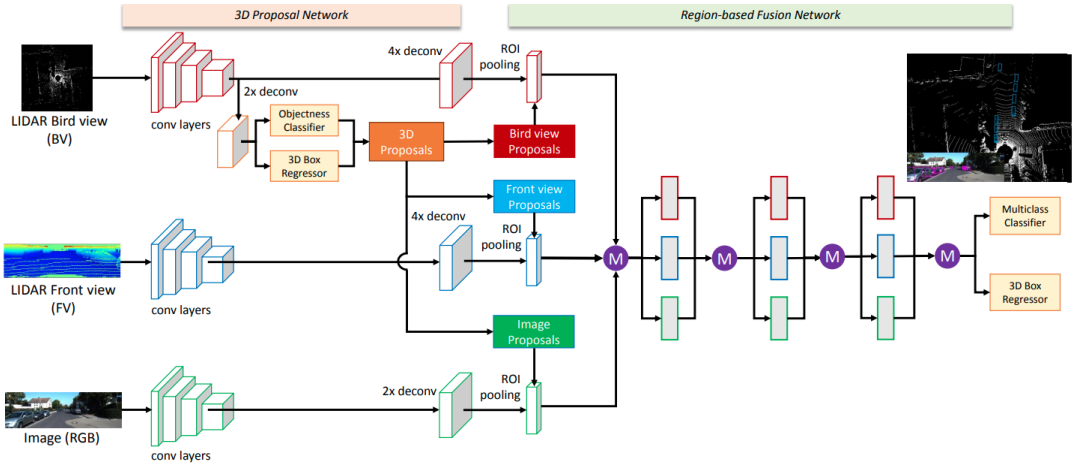


FIGURE 2.11: Multi-View 3D Object Detection Network for Autonomous Driving (MV3D) (Chen et al., 2017).

Second, the work proposed by (Ku et al., 2018) is introduced. Different from MV3D, this work only takes RGB images and their corresponding BEV images as inputs. Like this, they hugely reduce computation costs and make the proposed model run at about 10 FPS. And also they achieve higher accuracy than MV3D. Note that this work still has a gap with LiDAR-based methods. There are two main reasons as follows:

1. They do not fully fuse the two kinds of features from RGB images and point clouds.

2. There exists important information loss during the conversion from point clouds to BEV images.

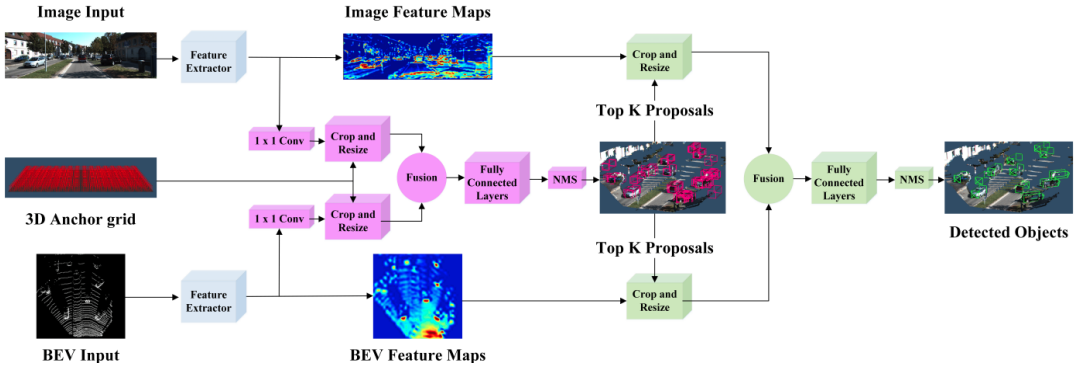


FIGURE 2.12: Joint 3D Proposal Generation and Object Detection from View Aggregation (AVOD) (Ku et al., 2018).

2.5 Dataset and Metrics

The proposed model is trained and evaluated on the KITTI dataset (Geiger, Lenz, and Urtasun, 2012). The KITTI object dataset possesses 7,518 testing frames and 7,481 training frames. Each frame is comprised of a point cloud, stereo RGB images (the left image and the right image), and calibration data. In this research, only a point cloud and the left image with their calibration data are used. To impartially compare the proposed approach with existing methods, the training dataset is divided into two subsets (training subset and validation subset) based on the same criteria, and the ratio of the two subsets is 1:1.

For KITTI's criteria, according to the size, truncation, and occlusion classes of objects, all objects are grouped into three difficulty classes: easy (E), moderate (M), and hard (H). Before October 8th, 2019, KITTI's object detection metric was defined as the 11-point average precision (AP) metric. Since then, the metric has been defined by 40 recall positions. Compared with the 11-point AP, the 40-point AP more properly assesses the quality of an algorithm based on the infinite approximation. Intersection-over-Union (IoU) is the generic evaluation criterion for object detection. In the evaluation of 2D, 3D, and bird's eye view (BEV) detection, the IoU is at the threshold of 0.7 for the car class and 0.5 for the pedestrian/cyclist class. For the average orientation similarity (AOS) we follow the approach in (Geiger, Lenz, and Urtasun, 2012) and define the AOS as:

$$AOS = \frac{1}{N} \sum_{r \in \{0, 0.1, \cdots, 1\}} \max_{\tilde{r}: \tilde{r} \geq r} s\left(\tilde{r}\right), \tag{2.1}$$

$$s\left(r\right) = \frac{1}{\left|D\left(r\right)\right|} \sum_{D(r)} \frac{1 + cos\Delta_{\theta}^{i}}{2} \delta_{i}, \tag{2.2}$$

where $N \in \{11, 40\}$, $r = \frac{TP}{TP+FN}$ is the PASCAL object detection recall, TP means the true positive, FN is the false negative, s is the orientation similarity, $D\left(r\right)$ represents the set of all object detections at recall, and Δ_{θ}^{i} is the difference in angle between estimated and ground truth orientation of detection i, $\delta \in \{0, 1\}$ is the penalty factor.

Chapter 3

Three-Attention Mechanisms for One-stage 3-D Object Detection Based on LiDAR and camera

Recently, 2D object detection (Han et al., 2019; Lv et al., 2018; Hoang, Huang, and Jo, 2020) with deep learning has drawn much attention. Most researchers study 3D object detection based on LiDAR point clouds using 2D detection methods. Point clouds generated by LiDAR are sparse and irregular. Hence, representative studies either convert point clouds into 2D front view images (Chen et al., 2017), 2D bird's-eye-view (BEV) images (Li, Zhang, and Xia, 2016), or structured voxel-grid representations (Zhou and Tuzel, 2018; Wen and Jo, 2019). Then, 2D convolutional layers are used to extract features from the converted images. Some point-based methods (Charles et al., 2017; Qi et al., 2017) directly utilize multi-layer perceptron (MLP) to aggregate features from point clouds. However, LiDAR-based approaches suffer 3D information loss in distant regions due to the sparsity of the point clouds. Compared with point-based methods (Charles et al., 2017; Qi et al., 2017), the BEV-based method (Li, Zhang, and Xia, 2016) is a little faster, however, it suffers partial information loss during the conversion. This work employs the RGB^{D} image to reduce the information loss.

2D RGB images possess dense texture, and high-resolution images have enough cues for small objects. However, it is difficult to extract accurate 3D localization features when using monocular images due to the lack of depth information (Chen et al., 2016; Xu and Chen, 2018; He and Soatto, 2019). Currently, even if stereo images

are used (Li, Chen, and Shen, 2019), the accuracy of the estimated depth is not guaranteed. Therefore, some studies (Chen et al., 2017; Ku et al., 2018; Mozifian, 2018; Li et al., 2019) take mutual advantage of 2D images and point clouds to achieve accurate 3D object detection. These methods directly fuse the view-specific features by a common concatenation (Chen et al., 2017) or an element-wise mean operation (Ku et al., 2018), resulting in poor accuracy of 3D object detection. This paper adopts the region of interest attention (RA) fusion mechanism to deeply merge the view-specific features.

MV3D (Chen et al., 2017) and AVOD (Ku et al., 2018) adopt the two-stage frameworks to detect 3D objects based on point clouds and RGB images. The first stage generates 3D proposals, and the following stage refines the proposals to predict 3D objects. Compared with the one-stage method, the two-stage 3D detection model is relatively time-consuming. Therefore, some works (Mozifian, 2018; Li et al., 2019) utilize the one-stage framework to detect 3D objects. Without the second stage to refine 3D proposals, the above one-stage works yield a worse detection accuracy than the two-stage methods. After analysis, enhancing the feature representation of one-stage methods is the most effective way to improve 3D object detection. Hence, a global feature attention (GFA) mechanism is used for boosting global feature representation.

To overcome the above drawbacks, this paper presents a novel one-stage 3D object detection framework, as shown in Figure 3.1, based on three-attention mechanisms, called TAO3D, which takes raw point cloud and RGB image as inputs. Three attention mechanisms are used to obtain discriminative features. First, the height attention (HA) mechanism is introduced as an auxiliary attention module before the RGB image is fed into a network. Second, a global feature attention (GFA) mechanism models the long-range dependencies in the channel and spatial dimensions simultaneously at the feature extraction phase. Finally, a region of interest attention (RA) mechanism weights RGB image ROIs and BEV ROIs using two learnable parameters.

The main contributions of this framework are summarized as follows:

1. It takes the raw LiDAR point clouds and RGB^D images as inputs instead of the RGB image. RGB^D images contain the height information from point clouds.

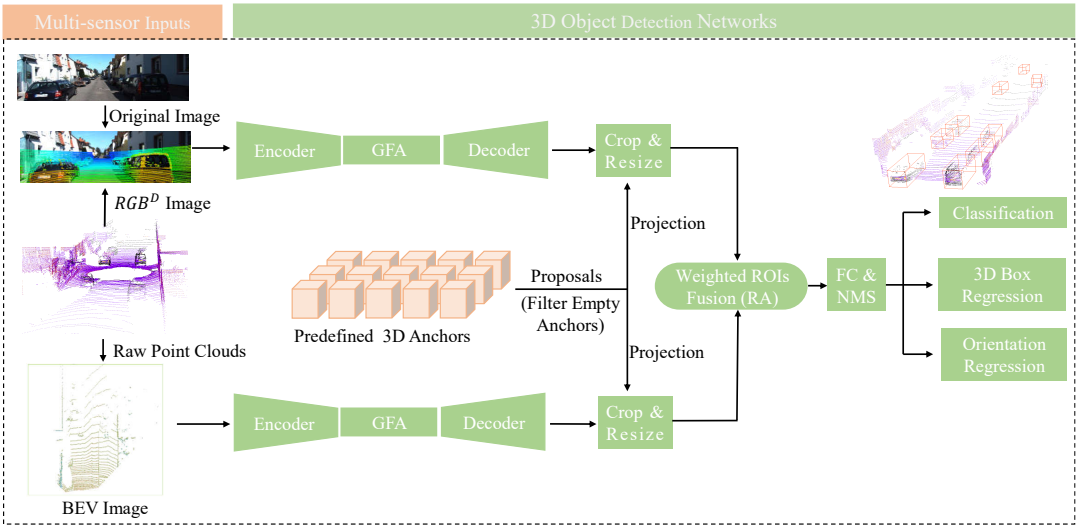


FIGURE 3.1: The architecture of a one-stage 3D object detection network based on LiDAR and camera. The model first employs two sibling branches to extract the features from RGB^D images and BEV images, respectively. Second, the prior anchors have filtered the empty anchors, and then projected onto RGB^D feature maps and BEV feature maps to crop equal length view-specific ROIs. Finally, the fused ROIs are utilized for classification and regression. Best viewed with color.

2. The GFA mechanism captures the feature dependencies in both the channels and spatial dimensions at the feature extraction stage and makes the features much more discriminative.

3. The RA mechanism weights the paired BEV ROIs and RGB image ROIs firstly and then fuses them using the addition operation. This gives more weight to important features.

The proposed one-stage 3D object detection framework outperforms state-of-the-art LiDAR-Camera-based methods on the KITTI benchmark (Geiger, Lenz, and Urtasun, 2012).

3.1 Related Work

This section mainly reviews the related works for 3D object detection based on LiDAR and camera, and the attention networks for computer vision tasks.

3.1.1 LiDAR-Camera-Based 3D Object Detection

MV3D (Chen et al., 2016) first introduced multi-modality (RGB image, front image, BEV image) 3D object detection with three backbones to extract view-specific features. Compared with MV3D (Chen et al., 2016), AVOD (Ku et al., 2018) only takes RGB images and BEV images as inputs to reduce model runtime. Both MV3D and AVOD make use of a two-stage 3D object detection framework. To speed up training and inference, AVOD-SSD (Mozifian, 2018) and (Li et al., 2019) adopt a one-stage 3D object detection framework. The models run a little faster than AVOD (Ku et al., 2018), but both the performances greatly drop. In the first step, the proposed work enhances the one-stage detection framework with the RGB^D images at the input phase of the RGB image.

3.1.2 Attention Networks

Attention modules model long-range dependencies and have been widely applied in segmentation tasks (Fu et al., 2019; Wang et al., 2017). DANet (Fu et al., 2019) introduces a self-attention mechanism to capture rich contextual dependencies for scene segmentation, which models the semantic interdependencies in the channels and spatial dimensions, respectively. CBAM (Woo et al., 2018) sequentially infers attention maps along two separate dimensions, channel and spatial, then the attention maps are multiplied to the input feature map for adaptive feature refinement. MCF3D (Wang et al., 2019) introduces a self-attention mechanism for 3D object detection. (Wang et al., 2018) designed a non-local method to obtain the response at a position as a weighted sum of the features at all positions. (Hu, Shen, and Sun, 2018) presented a channel attention module using a squeeze and excitation network. (Fu et al., 2019) reported a dual-attention network for the scene segmentation task. To speed up the scene segmentation task, (Fu et al., 2020) further proposed a compact dual-attention network. Different from the above methods, the proposed work employs two attention mechanisms from the global to the local to boost 3D object detection. In the second step, the proposed method utilizes the GFA to enhance the global feature representations. In the final step, the RA is used to enhance the local feature representations.

3.2 The Proposed Architecture

The main innovation of the proposed framework, as depicted in Figure 3.1, employs three-attention mechanisms to make extracted features discriminative. The proposed 3D detection framework mainly includes two parts: the first one is the multi-sensor inputs, and the other one is the 3D object detection networks.

3.2.1 Multi-sensor Inputs

This paper directly takes the raw LiDAR point clouds and RGB images as inputs. In the preprocessing, the BEV images and RGB^D images are generated simultaneously by fixed means. Note that LiDAR and camera use two different coordinate systems. In the coordinate system of LiDAR, the x-axis points forward, the y-axis points to the left of the vehicle, and the z-axis points upward. However, in the camera's coordinate system, the x-axis points to the right of the car, the y-axis points downward, and the z-axis points forward. That is why after the height information in LiDAR is projected onto the RGB image plane, it is referred to as depth.

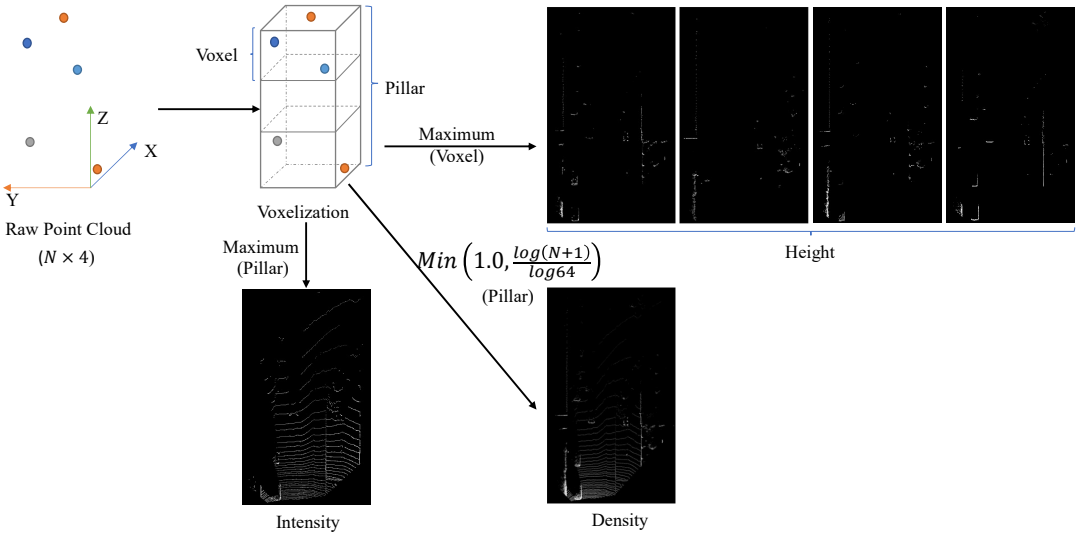


FIGURE 3.2: The details to generate the BEV image.

Bird's-Eye-View Representation. Point clouds are generated by LiDAR, which encodes the 3D (x, y, z) coordinates and intensity information (I) of surrounding objects. Like AVOD (Ku et al., 2018), a six-channel BEV image encodes the density and height information in each voxel of a LiDAR frame. Different from MV3D (Chen et al., 2017), the BEV map does not encode the intensity of point clouds. Specifically,

the area to encode BEV image is $\{x, y, z \mid x \in [0, 70], y \in [-40, 40], z \in [-2.3, 0.2]\}$. The voxel grid size is 0.1 meter on both of the x-axis and the y-axis. To keep as much height information as possible, the point clouds are equally sliced into five slices along the z-axis, and the height value is the absolute height relative to the ground. The density map is encoded as $min\left(1.0, \frac{log(N+1)}{log(64))}\right)$ in each pillar, where N is the number of points in one pillar. Note that the density features are computed for the whole point clouds while the height feature is computed for five slices. The details to generate the BEV image as shown in Figure 3.2.

RGB^D Representation. MCF3D (Wang et al., 2019) encoded the intensity of point clouds as an additional channel of the original RGB image and named it RGB-I. Different from MCF3D, this paper embeds the projected height information of point clouds into the original RGB image, and calls it RGB^D with 3 channels. The whole process is divided into three steps. First, point clouds (X, Y, Z) are mapped onto the original image $(W \times H)$ plane as follows:

$$\begin{pmatrix} u & v & 1 \end{pmatrix}^T = \mathbf{M} \cdot \begin{pmatrix} X & Y & Z & 1 \end{pmatrix}^T, \tag{3.1}$$

$$\mathbf{M} = \mathbf{P}_{rect} \cdot \begin{pmatrix} \mathbf{R}_{velo}^{cam} & \mathbf{t}_{velo}^{cam} \\ 0 & 1 \end{pmatrix}, \tag{3.2}$$

where (u, v) is the image coordinate, $\mathbf{P}_{rect}$ is a project matrix, $\mathbf{R}_{velo}^{cam}$ is the rotation matrix from LiDAR to the camera, $\mathbf{t}_{velo}^{cam}$ is a translation vector, and $\mathbf{M}$ is the homogeneous transformation matrix from LiDAR to the camera.

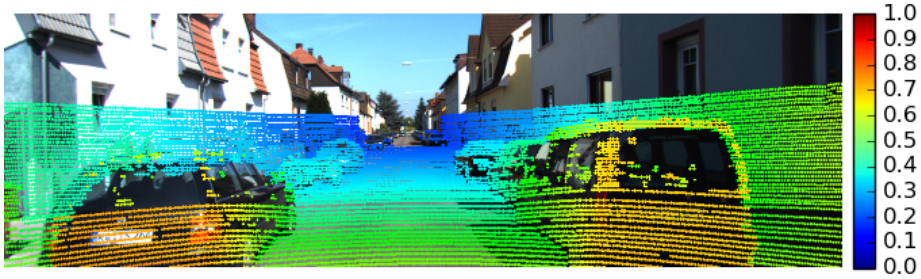
Second, the points $\{(x, y, z) \mid x \in X, y \in Y, z \in Z\}$ located into the image size $W \times H$ are kept. Meanwhile, the LiDAR points are projected to the camera coordinates and denoted as (x_c, y_c, z_c):

$$\begin{pmatrix} x_c & y_c & z_c \end{pmatrix}^T = \mathbf{M} \cdot \begin{pmatrix} x & y & z & 1 \end{pmatrix}^T. \tag{3.3}$$

Finally, z_c is mapped between 0 and 255 and then assigned to the corresponding image coordinate (u, v). Figure 3.3 shows the difference between the original RGB image and the RGB^D image.

Proposals Generation. $\{x, y \mid x \in [0, 70], y \in [-40, 40]\}$ are the area of LiDAR point

(A) Original RGB image.



(B) RGB^D image.

FIGURE 3.3: The top image is the original RGB image and the bottom one is the RGB^D image. Red color means the depth is shallow, and blue color means the depth is deep. Best viewed with color.

cloud, a set of 3D prior anchor boxes are placed onto it. Each 3D prior anchor box is parameterized by two heights (h_1, h_2) and four corners $(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4)$ in meters. However, where to put the anchors? In this work, we put anchors on the fitted road plane clustered from each frame of raw point cloud, as shown in Figure 3.4. To generate the 3D prior anchor box grid, (c_x, c_y) pairs are sampled at an interval of 0.5 meters in the above area, and c_z is computed based on the LiDAR's height above the ground plane (Ku et al., 2018). In this way, 89,600 anchors are generated in total. The size (w, h, l) is clustered from the ground truth of KITTI's training dataset (Geiger, Lenz, and Urtasun, 2012). For the car class, (w, l, h) takes the values of $(1.58, 3.51, 1.51)$ and $(1.65, 4.23, 1.55)$. For the pedestrian and cyclist class, (w, l, h) takes the values of $(0.63, 0.82, 1.77)$ and $(0.57, 1.77, 1.72)$, respectively. Specifically, each location has four anchors with two sizes and two orientations $\{0°, 90°\}$ for the car class.

Since the LiDAR point cloud is sparse, this causes a large number of empty anchors. After our statistics, there are about 5K to 25K anchors that contain LiDAR points. To speed up computation, the empty anchors are removed by computing an integral image over the point occupancy map (Ku et al., 2018) in both the training and testing stages. Based on the non-empty 3D anchors, the sampling method,

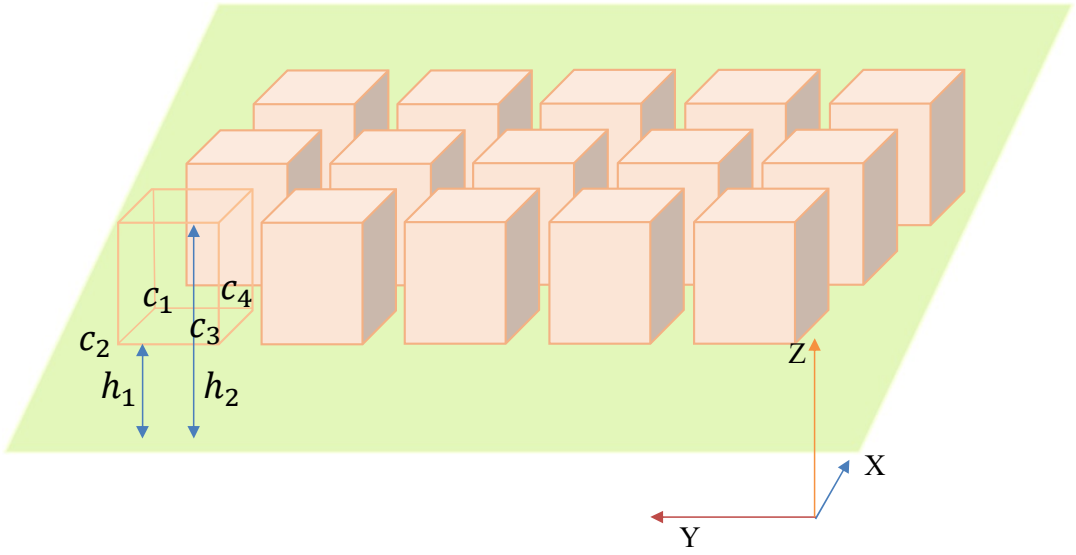


FIGURE 3.4: The plane-based method to put anchors.

as introduced in section 3.2.7, is employed to generate the 3D proposals. The 3D proposals are projected onto the BEV and RGB image plane to get the paired view-specific ROI crops, and the ROI crops are resized to a fixed size $N \times N \times C_r$. Note that the 3D proposal generation is completed in the preprocessing.

3.2.2 3D Object Detection Networks

This section will introduce the 3D detection network in the order of use, as shown in Figure 3.1.

3.2.3 Feature Encoder and Decoder

The feature detector comprises two sibling branches, one is for RGB image feature extraction, and the other one is for BEV feature extraction. Each branch consists of a feature encoder and feature decoder. VGG-16 (Simonyan and Zisserman, 2015) is chosen as the feature encoder. Our encoder, as shown in Figure 3.5, differs from the VGG-16 encoder as follows:

- The first four convolution blocks are kept, and the fifth convolution block and the fully connected layers are removed.
- All convolution channel numbers are reduced to half of the original VGG-16.

For the decoder, three deconvolutional layers are used to obtain a high-resolution feature map. The high-resolution map offers more information for small objects. Same as FPN (Lin et al., 2017), lateral connections link the encoder and the decoder to build high-level semantic feature maps at all scales. Figure 3.5 shows the details.

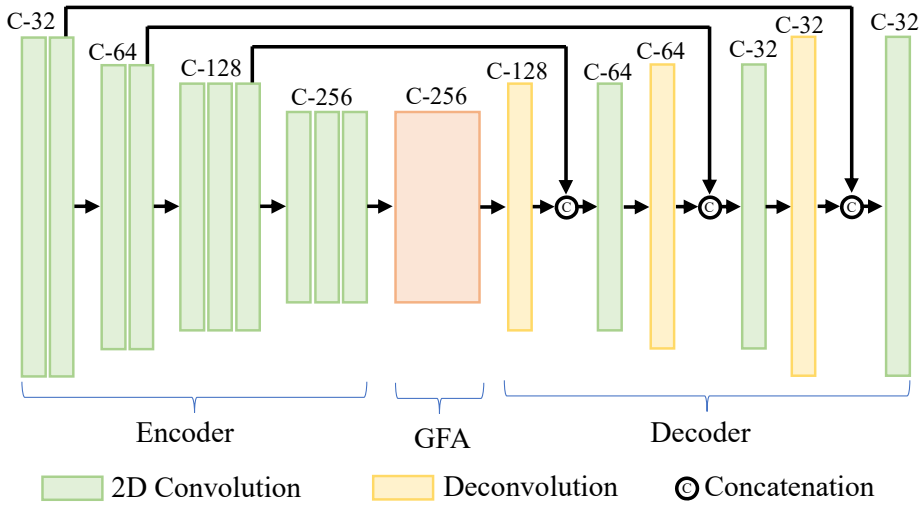


FIGURE 3.5: The feature extraction network, which includes three parts: encoder, GFA, and decoder. 'C-#' means the number of feature map channels. Best viewed with color.

3.2.4 Global Feature Attention

Inspired by DANet (Fu et al., 2019), the global feature attention (GFA) mechanism is proposed to integrate local features with their global dependencies adaptively. The GFA mechanism is much more efficient and also requires less computation as compared to DANet. The GFA mechanism includes two attention networks, as shown in Figure 3.6. One is the position attention network (PAN), and the other one is the channel attention network (CAN).

The PAN, as shown in Figure 3.6a, focuses on modeling rich contextual relationships over local features. Given the local feature $\mathbf{A} \in \mathbb{R}^{H \times W \times C}$, first $\mathbf{A}$ is reshaped to $\mathbf{D} \in \mathbb{R}^{N \times C}$, where $N = W \times H$ is the number of pixels. Meanwhile, $\mathbf{A}$ is reshaped and transposed to $\mathbf{A}^{RT} \in \mathbb{R}^{C \times N}$. After that, matrix multiplication is employed between $\mathbf{D}$ and $\mathbf{A}^{RT}$ and also the softmax function is utilized to compute the spatial attention map $\mathbf{S} \in \mathbb{R}^{N \times N}$:

$$s_{ji} = \frac{exp(\mathbf{D}_i \cdot \mathbf{A}_j^{RT})}{\sum_{i=1}^{N} exp(\mathbf{D}_i \cdot \mathbf{A}_j^{RT})}, \tag{3.4}$$

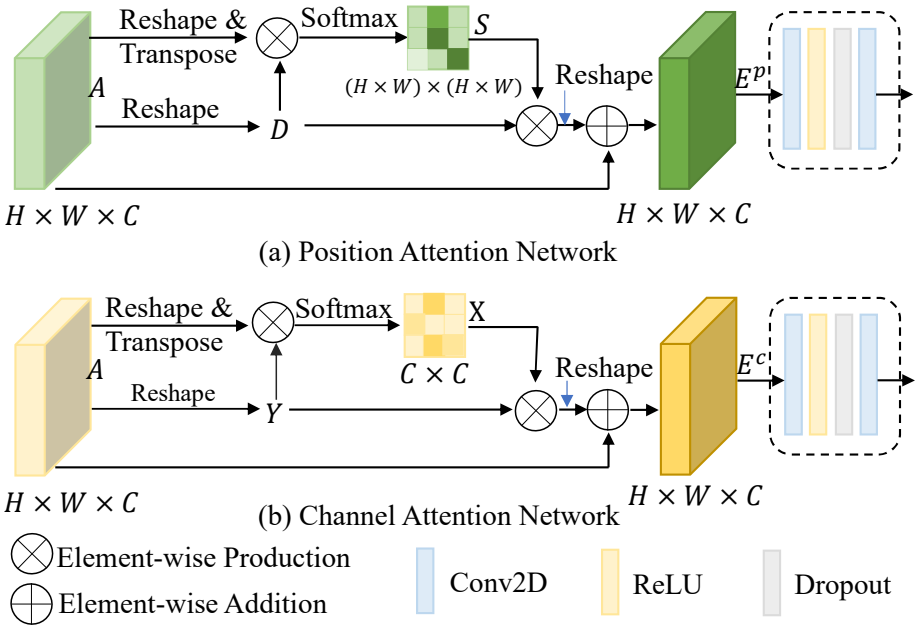


FIGURE 3.6: The global feature attention mechanism includes the position attention network, the channel attention network, and an auxiliary convolution block. Best viewed with color.

where s_{ji} measures the *i-th* position's influence on the *j-th* position, and $i, j \in [1, W \times H]$. The more similar the feature representations of the two locations, the higher the correlation between them.

Second, matrix multiplication is utilized between **D** and **S**, and the product is reshaped to $\mathbb{R}^{H \times W \times C}$. Finally, an element-wise sum operation is performed as follows:

$$\mathbf{E}_j^p = \alpha \sum_{i=1}^{N} (s_{ji} \mathbf{D}_i) + \mathbf{A}_j, \tag{3.5}$$

where α is a learnable parameter to re-weight the new generative feature map. From Equation 5, it can be concluded that the resulting feature $\mathbf{E}_j^p$ at each location j is a weighted sum of the feature at all locations and the original feature.

The CAN, as shown in Figure 3.6b, is designed to exploit the interdependencies between channel maps, since each high-level channel map possesses different semantic responses. The CAN emphasizes the interdependence of the feature maps and boosts the feature representation of specific semantics. The whole reasoning process is the same as that of the PAN, and the difference is that **A** is reshaped to

$\mathbf{Y} \in \mathbb{R}^{N \times C}$ firstly. The details are as follows:

$$x_{ji} = \frac{exp(\mathbf{A}_j^{RT} \cdot \mathbf{Y}_i)}{\sum_{i=1}^{C} exp(\mathbf{A}_j^{RT} \cdot \mathbf{Y}_i)}, \tag{3.6}$$

where x_{ji} measures the *i-th* channel's influence on the *j-th* channel, and $i, j \in [1, C]$.

$$\mathbf{E}_j^c = \beta \sum_{i=1}^{C} (\mathbf{Y}_i x_{ji}) + \mathbf{A}_j, \tag{3.7}$$

where β is a learnable parameter to re-weight the new generative feature map. The feature $\mathbf{E}_j^c$ at each channel j is a weighted sum of the feature at all channels and the original feature. This adds the benefit of enhancing the distinguishing ability of each channel.

Also, an auxiliary convolutional block is appended to each PAN and CAN, which includes a 2D convolutional layer, a Rectified Linear Unit (ReLU) activation function, a dropout layer (rate=0.5), and a 2D convolutional layer. The filter size of the two convolutional layers is 3×3. Note that the two outputs' shapes of 2D convolutions are the same as those of the PAN and the CAN. The auxiliary convolution block achieves 1.42% gains in 3D object detection.

Finally, an element-wise sum operation is performed for the outputs of the PAN and the CAN. Then the result is fed into the next stage.

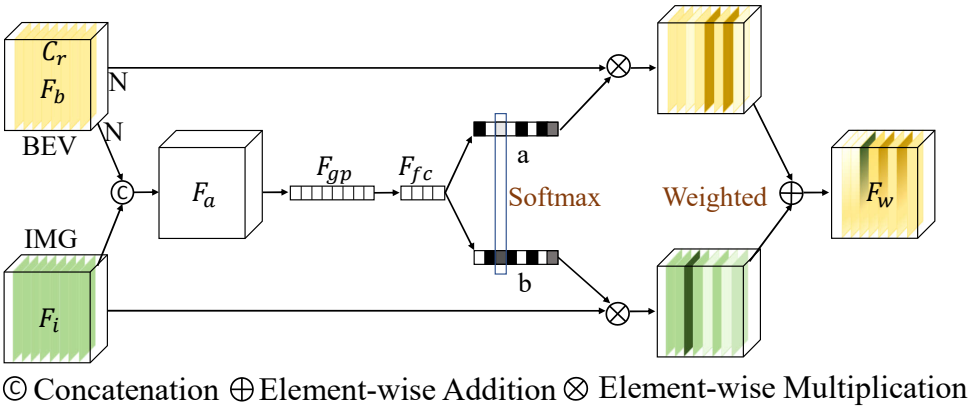


FIGURE 3.7: The region of interest attention (RA). It introduces a soft self-attention mechanism to weight each channel of the BEV and RGB image as pair and is a new fusion method besides the element-wise addition and concatenation operation.

3.2.5 Region of Interest Attention (RA)

MV3D (Chen et al., 2017) and AVOD (Ku et al., 2018) only simply combine the ROI crops from both the RGB image proposals and the BEV proposals by an addition operation or a concatenate operation. This paper introduces the RA, as shown in Figure 3.7, to weight RGB image ROIs and BEV ROIs by channel. Specifically, for any given BEV ROI $\mathbf{F}_b \in \mathcal{R}^{N\times N\times C_r}$ and an image ROI $\mathbf{F}_i \in \mathbb{R}^{N\times N\times C_r}$, first $\mathbf{F}_b$ and $\mathbf{F}_i$ are fused as $\mathbf{F}_a \in \mathbb{R}^{N\times N\times 2C_r}$ by a channel concatenation operation. Second, the $\mathbf{F}_a$ is fed into a global mean-pooling to output $\mathbf{F}_{gp} \in \mathbb{R}^{2C_r}$. Third, the $\mathbf{F}_{gp}$ is fed into a two-layer fully connected layers (FC). The first layer of the FC outputs $\mathbf{F}_1 \in \mathbb{R}^{d\times 1}$, where $d\ \{d \mid d = max(2C_r/r, 32)\}$ is a parameter based on a reduction ratio r to optimize the efficiency of this model. The second layer of the FC outputs $\mathbf{F}_{fc} \in \mathbb{R}^{2C_r\times 1}$. The $\mathbf{F}_{fc}$ is reshaped as $\mathbf{F}_2 \in \mathbb{R}^{2\times C_r}$. Then a softmax function is used for the $\mathbf{F}_2$ by channel.

$$\begin{aligned} a_c &= \frac{e^{A_c}}{e^{A_c} + e^{B_c}}, \\ b_c &= \frac{e^{B_c}}{e^{A_c} + e^{B_c}}, \end{aligned} \tag{3.8}$$

where $\mathbf{A}$, $\mathbf{B}$ are the first-row vector and the second-row vector of $\mathbf{F}_2$, $\mathbf{a}$ and $\mathbf{b}$ are the attention vector for $\mathbf{F}_b$ and $\mathbf{F}_i$, respectively, and $c \in [1, C_r]$ is the channel number of each ROI.

Finally, the fusion of the paired ROIs with the RA mechanism is as follows:

$$\begin{aligned} \mathbf{F}_w^c &= a_c \cdot \mathbf{F}_b^c + b_c \cdot \mathbf{F}_i^c, \\ w.r.t &\quad a_c + b_c = 1, \end{aligned} \tag{3.9}$$

where $\mathbf{F}_w$ is the weighted sum between $\mathbf{F}_b$ and $\mathbf{F}_i$. Compared with the previous fusion methods, the RA is an attention mechanism for the fusion of view-specific ROIs.

3.2.6 Loss Function

The equal-length feature $\mathbf{F}_w$ is fed into a three-layer FCs (2048, 2048, 2048) to deeply merge and then the fused tensor is fed into a 3D detection head with three parallel branches: classification (class no.), box regression (10), and angle regression (1). Note

that each branch only has one FC layer and the number in each bracket means the dimension of FC. MV3D (Chen et al., 2017) encodes a 3D box as eight corners and regresses them. However, it does not consider the physical constraints of a 3D box. To reduce the redundancy and keep the physical constraints, AVOD (Ku et al., 2018) encodes a 3D box with four corners and two heights. Different from AVOD, our method proposes a plane-based 3D bounding box with an 11-dimensional vector $(x_1 \cdots x_4, y_1 \cdots y_4, h_1, h_2, \theta)$. The corresponding regression residuals between the 3D anchors and ground truth are defined as follows:

$$\begin{aligned} \Delta x &= \frac{x_c^g - x_c^a}{d^a}, \quad \Delta y = \frac{y_c^g - y_c^a}{d^a}, \\ \Delta h &= log(\frac{h^g}{h^a}), \quad \Delta \theta = sin(\theta^g - \theta^a), \end{aligned} \tag{3.10}$$

where $d^a = \sqrt{(x_2 - x_1)^2 + (y_4 - y_1)^2}$ is the diagonal of the base of the anchor box. The localization loss function and the angle loss function are as follows:

$$L_{box} = \sum_b Smooth_{L1}(\Delta b), \tag{3.11}$$

$$L_{angle} = \sum_\theta Smooth_{L1}(\Delta \theta), \tag{3.12}$$

where the $b \in (x_1 \cdots x_4, y_1 \cdots y_4, h_1, h_2)$ and $Smooth_{L1}$ is the smooth L1 loss function in the Fast R-CNN (Girshick, 2015).

For the object classification loss, the focal loss (Lin et al., 2020) is used:

$$L_{cls} = -\alpha_a (1 - p^a)^\gamma log(p^a), \tag{3.13}$$

where p^a is the class probability of an anchor, $\alpha = 0.25$, and $\gamma = 2$. The total loss can be formulated as follows:

$$Loss = \frac{1}{N_{pos}} (\beta_1 L_{box} + \beta_2 L_{cls} + \beta_3 L_{angle}), \tag{3.14}$$

where N_{pos} is the number of positive anchors and $\beta_1 = 7.0$, $\beta_2 = 5.0$, and $\beta_3 = 1.0$. For the car class, an anchor is defined as positive if it has a 2D IoU greater than 0.60 (pedestrian/cyclist is 0.3) with its paired ground truth. If it has a 2D IoU less than

0.55 (pedestrian/cyclist is 0.3), the anchor is labeled as negative. The other anchors are ignored when computing the loss.

3.2.7 Training and Inferring

In training, the proposed model is trained using mini-batches containing 16,384 proposals (positive and negative ratio 1:1) for one frame.

In inferring (validation and testing), the non-empty anchors will be directly used as the proposals to crop and resize the view-specific ROIs. 2D non-maximum suppression (NMS) at an IoU threshold of 0.01 on the BEV boxes is utilized to remove the redundant 3D proposals, and the top 15 3D predictions are kept.

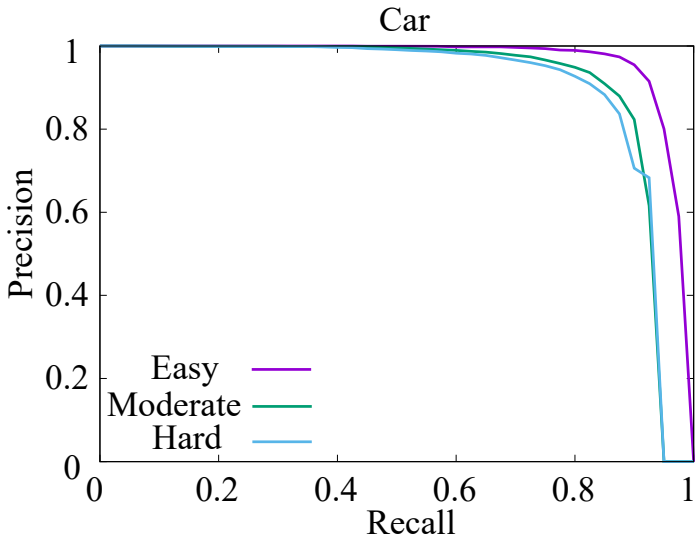
TABLE 3.1: Comparison with state-of-the-art methods. All methods are compared using the 3 difficulties: easy (E), moderate (M), and hard (H). For easy understanding, the top two numbers are highlighted in bold and italic for each column and the second best is shown in blue. All methods accept RGB images and point clouds as input. "-" means that the data can not be found.

Method	Runtime (ms)	3D (%)				BEV (%)			
		E	M	H	mAP	E	M	H	mAP
MV3D	360	71.29	62.68	56.56	63.51	86.55	78.10	76.67	80.44
F-PointNet	170	83.76	70.92	63.65	72.78	88.16	84.02	76.44	82.87
PC-CNN	500	57.63	51.74	51.39	53.59	83.61	77.36	69.61	76.86
AVOD	80	83.11	74.02	67.84	74.99	-	-	-	-
AVOD-FPN	100	84.41	74.44	68.65	75.83	89.37	86.09	79.13	84.86
MVX-Net	150	85.50	73.30	67.40	75.40	89.50	84.90	79.00	84.47
MCF3D	160	84.11	75.19	74.23	77.84	88.82	86.11	79.31	84.75
AVOD-SSD	90	82.36	72.92	67.07	74.12	89.00	85.08	78.91	84.33
Cont-Fuse	60	**86.32**	73.25	67.81	75.79	**95.44**	**87.34**	82.43	**88.40**
C-Retina	90	78.62	72.77	67.21	72.87	89.01	84.69	78.71	84.14
Proposed	110	85.12	**76.23**	**74.46**	**78.60**	89.64	86.23	**85.60**	87.16

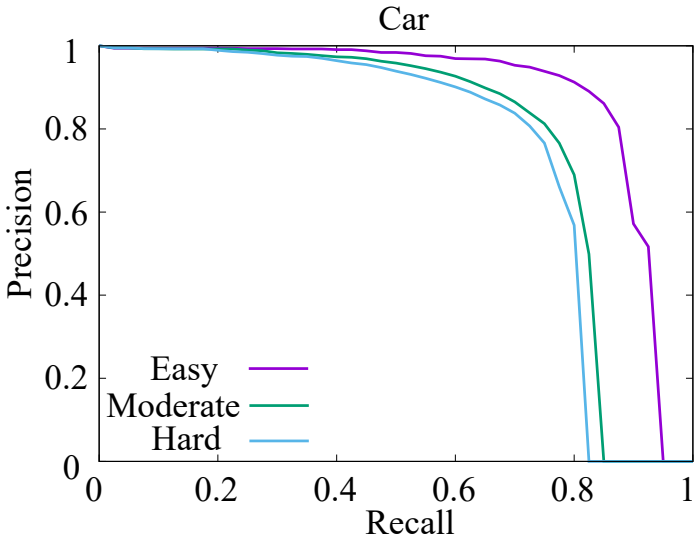
3.3 Experiments

3.3.1 Dataset and Metric

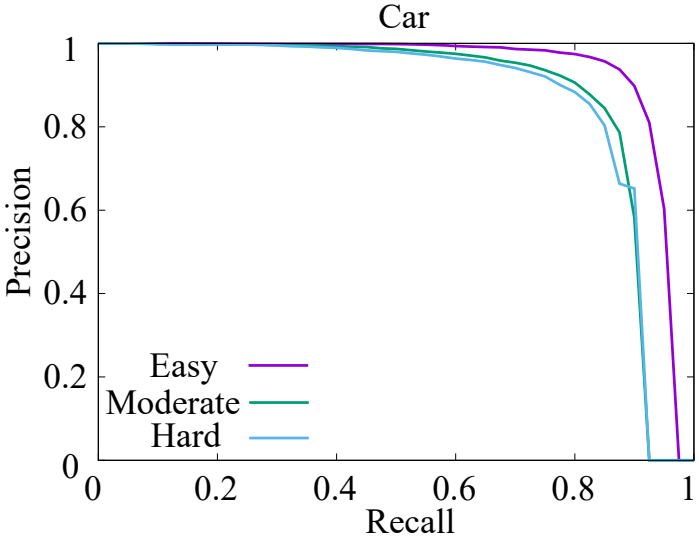
The proposed model is trained and evaluated on the KITTI dataset (Geiger, Lenz, and Urtasun, 2012). The KITTI object dataset possesses 7,481 training frames and 7,518 testing frames. Each frame is comprised of a point cloud, stereo RGB images, and calibration data. In this research, only a point cloud and the left image with



(A) 2-D



(B) 3-D



(C) BEV

FIGURE 3.8: Visualization of the Precision-Recall curve for the car class. From up to down are the curves of 2D, 3D, and BEV. In each curve, each color line denotes one difficulty of the car class. The curves are drawn according to the best-proposed model.

their calibration data are used. The KITTI includes seven classes: car, van, truck, pedestrian, person (sitting), cyclist, and tram. Because the number of other categories is small, the car, pedestrian, cyclist classes are used for comparison. MV3D (Chen et al., 2017) is the pioneer in the multi-modal 3D object detection and divides the training dataset into two subsets (ratio=3,712:3,769): the training subset and the validation subset. To compare fairly with its results, among the subsequent articles employ the same criteria as MV3D. Two models are trained for the car class and pedestrian/cyclist classes, respectively, since the training dataset has an unbalanced amount of training data for the car class and the pedestrian/cyclist classes.

KITTI's object detection metric is defined as 11-point Average Precision (AP). Intersection-over-Union (IoU) is the generic evaluation criterion for object detection. In the evaluation of 2D, 3D, and BEV detection, 2-D IoU is at the threshold of 0.7 for the car class and 0.5 for the pedestrian/cyclist classes. According to the bounding box height, truncation levels, and occlusion classes of objects, KITTI groups all objects into three difficulty classes: easy (E), moderate (M), and hard (H). In the evaluation, prediction results are evaluated by the program that comes with the KITTI dataset, and the program outputs the three results for the easy, moderate, and hard class, respectively.

3.3.2 Implementation Details

Since the 2D RGB camera images are of different sizes, the images are center-cropped into a uniform size of 1200×360. Each point cloud is voxelized as a $700\times800\times6$ BEV pseudo image. The proposed model is implemented using TensorFlow on one NVIDIA 1080 Ti GPU with a batch size of 1. Adam is the optimizer. Our model is trained for a total of 120K iterations with the initial learning rate of 0.0001, and decayed by 0.1 at 60K iterations and 90K iterations. The whole training process takes only 12 hours, and the proposed model is evaluated from 100K iterations to 120K iterations every 5K iterations.

3.3.3 Comparison with State-of-the-Art Methods

All experiments are evaluated on the KITTI validation subset. It should be noted that no currently published one-stage method publicly provides results on the pedestrian/cyclist classes for 3D object detection based on LiDAR and RGB images. Hence, the comparison is only for the car class in Table 3.1. All methods are grouped into three sets: one-stage methods, two-stage methods, and three-stage methods based on LiDAR and image. For a fair comparison, this article only compares with the state-of-the-art methods in the past five years that use LiDAR and images as input. Most of the methods only disclose 3D and BEV performance. Thus, 2D detection performance is not listed in Table 3.1.

In the 3D object detection, our proposed method outperforms all state-of-the-art methods with noticeable margins except for a slightly lower score than Cont-Fuse (Liang et al., 2018) in the 'E' column. Specifically, our proposed method achieves 1.04% gains on the most important 'M' column compared with the second-best performing method, and also outperforms the second-ranked MCF3D (Wang et al., 2019) by 0.76% on the mean average precision (mAP). In BEV object detection, the overall performance of Cont-Fuse (Liang et al., 2018) is better than ours, but our method outperforms Cont-Fuse (Liang et al., 2018) in the 'H' column with a big margin of 3.17%. For the inference time, the proposed method still achieves a comparable speed by taking the high precision into account. The precision-recall curves of the best-proposed model are shown in Figure 3.8.

To further understand the proposed model, Table 3.2 shows the number of parameters for each component in the proposed framework.

TABLE 3.2: The number of parameters for each component.

Component	Total	Base Network	GFA	RA
Number of Parameters	20,575,616	15,852,928	4,718,592	4,096

Since the state-of-the-art methods, shown in Table 3.1, do not provide the pedestrian/cyclist classes results, the pedestrian/bicycle results cannot be compared with other methods. This paper provides the results of pedestrian/bicycle for reference in Table 3.3.

TABLE 3.3: The 3D and BEV object detection accuracy for the pedestrian and cyclist classes. To ensure the best visual effect, the table does not show the 'Easy (E)' results.

Class	2D (%)		3D (%)		BEV (%)	
	M	H	M	H	M	H
Pedestrian	58.44	52.11	65.39	59.29	65.47	59.38
Cyclist	43.58	38.97	43.23	38.31	43.27	38.37

3.3.4 Ablation Study

In this section, massive experiments are utilized for analysis and ablation of the proposed model on the KITTI validation subset. To ensure the best visual effect, all tables do not show the 'Easy (E)' results.

Figure 3.9 shows the qualitative results of 3D object detection. To more directly compare the prediction results with the ground truth values, the green and red color represent the ground truth and prediction, respectively. It can be seen that our method can detect and localize 3D objects well. Compared with the pedestrian/bicycle results, the performance of the car detection is much better due to the larger size of the cars.

Effect of Global Feature Attention. Table 3.4 shows how does the GFA affects 3D performance. Two feature encoders are used to extract the BEV features and the RGB image features, respectively. This paper explores the impact of the GFA on RGB image features and BEV features, respectively. Relative to the BEV features (the second row), the GFA is more helpful for RGB image feature extraction (the first row) with 1.06% gains in the moderate class. In addition, one more experiment is employed to analyze the effect of the auxiliary convolutional block in section 3.2.4. Compared with the results in the last row, the auxiliary convolutional block (the third row) achieves a 1.42% and 5.56% gains in the moderate and hard classes of 3D performance, respectively. The reason is that the convolutional layer further merges the fused features.

Effect of Diversity Combination. Table 3.5 shows the combinations of different proposed methods and their corresponding performances. Among the four approaches, in terms of a single method, FPN contributes a maximum of 0.95% in 3D performance, because the FPN integrates the high-level semantic feature maps at all scales. Besides, RGB^D images contribute to BEV detection with a 0.89% increase. Compared

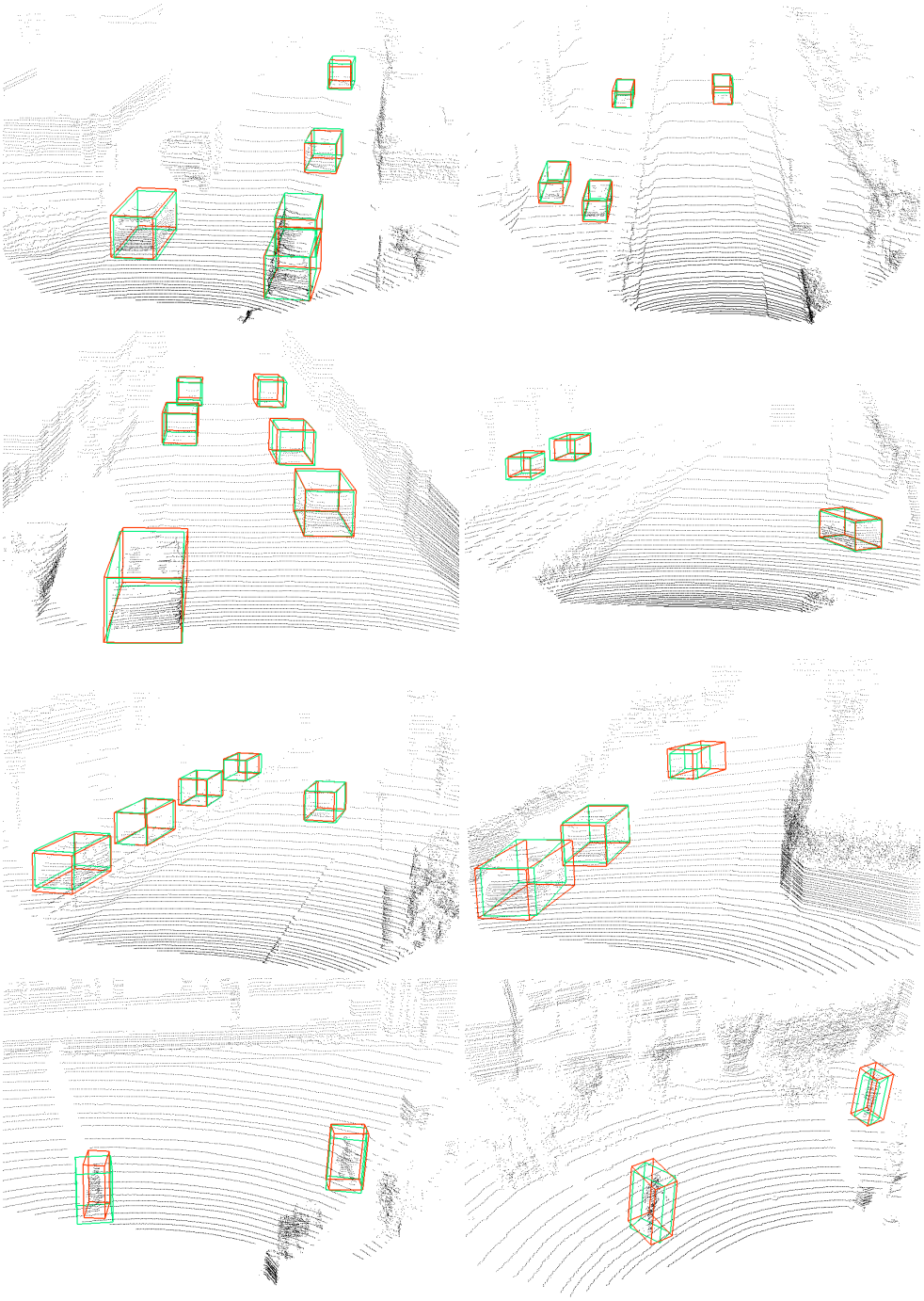


FIGURE 3.9: Visualizations of 3D detection results on point clouds. The green color denotes the ground truth and the red color represents the prediction. The first two rows are the results of the car class. The last row is the results of cyclist and pedestrian classes.

with the RA (the third row), the GFA (the fourth row) is more helpful to boost 3D performance due to it enhances the global feature representations. Each proposed method only slightly improves the detection performance, however, the proposed framework greatly boosts the detection accuracy based on all proposed methods. Compared with the baseline (the first row), the best-proposed combination (the last

TABLE 3.4: The effect of global feature attention. 'Where use it?' means which feature encoder uses the GFA. The third and the fourth rows show the effect of the auxiliary convolutional block. The best performance is highlighted in bold for the 3D column.

Where use it?	2D (%)		3D (%)		BEV (%)	
	M	H	M	H	M	H
RGB Image	87.85	86.82	74.71	68.46	85.61	79.41
BEV	87.87	86.75	73.65	67.94	85.56	79.31
RGB Image+BEV	88.18	86.53	**75.87**	**73.98**	85.35	85.44
w/o Conv.	87.61	86.27	74.45	68.42	85.08	85.25

row) achieves 1.48%, 3.31%, and 1.15% gains in 2D, 3D, and BEV, respectively. As can be seen, the proposed methods are quite useful for boosting 3D performance.

TABLE 3.5: The effect of different proposed methods. 2D, 3D, and BEV performance are compared on the 'Moderate' difficulty for the car class. *FPN* denotes the feature pyramid network. The best performance is highlighted in bold for each column.

Method Combinations				2D (%)	3D (%)	BEV (%)
FPN	RA	GFA	RGB^D			
				86.99	72.92	85.08
√				87.42	73.87	85.36
	√			87.67	73.22	85.30
		√		87.20	73.34	85.89
√	√			87.43	74.55	85.52
√		√		87.78	73.11	85.20
	√	√		87.55	74.07	85.17
√	√	√		88.18	75.87	85.35
√	√	√	√	**88.47**	**76.23**	**86.23**

Effect of RGB^D image. Based on the best combination in Table 3.5, three sets of experiments are used to study the effect of RGB, RGB-I, and RGB^D for the car class, respectively. Table 3.6 shows that RGB^D surpasses the other two images in all aspects. Compared with the RGB image, the RGB-I has very limited performance improvement for the model, and even worse than the RGB image in most performances. In terms of 3D performance, the RGB^D achieves 0.36%, 0.48% gains in the moderate, and hard difficulty, respectively. The experimental results verify that the RGB^D indeed preserves more 3D information of a point cloud.

Effect of Region-of-Interest Attention. The RA is a soft attention network for the fusion of paired view-specific ROIs. First, the input size of the RA is analyzed. Taking the efficiency factor into account, three kinds of input sizes are utilized for analysis. The experimental results are shown in Table 3.7. As can be seen, the input size 7×7

TABLE 3.6: The comparison of RGB, RGB-I, and RGB^D. The best performance is highlighted in bold for the 3D column.

Class	2D (%)		3D (%)		BEV (%)	
	M	H	M	H	M	H
RGB	88.18	86.53	75.87	73.98	85.35	85.44
RGB-I	88.24	86.94	75.46	68.94	85.48	79.38
RGB^D	**88.47**	**86.98**	**76.23**	**74.46**	**86.23**	**85.60**

is the best candidate for efficiency and 3D detection accuracy. This may be due to the three reasons:

1. Most of the proposals with an approached size, 7×7;
2. If the cropped feature map based on the proposal is resized to a small size, such as 5×5, and the important features may be lost;
3. If the cropped feature map is resized to a large size, the important feature may be diluted.

TABLE 3.7: The effect of input size for the RA. The best performance is highlighted in bold for the 3D column.

RA size	2D (%)		3D (%)		BEV (%)	
	M	H	M	H	M	H
5×5	87.68	86.40	73.10	67.57	85.37	79.30
7×7	87.43	86.84	**74.55**	**68.44**	85.52	79.37
9×9	87.49	86.31	74.28	68.05	85.47	79.44

MV3D (Chen et al., 2017) employs the concatenation operation to fuse view-specific ROIs. AVOD (Ku et al., 2018) exploits the addition operation for fusion. The concatenation as comparison to the addition operation benefits to boost performance, but the performance improvement is a little. Differ from these two operations, the proposed RA pays more attention to the important features through learning. In 3D performance of moderate difficulty class, RA achieves 0.68%, 0.42% gains as compared to the Addition and Concatenation operation, respectively, as shown in Table 3.8.

Table III-VIII show that the 2D IoU metric is more helpful for 2D and BEV performance. It ignores the impact of proposals' height during the stage of proposal generation, and it is a drawback for 3D object detection accuracy. Note that the proposed method achieves the best performance in 2D and BEV, hence, the 3D performance is only compared in Table III-VIII.

TABLE 3.8: The effect of the different fusion methods. The best performance is highlighted in bold for the 3D column.

Fusion Method	2D (%)		3D (%)		BEV (%)	
	M	H	M	H	M	H
Addition	87.42	86.63	73.87	68.25	85.36	79.20
Concatenation	87.85	86.90	74.13	67.98	85.79	79.22
RA	87.43	86.84	**74.55**	**68.44**	85.52	79.37

3.4 Conclusion and Future Works

This paper proposes a one-stage 3D object detection framework based on LiDAR and Camera, which benefits from the three attention mechanisms to boost 3D detection accuracy. First, the height attention mechanism is introduced into the input RGB images to generate the RGB^D images. Second, the global feature attention mechanism is utilized for both the RGB image and the BEV branches at the feature extraction stage. It benefits by capturing the discriminative features from both the channels and spatial dimensions. Finally, the region-of-interest attention mechanism is employed to fuse the paired view-specific ROIs. Our proposed method greatly improves the 3D object detection performance, and it outperforms all state-of-the-art methods based on LiDAR and Camera in 3D object detection.

In the future, the way to generate BEV images can be replaced with a learnable feature generator like SECOND (Yan, Mao, and Li, 2018). Additionally, the method of anchor generation can be changed from anchor-based to anchor-free. In this way, with the advantages of RGB images and LiDAR point clouds, 3D object detection based on LiDAR and camera can achieve better performance than LiDAR-based methods.

Chapter 4

Fast and Accurate 3D Object Detection for Lidar-Camera-Based Autonomous Vehicles Using One Shared Voxel-Based Backbone

This paper proposes a novel point-wise fusion strategy between point clouds and RGB images. The proposed method directly extracts pointwise features from the raw RGB image based on the raw point cloud first. Then, it fuses the two pointwise features and feeds them into a 3D neural network. The structure, as shown in Figure 4.1, has only one backbone to extract features, making the proposed model much faster than state-of-the-art LiDAR and camera fusion methods.

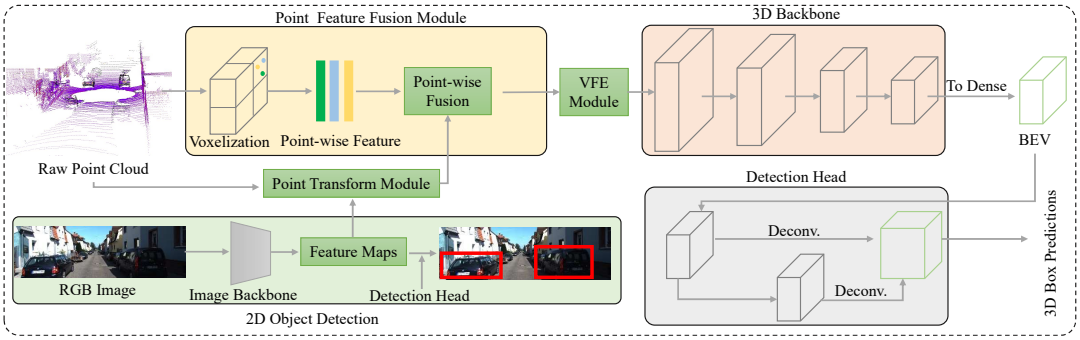


FIGURE 4.1: The architecture of the proposed one-stage 3D object detection network for the LiDAR and camera. It mainly includes the input data, the point feature fusion module, the 3D backbone, and the detection head. The gray box and green box represent the convolutional block and feature map, respectively.

The key contributions of this work are as follows:

- This paper presents an early-fusion method to exploit both LiDAR and camera data for fast multi-class 3D object detection with only one backbone, achieving a good balance between accuracy and efficiency.
- This paper proposes a highly-efficient pointwise feature fusion module, which directly extracts the RGB image point feature based on a point cloud and fuses the extracted RGB image point feature with the corresponding feature of the point cloud.
- This paper also enhances 3D object detection with an RGB^+ image, which preserves the information projected from its corresponding point cloud.

The presented one-stage 3D multi-class object detection framework outperforms state-of-the-art LiDAR-camera-based methods on the KITTI benchmark (Geiger, Lenz, and Urtasun, 2012) both in terms of the speed and accuracy.

4.1 Related Work

This section starts by reviewing recent works in applying convolutional neural networks (CNNs) to 3D object detection based on LiDAR, and then focuses on methods specific to multi-modal 3D object detection from point clouds and RGB images.

4.1.1 LiDAR-based 3D Object Detection

Recently, there have been three main 3D object detectors based on LiDAR: voxel-based detectors, point-based detectors, and graph-based detectors. Voxel-based methods (Zhou and Tuzel, 2018; Wen and Jo, 2019; Yan, Mao, and Li, 2018; Lang et al., 2019) first voxelize the raw point cloud over a given range and then utilize a 3D CNN or 2D CNN to extract features. Unlike VoxelNet (Zhou and Tuzel, 2018), Yan et al. (Yan, Mao, and Li, 2018) replaced a 3D CNN by a 3D sparse convolutional network, and Lang et al. (Lang et al., 2019) directly organized point clouds in vertical columns (pillars) to generate 2D BEV images. Point-based detectors (Charles et al., 2017; Qi et al., 2017; Yang et al., 2020; Wen, Vo, and Jo, 2020) directly deal with the raw point cloud. Charles et al. (Charles et al., 2017) pioneered the method used to deal with each point independently using their shared MLPs. Based on PointNet (Charles et

al., 2017), Charles et al. (Qi et al., 2017) further introduced the metric space distances to learn local features with increasing contextual scales. Yang et al. (Yang et al., 2020) abandoned the upsampling layers in PointNet++ to boost the inference speed. The proposed method voxelizes a point cloud using a dynamic voxelization method compared with the hard voxelization method in (Zhou and Tuzel, 2018) and aims to avoid information loss during voxelization.

4.1.2 Multi-modal 3D object detection

3D Object detection in point clouds and RGB images is a fusion problem. As such, it is natural to extract the RGB image feature and the point cloud feature with two different backbones, respectively, which is the paradigm present in all previous works (Chen et al., 2017; Ku et al., 2018; Wen and Kang-Hyun, 2020; Wen and Jo, 2021; Xu, Anguelov, and Jain, 2018; Sindagi, Zhou, and Tuzel, 2019; Qi et al., 2018; Liang et al., 2018; Wang et al., 2019; Liang et al., 2019). Obviously, by employing two heavy backbones, these approaches are very slow and consume a great deal of memory. In the paradigm, these methods are designed to either study how to fuse or how to improve accuracy based on state-of-the-art fusion methods, e.g., AVOD (Ku et al., 2018) changes the feature generation method in MV3D (Chen et al., 2017) from hand-crafted techniques to automation to improve the running speed of the model. According to different fusion methods, these methods can be divided into two categories: pointwise fusion (Xu, Anguelov, and Jain, 2018; Sindagi, Zhou, and Tuzel, 2019) and region of interest (ROI)-based fusion (Chen et al., 2017; Ku et al., 2018; Qi et al., 2018; Liang et al., 2018; Wang et al., 2019; Liang et al., 2019; Yoo et al., 2020). Compared with the ROI-based fusion, pointwise fusion is more flexible. Inspired by pointwise fusion, this article will explore whether it is possible to directly aggregate the point features of the raw RGB image with point cloud features. Different from the previous methods, the proposed method only has one backbone. Additionally, the proposed model takes the RGB^+ image as the input, instead of using an RGB image.

In this paper, we first present an early-fusion method to exploit both LiDAR and camera data for fast 3D object detection with only one backbone, and it achieves a good balance between accuracy and efficiency. Thanks to the novel pointwise feature fusion module, which makes the fusion between LiDAR and camera data high

efficient. To further improve the detection performance, we propose the RGB^+ image as the input.

4.2 Proposed Approach

The proposed model, as shown in Figure 4.1, takes point clouds and RGB images as inputs and predicts oriented 3D bounding boxes for cyclists, pedestrians, and cars. This model includes four main parts: (1) A point feature fusion module that extracts the point features from the RGB image and fuses the extracted features with the corresponding point cloud features, (2) a voxel feature encoder (VFE) module and a 3D backbone to process the fused pointwise features into a high-level representation, (3) a detection head that regresses and classifies the 3D bounding boxes, and (4) a loss function.

4.2.1 Point Feature Fusion Module

The fusion module, shown in Figure 4.2, consists of three submodules: the point transform module, the voxelization of point clouds, and the pointwise fusion module. Since this module involves the input of raw data, before introducing the module, the input data is first introduced.

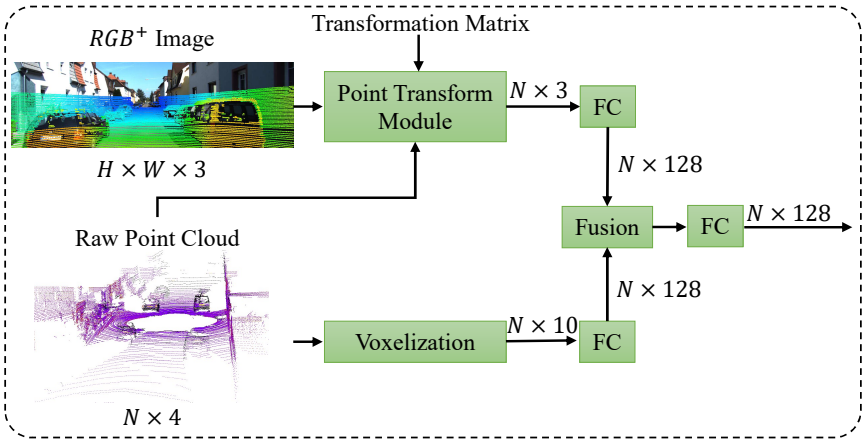


FIGURE 4.2: Visualization of the point feature fusion module. N is the number of points in a point cloud, and FC denotes one fully connected layer.

Input Data. This model accepts point clouds and RGB images as the input. To reduce the loss of raw point-cloud information during voxelization, a LiDAR point

cloud is projected onto an RGB image and embedded into the image to generate a new image with three channels, called RGB^{+}. The RGB^{+} object has two typical representations: the RGB$^{\mathbf{I}}$ portion that embeds the intensity of point clouds into an RGB image, and the RGB$^{\mathbf{D}}$ representation that embeds the Z-axis value of point clouds into the image. The detailed process of the RGB^{+} generation is divided into the three following steps:

(1) First, point clouds (X, Y, Z) are mapped onto the original image $(W \times H)$ plane as follows:

$$\begin{pmatrix} u & v & 1 \end{pmatrix}^T = \mathbf{M} \cdot \begin{pmatrix} X & Y & Z & 1 \end{pmatrix}^T, \tag{4.1}$$

$$\mathbf{M} = \mathbf{P}_{rect} \cdot \begin{pmatrix} \mathbf{R}_{velo}^{cam} & \mathbf{t}_{velo}^{cam} \\ 0 & 1 \end{pmatrix}, \tag{4.2}$$

where (u, v) is the image coordinate, $\mathbf{P}_{rect}$ is a project matrix, $\mathbf{R}_{velo}^{cam}$ is the rotation matrix from LiDAR to the camera, $\mathbf{t}_{velo}^{cam}$ is a translation vector, and $\mathbf{M}$ is the homogeneous transformation matrix from LiDAR to the camera.

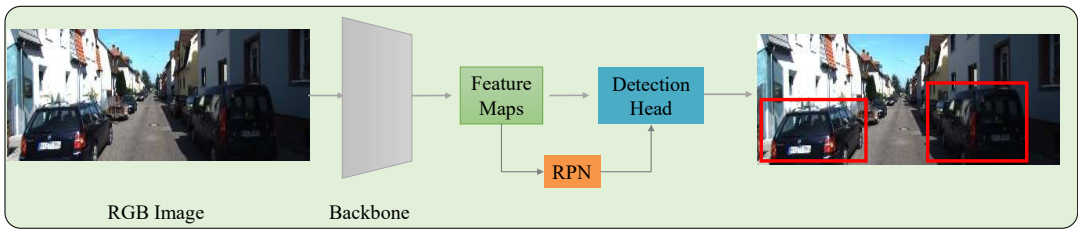
(2) Second, the points $\{(x, y, z) \mid x \in X, y \in Y, z \in Z\}$ located in the image of size $W \times H$ are kept. Meanwhile, the LiDAR points are projected to the camera coordinates and denoted as (x_c, y_c, z_c):

$$\begin{pmatrix} x_c & y_c & z_c \end{pmatrix}^T = \mathbf{M} \cdot \begin{pmatrix} x & y & z & 1 \end{pmatrix}^T. \tag{4.3}$$

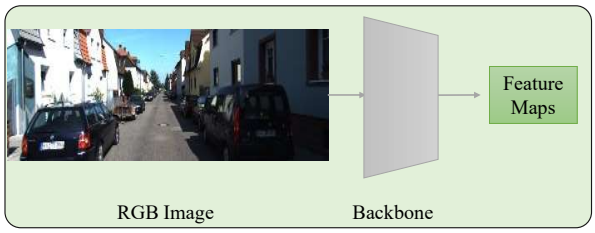
(3) Finally, z_c is mapped between 0 and 255 and then assigned to the corresponding image coordinate (u, v) to generate the RGB$^{\mathbf{D}}$ object. Similarly, the intensity of the point cloud for each color channel is mapped between 0 and 255 and then assigned to the corresponding image coordinate (u, v) to obtain the RGB$^{\mathbf{I}}$ data structure. This process uses the circle function of the OpenCV library.

Point Transform Module. This module extracts point features from the RGB^{+} image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$ based on the raw point cloud. First, a point cloud $\mathbf{P} \in \mathbb{R}^{N \times 3}$ is projected onto its corresponding image by Eq. 4.1 to obtain the corresponding image coordinates (u_i, v_i). Second, the RGB^{+} and the (u_i, v_i) are fed into the image sampler (Jaderberg et al., 2015), outputting the image point feature $\mathbf{P_i} \in \mathbb{R}^{N \times 3}$, where N is the number of points in the point cloud.

In this paper, we explore which image features are better for fusion. There three kinds of features: the fine features from Faster R-CNN, the coarse features from a backbone, and the features from raw image features, as shown in Figure 4.3. In ablation study, we further show more details, as shown in Table 4.5.



(A) Fine image features from Faster R-CNN



(B) Coarse features from a backbone

(C) Raw RGB image

FIGURE 4.3: Three kinds of image features for fusion.

Voxelization. Voxelization divides the point cloud into evenly spaced voxel grids and then generates a many-to-one mapping between 3D points and their corresponding voxels. The details are shown in Figure 4.4. Currently, there exist two voxelization methods: hard voxelization(Zhou and Tuzel, 2018) and dynamic voxelization (Zhou et al., 2020). Compared with the former, dynamic voxelization makes the detection more stable by preserving all the raw points and voxel information. This work applies the dynamic voxelization method. Given a point cloud $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_N\}$, the process assigns N points to a buffer of size $N \times F$, where N is the number of points and F denotes the feature dimension. Specifically, each point $p_i = [x_i, y_i, z_i, r_i]$ (containing the XYZ coordinates and the reflectance value) in a voxel is denoted by its inherent information (x_i, y_i, z_i, r_i), its relative offsets (x_v, y_v, z_v) with respect to the centroid of the points in the voxel, and its relative offsets (x_p, y_p, z_p) with respect to the centroid of the points in the pillar. Finally, the output point-wise feature is $\mathbf{P_v} \in \mathbb{R}^{N \times 10}$, and the resulting size of the 3D voxel grid is $\left(\frac{W}{s_y}, \frac{H}{s_x}, \frac{D}{s_z}\right)$, where (s_y, s_x, s_z) gives the voxel sizes, and (W, H, D) are the ranges along the Y-axis, X-axis, Z-axis, respectively.

Point-wise Fusion. This module fuses the pointwise features $\mathbf{P_i}$ and $\mathbf{P_v}$. Since the dimensions of the two features are different, two fully connected (FC), one for each

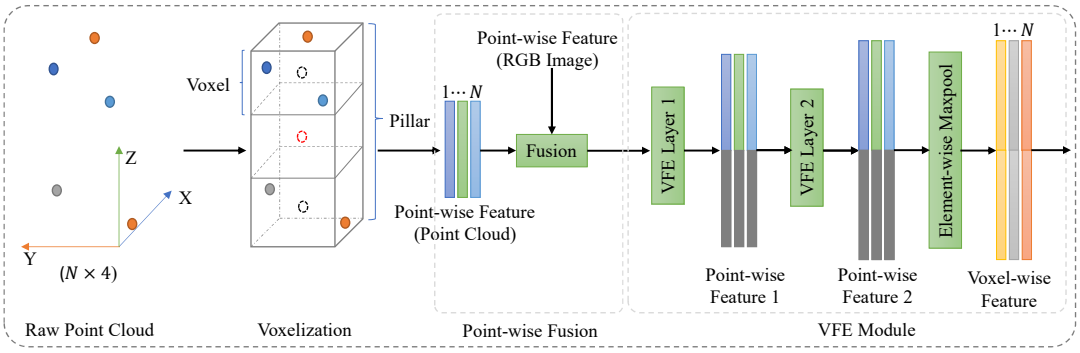


FIGURE 4.4: The details for the voxelization, pointwise fusion, and VFE module. The black circle and red circle are the centroid of a voxel and a pillar, respectively. The black circle and red circle are only for demonstration and are all virtual. To understand the figure more clearly, the number of points N only takes the value of five.

feature, are used to adjust their dimensions to be the same. There are two common fusion methods for ROIs: addition and concatenation. Therefore, this paper will analyze which fusion method is the most suitable for the pointwise features in Table 4.3 in the ablation section. After the fusion operation, one FC layer is utilized to further merge the fused features and output the result as $\mathbf{P_f}$.

4.2.2 Voxel Feature Encoder Module

This section introduces the voxel feature encoder module and the 3D backbone, in that order.

Upon completing the pointwise fusion, the fused feature $\mathbf{P_f}$ is transformed through the VFE layer which is composed of a fully connected network (FCN), into a feature space, where information from the point features $\mathbf{f_i} \in \mathbb{R}^m$ can be aggregated to encode the shape of the surface contained within the voxel (Zhou and Tuzel, 2018; Wen and Jo, 2019; Sindagi, Zhou, and Tuzel, 2019), where $i \in [1, N]$ and m is the feature dimension of a point. The FCN consists of a linear layer followed by a batch normalization layer, and a ReLU layer. An elementwise max-pooling process is used to locally aggregate the transformed features and output a feature $\vec{\mathbf{f}}$ for $\mathbf{P_f}$. Finally, the max-pooled feature $\vec{\mathbf{f}}$ is concatenated with each point feature $\mathbf{f_i}$ to generate the final feature $\mathbf{P_{vfe}}$. his work stacks two such VFE layers and both of the output lengths are 128. This means the shape of $\mathbf{P_{vfe}}$ is $N \times 128$. The details are shown in Figure 4.4.

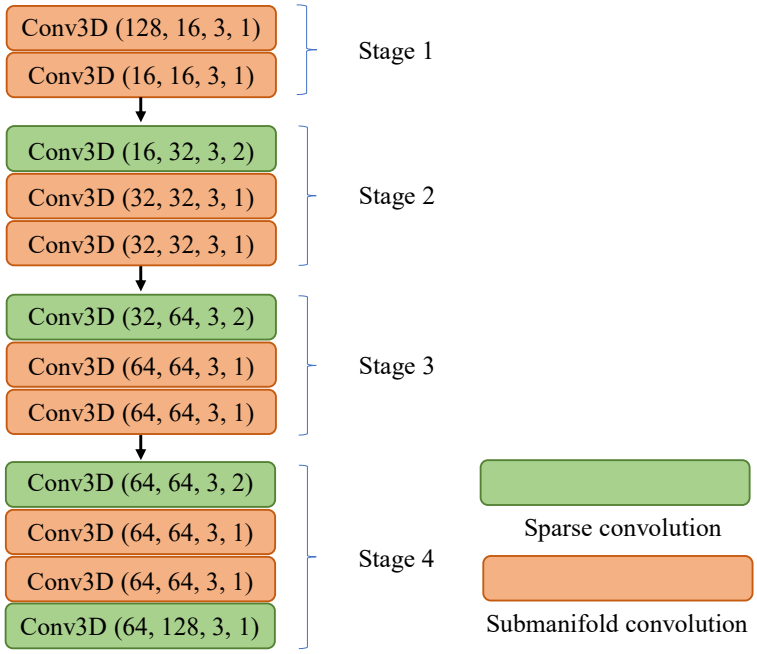


FIGURE 4.5: The 3D backbone architecture. Conv3D (cin, cout, k, s) denotes a convolutional block, where the parameters cin, cout, k, and s represent the input-channel numbers, the output-channel numbers, the kernel size, and the stride, respectively. Each block consist of a 3D convolutional layer followed by a batch normalization layer and a ReLU layer.

4.2.3 3-D Sparse Convolutional Backbone

The 3D backbone takes the feature $\mathbf{P_{vfe}}$ and it's corresponding index of 3D coordinates $(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ as inputs. The backbone is widely used in (He et al., 2020; Shi et al., 2020) and has twelve 3D sparse convolutional layers and is divided into four stages according to feature resolution, as shown in Figure 4.5. The four-stage feature resolutions in the order of (W, H, D) are $(1600, 1408, 41)$, $(800, 704, 21)$, $(400, 352, 11)$, and $(200, 176, 2)$. Specifically, each stage has two kinds of 3D convolutional layers: the submanifold convolution (Yan, Mao, and Li, 2018) and the sparse convolution. The former does not generate new points and shares the point coordinate indices in each stage; hence, the submanifold convolution runs very fast. The latter is a sparse version of the dense 3D convolution. Usually, these two convolutions are used in conjunction to achieve the speed/accuracy balance. The details and numbers of input and output channels are illustrated in Figure 4.5. The sparse feature map after the 3D sparse convolution needs to be converted into the dense feature map $\mathbf{F_d} \in \mathbb{R}^{200\times176\times256}$. The detailed configuration is given in Table 4.1.

4.2.4 Detection Head

The input data of the detection head is the dense feature map $\mathbf{F_d}$. The detection head is comprised of three convolution blocks. Block 1 has five 2D convolutional layers and outputs the feature map $\mathbf{F_1} \in \mathbb{R}^{100\times88\times128}$. Similarly, block 2 also has five 2D convolutional layers and takes the feature map $\mathbf{F_1}$ as input and outputs the feature map $\mathbf{F_2} \in \mathbb{R}^{50\times44\times256}$. Block 3 has two transpose layers and one 2D convolutional layer. $\mathbf{F_1}$ and $\mathbf{F_2}$ are transposed as the feature map $\mathbf{F_3} \in \mathbb{R}^{100\times88\times256}$ and the feature map $\mathbf{F_4} \in \mathbb{R}^{100\times88\times256}$, respectively. Finally, the feature maps $\mathbf{F_3}$ and $\mathbf{F_4}$ are concatenated as the feature map $\mathbf{F} \in \mathbb{R}^{100\times88\times512}$. The feature map $\mathbf{F}$ is mapped to three desired learning targets: (1) a classification score map $\mathbf{F_{score}} \in \mathbb{R}^{100\times88\times18}$, (2) a box regression map $\mathbf{F_{box}} \in \mathbb{R}^{100\times88\times42}$, and (3) a direction regression map $\mathbf{F_{dir}} \in \mathbb{R}^{100\times88\times12}$. The detailed configuration is given in Table 4.1.

TABLE 4.1: The network configuration for the 3D backbone and the detection head. The output sizes (W, H, Depth) and (W, H, Channel) are for the 3D backbone and the detection head, respectively. The structure $[type, size, stride] \times Number$ represents the convolutional type, filter size, stride, and the number of layers.

Network	Output Size	Name	Layer
3D Backbone	(1600, 1408, 41)	Stage1	S_Conv3D, 3, s1 Sub_Conv3D, 3, s1
	(800, 704, 21)	Stage2	S_Conv3D, 3, s2 Sub_Conv3D, 3, s1 Sub_Conv3D, 3, s1
	(400, 352, 11)	Stage3	S_Conv3D, 3, s2 Sub_Conv3D, 3, s1 Sub_Conv3D, 3, s1
	(200, 176, 2)	Stage4	S_Conv3D, 3, s2 Sub_Conv3D, 3, s1 Sub_Conv3D, 3, s1 Sub_Conv3D, 3, s1
Detection Head	(100, 88, 128)	Block1	[Conv2D, 3, s1]x4 Conv2D, 3, s2
	(50, 44, 256)	Block2	[Conv2D, 3, s1]x4 Conv2D, 3, s1
	(100, 88, 512)	Block3	DeConv2D, 3, s1 DeConv2D, 3, s2 Conv2D, 3, s1

4.2.5 Loss Function

This work utilizes the same loss functions in PointPillars (Lang et al., 2019) and SECOND (Yan, Mao, and Li, 2018). The 3D ground truth boxes and anchors are parameterized as $(x, y, z, l, w, h, \theta)$, where (x, y, z) denote the box's center, (l, w, h) represent the box's size, and θ is the yaw rotation around the Z-axis. The corresponding regression residuals between the 3D anchors and ground truth are defined as follows:

$$\begin{aligned} \Delta x &= \frac{x^g - x^a}{d^a}, & \Delta y &= \frac{y^g - y^a}{d^a}, \\ \Delta z &= \frac{z^g - z^a}{h^a}, & \Delta l &= log\left(\frac{l^g}{l^a}\right), \\ \Delta w &= log\left(\frac{w^g}{w^a}\right), & \Delta h &= log(\frac{h^g}{h^a}), \\ \Delta \theta &= sin(\theta^g - \theta^a), \end{aligned} \tag{4.4}$$

where the superscripts g and a represent the ground truth box and the anchor, respectively. The variable $d^a = \sqrt{(w^a)^2 + (l^a)^2}$ is the diagonal of the base of the anchor box.

The regression loss function is as follows:

$$\mathcal{L}_{reg} = \sum_{b} Smooth_{\mathcal{L}1}(\Delta b), \tag{4.5}$$

where the input dimensions are $b \in (x, y, z, w, l, h, \theta)$ and $Smooth_{\mathcal{L}1}$ is the smooth $\mathcal{L}1$ loss function in the Fast R-CNN module.

Since the yaw angle $\theta \in [-\Pi, \Pi]$ has two directions $\{+, -\}$, and the angle regression loss cannot distinguish the directions. A softmax classification loss is utilized to compute the discretized direction loss (Yan, Mao, and Li, 2018), $\mathcal{L}_{dir}$. If the yaw angle θ around the Z-axis of the ground truth is greater than zero, the direction is positive; otherwise, the direction is negative.

For the object classification loss, the focal loss (Lin et al., 2020) is used:

$$\mathcal{L}_{cls} = -\alpha_a (1 - p^a)^\gamma log(p^a), \tag{4.6}$$

where p^a is the class probability of an anchor, $\alpha = 0.25$, and $\gamma = 2$. The total loss can be formulated as follows:

$$\mathcal{L}oss = \frac{1}{N_{pos}}(\beta_1 \mathcal{L}_{box} + \beta_2 \mathcal{L}_{cls} + \beta_3 \mathcal{L}_{dir}), \tag{4.7}$$

where N_{pos} is the number of positive anchors and $\beta_1 = 2.0$, $\beta_2 = 1.0$, and $\beta_3 = 0.2$. For the car class, an anchor is defined as positive if it has a 2D IoU greater than 0.60 (pedestrian/cyclist is 0.35) with its paired ground truth. If it has a 2D IoU less than 0.45 (pedestrian/cyclist is 0.2), the anchor is labeled as negative. The other anchors are ignored when computing the loss.

4.3 Experiments

This section introduces the dataset, the experimental settings, and the results in detail.

4.3.1 Dataset and Metrics

The proposed model is trained and evaluated on the KITTI dataset (Geiger, Lenz, and Urtasun, 2012). The KITTI object dataset possesses 7,518 testing frames and 7,481 training frames. Each frame is comprised of a point cloud, stereo RGB images (the left image and the right image), and calibration data. In this research, only a point cloud and the left image with their calibration data are used. To impartially compare the proposed approach with existing methods, the training dataset is divided into two subsets (training subset and validation subset) based on the same criteria, and the ratio of the two subsets is 1:1.

For KITTI's criteria, according to the size, truncation, and occlusion classes of objects, all objects are grouped into three difficulty classes: easy (E), moderate (M), and hard (H). Before October 8th, 2019, KITTI's object detection metric was defined as the 11-point average precision (AP) metric. Since then, the metric has been defined by 40 recall positions. Compared with the 11-point AP, the 40-point AP more properly assesses the quality of an algorithm based on the infinite approximation. Intersection-over-Union (IoU) is the generic evaluation criterion for object detection. In the evaluation of 2D, 3D, and bird's eye view (BEV) detection, the IoU is at the

TABLE 4.2: Performance comparison using the KITTI validation dataset. The results are evaluated by the mean Average Precision with 11 recall positions. For easy understanding, the top result is highlighted in bold for each column in each class and the second best is shown in blue. *I* and *L* denote the RGB image and LiDAR, respectively.

Class	Method	Publish Year	Input	FPS	3D Performance (%)			
					Easy	Moderate	Hard	mAP
Car	VoxelNet	2017	L	4.3	81.97	65.46	62.85	70.15
	SECOND	2018	L	20.0	87.43	76.48	69.10	77.67
	PointRCNN	2018	L	10.0	88.88	78.63	77.38	81.63
	F-PointRCNN	2019	L	15.4	89.12	79.00	77.48	81.87
	MV3D	2017	I+L	2.8	71.29	62.68	56.56	63.51
	F-PointNet	2017	I+L	5.9	83.76	70.92	63.65	72.78
	PC-CNN	2018	I+L	2.0	57.63	51.74	51.39	53.59
	AVOD	2018	I+L	12.5	83.11	74.02	67.84	74.99
	AVOD-FPN	2018	I+L	10.0	84.41	74.44	68.65	75.83
	ContFusion	2018	I+L	16.7	86.32	73.25	67.81	75.79
	MVX-Net	2019	I+L	6.7	85.50	73.30	67.40	75.40
	MCF3D	2019	I+L	6.3	84.11	75.19	74.23	77.84
	TAO3D	2020	I+L	9.0	85.12	76.23	74.46	78.60
	KDA3D	2020	I+L	7.7	**88.45**	**78.85**	**78.46**	**81.92**
	Proposed	-	I+L	**17.8**	88.04	77.60	76.23	80.62
Ped.	VoxelNet	2017	L	4.3	57.86	53.42	48.87	53.38
	AVOD-FPN	2018	I+L	10.0	-	58.80	-	-
	F-PointNet	2018	I+L	5.9	**70.00**	61.32	53.59	61.64
	MCF3D	2019	I+L	7.7	68.54	**64.93**	**59.47**	**64.31**
	KDA3D	2020	I+L	8.3	63.34	60.12	54.36	59.27
	Proposed	-	I+L	**17.8**	66.65	60.49	54.51	60.55
Cyc.	VoxelNet	2017	L	4.3	67.17	47.65	45.11	53.31
	AVOD-FPN	2018	I+L	10.0	-	49.70	-	-
	F-PointNet	2018	I+L	5.9	77.15	56.49	53.37	62.34
	MCF3D	2019	I+L	7.7	**78.18**	51.06	50.43	59.89
	KDA3D	2020	I+L	8.3	77.19	57.43	54.56	63.06
	Proposed	-	I+L	**17.8**	75.87	**60.07**	**55.87**	**63.94**

threshold of 0.7 for the car class and 0.5 for the pedestrian/cyclist class. For the average orientation similarity (AOS) we follow the approach in (Geiger, Lenz, and Urtasun, 2012) and define the AOS as:

$$AOS = \frac{1}{N} \sum_{r \in \{0, 0.1, \cdots, 1\}} \max_{\tilde{r}: \tilde{r} \geq r} s\left(\tilde{r}\right), \tag{4.8}$$

$$s\left(r\right) = \frac{1}{|D\left(r\right)|} \sum_{D(r)} \frac{1 + cos\Delta_{\theta}^{i}}{2} \delta_{i}, \tag{4.9}$$

where $N \in \{11, 40\}$, $r = \frac{TP}{TP+FN}$ is the PASCAL object detection recall, TP means the true positive, FN is the false negative, s is the orientation similarity, $D\left(r\right)$ represents the set of all object detections at recall, and Δ_{θ}^{i} is the difference in angle between estimated and ground truth orientation of detection i, $\delta \in \{0, 1\}$ is the penalty factor.

4.3.2 Experimental Settings

The proposed model is an end-to-end 3D detector for three classes: the car, pedestrian, and cyclist. When designing the anchors for the three classes, different classes employ different sizes (w, l, h). The sizes $(1.6, 3.9, 1.56)$, $(0.6, 0.8, 1.73)$, and $(0.6, 1.76, 1.73)$ are for the car, the pedestrian, and the cyclist, respectively. Note that each anchor has two directions $\{0^{\circ}, 90^{\circ}\}$, which means that each location has six anchors. The detection area in the point cloud is $\{(x, y, z) \mid x \in [0, 70.4], y \in [-40, 40], z \in [-3, 1]\}$.

The framework is based on Pytorch and programmed by the python language. This model is trained from scratch based on Adam optimizer. The whole network is trained with a batch of size 10 and the initial learning rate is 0.003 for 80 epochs on one TITAN RTX GPU. This work also adopts the cosine annealing learning rate for the learning rate decay. The entire training time is around 12 hours.

For data augmentation, this work employs the widely used augmentations found in (Zhou and Tuzel, 2018; Yan, Mao, and Li, 2018; Lang et al., 2019), including global scaling $[0.95, 1.05]$, global rotation around the Z-axis $[-45^{\circ}, 45^{\circ}]$, and the random flipping along the X-axis.

TABLE 4.3: Performance comparison using the KITTI testing dataset. The results of cars are evaluated by the mean Average Precision with 40 recall positions. The top performance is highlighted in bold only for the mAP columns and FPS column, and the second-best is shown in blue.

Method	FPS	AP_{BEV} (IoU = 0.7)				AP_{3D} (IoU = 0.7)		
		Easy	Moderate	Hard	mAP	Easy	Moderate	Hard
MV3D	2.8	86.00	76.90	68.50	77.13	71.10	62.40	55.10
F-PointNet	5.9	88.70	84.00	75.30	82.67	81.20	70.40	62.20
AVOD	12.5	86.80	85.40	77.70	83.30	73.60	65.80	58.40
AVOD-FPN	10.0	88.50	83.80	77.90	83.40	81.90	71.90	66.40
ContFusion	16.7	94.07	85.35	75.88	**85.10**	83.68	68.78	61.67
MVX-Net	6.7	89.20	85.90	78.10	84.40	83.20	72.70	65.20
Proposed	**23**	89.61	85.08	80.42	85.04	81.11	72.93	67.24

4.3.3 Results and Analysis

Most of the LiDAR-camera-based methods only provide the results in the KITTI validation dataset for three classes, hence, this work first compares the results in the validation dataset. In addition, for the car class, this paper also compares the results based on the KITTI testing dataset.

This work achieves competitive results compared with other state-of-the-art methods, the details are illustrated in Table 4.2. The results are mainly compared with the LiDAR and RGB camera-based methods. Usually, the LiDAR-based methods run much faster than the LiDAR-camera-based approaches. To show the superiority of the proposed model in speed, the classic LiDAR-based methods are also listed in Table 4.2. As can be seen, the proposed model mainly competes with MCF3D (Wang et al., 2019) and KDA3D (Wang et al., 2020) in comprehensive performance. For the cyclist class, the proposed model outperforms the KDA3D (Wang et al., 2020). In the car class, our model is slightly inferior to the KDA3D (Wang et al., 2020). However, the speed of our model runs 2× faster than KDA3D. Note that the proposed model is an end-to-end multi-class detector, however MCF3D (Wang et al., 2019) and KDA3D (Wang et al., 2020) train two models for the car class, and the pedestrian/cyclist classes, respectively. F-PointNet (Qi et al., 2018) is actually a LiDAR-based method that utilizes the location of the object in the 2D RGB image to quickly guide the model convergence.

The proposed model is also evaluated using the more challenging dataset: the KITTI testing dataset. In Table 4.3, this part only compares the proposed method

with state-of-the-art methods in three aspects: BEV, 3D, and 2D. Since it requires a great deal of data to compare these three performances, here, the results are simply compared based on the mean average precision (mAP). For the 3D performance, the proposed model has the best performance. For the BEV and 2D performances, the proposed method is the second-best, but the overall performance of the proposed method outperforms state-of-the-art methods when taking accuracy and speed into account. The results of the proposed method can be retrieved on the KITTI website based on the name of the proposed method, PFF3D. For the inference time, the proposed method still achieves a comparable speed by taking the high precision into account. The precision-recall curves of the best-proposed model are shown in Figure 4.6, Figure 4.7 and Figure 4.8.

Figure 4.9 presents some qualitative results. As can be seen in the figures, each object can be detected by the proposed model and the predicted bounding boxes are well-matched with their corresponding ground truth boxes. Even in very complex scenes, the proposed model can detect objects quite well, as shown in the last two rows of Figure 4.9.

4.4 Ablation Studies

This section analyzes the proposed methods individually by conducting ablation experiments using the KITTI validation dataset.

Effect of the Point Feature Fusion Module. This section analyzes the point feature fusion module based on the three classes in detail. In Table 4.4, the 'Addition' and 'Concatenation' represent the respective addition and concatenation fusion methods. The parameter'FC' means the fully connected layer followed after the fusion operation, as shown in Figure 4.2. The experimental results show that the combination of the addition operation and FC of the proposed module is best for the three classes: the car, pedestrian, and cyclist. The data in the first row give the results of the proposed method when only taking a point cloud as input. Compared with the LiDAR-based method (the first row), the proposed method (the fourth row) achieves 0.45%, 0.57%, 1.83%, and 0.6% gains in the 2D, AOS, BEV, and 3D performance, respectively. Compared with the performance improvement of cars, the proposed model is more helpful for improving the identification of pedestrians and cyclists.

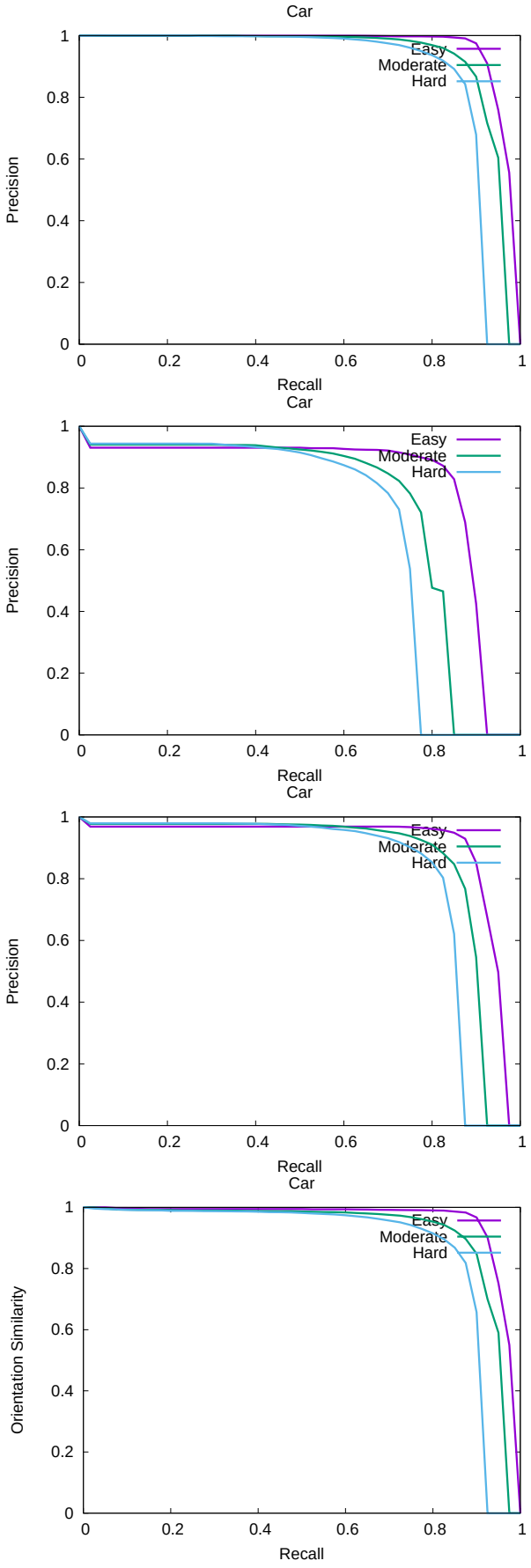


FIGURE 4.6: Visualization of the Precision-Recall curve for the car class. From up to down are the curves of 2D, 3D, BEV, and AOS. In each curve, each color line denotes one difficulty of the car class.

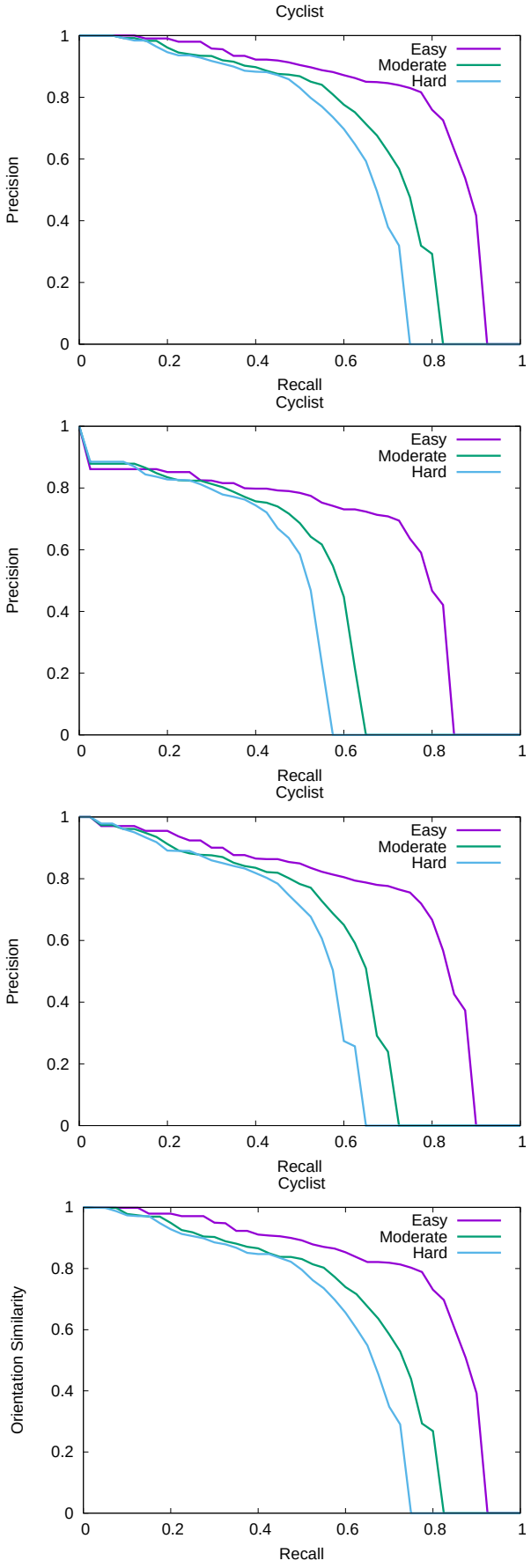


FIGURE 4.7: Visualization of the Precision-Recall curve for the cyclist class. From up to down are the curves of 2D, 3D, BEV, and AOS. In each curve, each color line denotes one difficulty of the cyclist class.

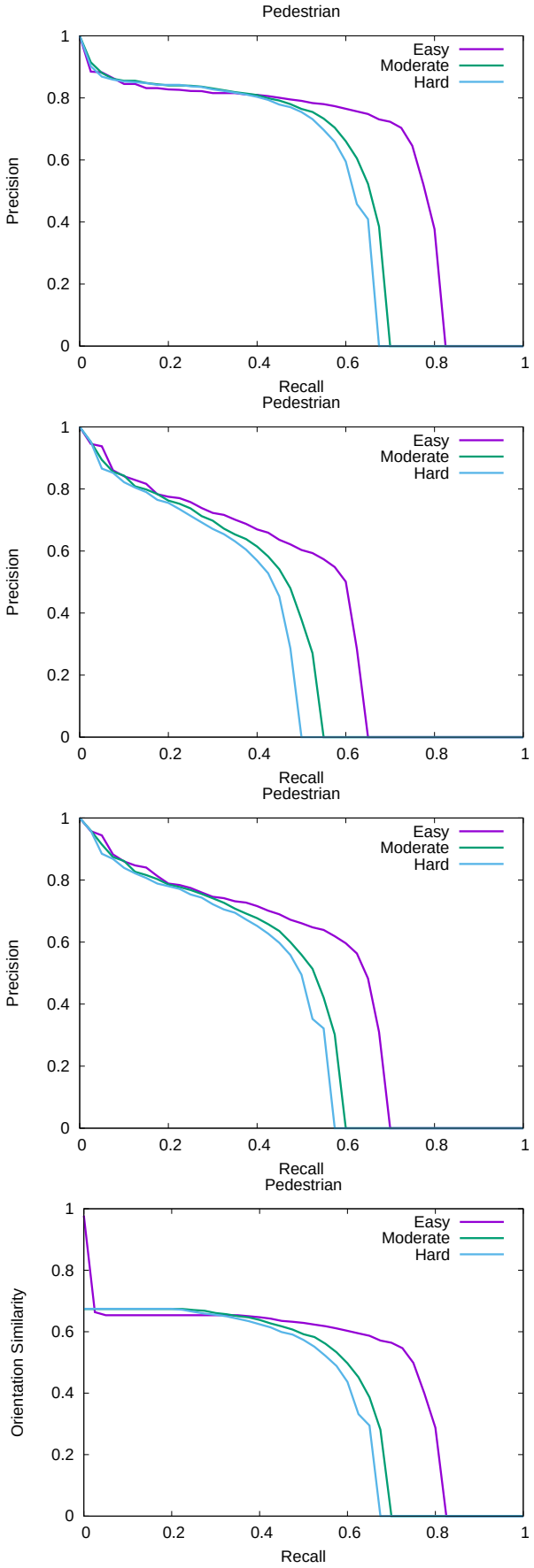


FIGURE 4.8: Visualization of the Precision-Recall curve for the pedestrian class. From up to down are the curves of 2D, 3D, BEV, and AOS. In each curve, each color line denotes one difficulty of the pedestrian class.

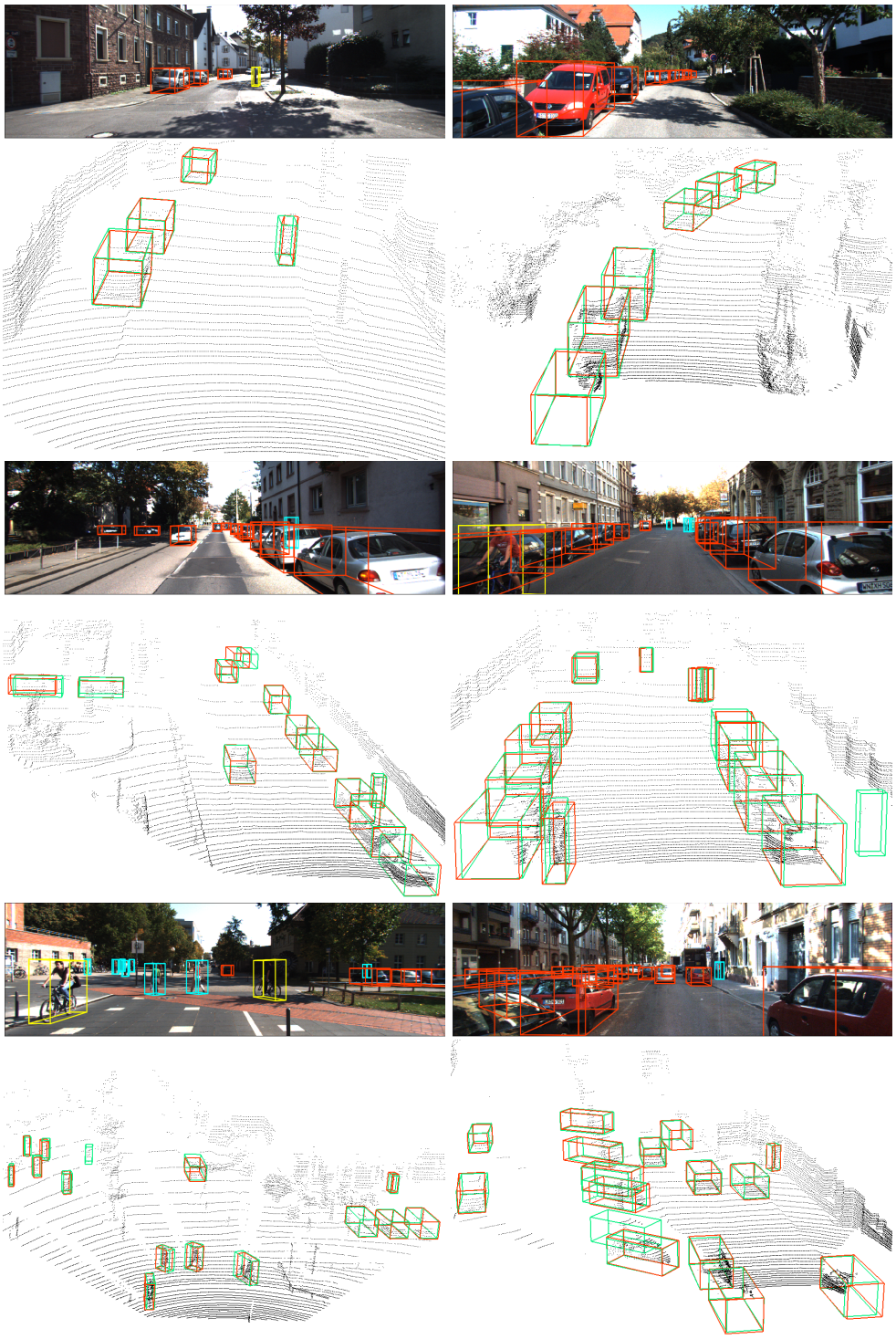


FIGURE 4.9: Qualitative results of the proposed method using the KITTI validation dataset. In the RGB images, the red, cyan, and yellow color represent the predictions for the car, pedestrian, cyclist, respectively. In the point cloud images, the green color denotes the ground truth, and the red color represents the prediction. The results in the point cloud images are used for a qualitative comparison.

TABLE 4.4: Effect of the point feature fusion module. The results are from the 'Moderate' difficulty category. The best result is highlighted in bold for each column.

Method			Cars (%)				Pedestrians (%)			
Addition	Concatenation	FC	2D	AOS	BEV	3D	2D	AOS	BEV	3D
			89.27	88.72	85.04	77.00	70.69	33.91	62.74	56.20
√			**89.74**	**89.39**	85.84	76.60	70.26	56.75	62.80	57.65
	√		89.54	85.84	**89.14**	77.01	69.99	56.59	62.71	57.74
√		√	89.72	89.29	86.97	**77.60**	**71.87**	**60.20**	64.22	**60.16**
	√	√	89.70	89.27	86.27	77.12	69.51	58.02	**65.89**	59.74

Effect of the Proposed Framework. The proposed 3D object detection framework is the first to directly project the raw RGB point features to a point cloud, as shown in Figure 2. The proposed approach is not without precedent but was discovered through experiments. Inspired by MVX-Net (Sindagi, Zhou, and Tuzel, 2019), we simply wanted to implement a lightweight design based on two backbones. One backbone was intended for 2D detection and the other one for 3D detection. First, ResNet-101 (He et al., 2016) was chosen as the backbone to extract features from RGB images. The results were as expected but the testing model ran very slowly. Then, ResNet-101 was replaced by ResNet-50 (He et al., 2016), and the model ran a little faster but the accuracy was almost the same. When using ResNetV1d-50 (He et al., 2016), the result was almost the same as the result for ResNet-50 (He et al., 2016). These results are thought-provoking. Hence, we boldly propose to map the raw point features of the RGB image to the point cloud without the 2D detection branch. The experimental results in Table 5 demonstrate that the proposed method is feasible. As can be seen in Table 4.5, the proposed approach not only drastically reduces the memory requirements for model operation, but also reduces the time of model training by half. It can be said that the proposed framework is lightweight, memory-saving, and energy-saving.

Effect of RGB$^+$. The RGB$^+$ construct includes two representations: the RGB$^{\mathbf{I}}$ and RGB$^{\mathbf{D}}$. In Table 4.6, there are three sets of experiments each for the car, pedestrian, and cyclist. The variable for each set of experiments is the input image. For the car class, the results of RGB$^{\mathbf{D}}$ outperform the results of RGB and RGB$^{\mathbf{I}}$ in all aspects. For the pedestrian and cyclist classes, the results of RGB$^{\mathbf{D}}$ surpasses both the results of RGB and RGB$^{\mathbf{I}}$ in some aspects. Hence, the RGB$^{\mathbf{D}}$ image is beneficial for improving 3D object detection.

TABLE 4.5: Effect of the proposed framework. The 'Time' column denotes the training time and the 'Memory' is the memory needed when the model is run for four batch sizes. The 'Rtime' column denotes the runtime. 'R' and 'V1' represent ResNet and ResNetV1d, respectively. '2D Image Branch' denotes that if the model use a full 2D image detection branch. The results of the cars are in the 'Moderate' difficulty category for the BEV and 3D.

2D Image Branch	Method	Time (hour)	Memory (MB)	Rtime (FPS)	BEV (%)	3D (%)
Yes	R101	28.0	19,500	9.5	85.95	76.82
	R50	23.5	12,550	11.0	86.29	76.92
	V1-50	25.0	12,700	10.6	85.51	76.48
	VGG11	16	11900	12.0	85.96	76.44
No	Ours	**11.5**	**4200**	**23**	86.17	**76.93**

TABLE 4.6: Effect of RGB^{+}. The results of the car class are in the 'Moderate' difficulty category.

Class	Input	2D (%)	AOS (%)	BEV (%)	3D (%)
Car	RGB	89.67	89.25	86.62	77.48
	RGB^{I}	89.51	88.90	86.27	76.92
	RGB^{D}	**89.72**	**89.29**	**86.97**	**77.60**
Ped.	RGB	70.38	46.09	**65.89**	58.99
	RGB^{I}	69.75	54.80	65.66	58.91
	RGB^{D}	**72.42**	**60.64**	64.22	**60.16**
Cyc.	RGB	62.19	58.33	57.83	55.53
	RGB^{I}	**67.52**	62.19	62.87	**61.37**
	RGB^{D}	67.21	**63.95**	**63.50**	60.07

4.5 Conclusion and Future Works

This paper is the first to propose a lightweight, memory-saving, and energy-saving framework for 3D object detection based on LiDAR and an RGB camera. Different from the existing frameworks, the proposed framework only employs one backbone to extract features from a point cloud and RGB image. The framework benefits from the proposed module, i.e., the point feature fusion module. The fusion module directly extracts the point features of RGB images and fuses them with the corresponding point cloud features. The experimental results using both the KITTI validation dataset and testing dataset demonstrate that the proposed method significantly improves the speed (23 FPS) of LiDAR-camera-based 3D object detection compared with other state-of-the-art approaches. Note that the proposed native model can achieve an inferring speed 23 FPS.

In the future, the proposed method will be directly used in the point-based methods (Yang et al., 2020), thereby achieving breakthroughs in both accuracy and speed.

Chapter 5

Conclusion

5.1 Conclusions

The works in this thesis focus on fast and accurate 3-D object detection, which takes raw point cloud and RGB image as inputs. A complete pipeline is presented alongside the discussion related to each component.

In Chapter 3, this thesis proposes a one-stage 3D object detection framework based on LiDAR and Camera, which benefits from the three attention mechanisms to boost 3D detection accuracy. First, the height attention mechanism is introduced into the input RGB images to generate the RGB^D images. Second, the global feature attention mechanism is utilized for both the RGB image and the BEV branches at the feature extraction stage. It benefits by capturing the discriminative features from both the channels and spatial dimensions. Finally, the region-of-interest attention mechanism is employed to fuse the paired view-specific ROIs. Our proposed method greatly improves the 3D object detection performance, and it outperforms all state-of-the-art methods based on LiDAR and Camera in 3D object detection. In the future, the way to generate BEV images can be replaced with a learnable feature generator like SECOND (Yan, Mao, and Li, 2018). Additionally, the method of anchor generation can be changed from anchor-based to anchor-free. In this way, with the advantages of RGB images and LiDAR point clouds, 3D object detection based on LiDAR and camera can achieve better performance than LiDAR-based methods.

In Chapter 4, this paper is the first to propose a lightweight, memory-saving, and energy-saving framework for 3D object detection based on LiDAR and an RGB

camera. Different from the existing frameworks, the proposed framework only employs one backbone to extract features from a point cloud and RGB image. The framework benefits from the proposed module, i.e., the point feature fusion module. The fusion module directly extracts the point features of RGB images and fuses them with the corresponding point cloud features. The experimental results using both the KITTI validation dataset and testing dataset demonstrate that the proposed method significantly improves the speed (23 FPS) of LiDAR-camera-based 3D object detection compared with other state-of-the-art approaches. Note that the proposed native model can achieve an inferring speed 23 FPS. In the future, the proposed method will be directly used in the point-based methods (Yang et al., 2020), thereby achieving breakthroughs in both accuracy and speed.

5.2 Future Works

In the future, on the one hand, we will study more efficient 3D object detection based on LiDAR. On the other hand, we explore more fast and efficient methods for multi-sensor 3-D object detection, thereby achieving breakthroughs in both accuracy and speed.

Appendix A

Publications

A.1 Journal

1. Li-Hua Wen and Kang-Hyun Jo, Three-Attention Mechanisms for One-Stage 3D Object Detection Based on LiDAR and Camera, IEEE Transactions on Industrial Informatics, Vol.17, No.10, pp.1-1, 2021.

2. Li-Hua Wen and Kang-Hyun Jo, Fast and Accurate 3D Object Detection for Lidar-Camera-Based Autonomous Vehicles Using One Shared Voxel-Based Backbone, in IEEE Access, vol. 9, pp. 22080-22089, 2021.

3. Li-Hua Wen and Kang-Hyun Jo, Deep Learning-Based Vision System for Autonomous Driving: A Comprehensive Survey, in Neurocomputing, 2021. (Minor Revision)

A.2 Conference

1. Li-Hua Wen and Kang-Hyun Jo, Vehicle localization and navigation on region with disappeared lane line marking, SII, Japan, 2016.

2. Li-Hua Wen and Kang-Hyun Jo, Efficient and robust drivable region extraction for autonomous vehicles, ICCAS, Korea, 2017.

3. Li-Hua Wen and Kang-Hyun Jo, Traffic sign recognition and classification with modified residual networks, SII, Taiwan (China), 2017.

4. Li-Hua Wen and Kang-Hyun Jo, Fully Convolutional Neural Networks for 3D Vehicle Detection Based on Point Clouds, ICIC, Nanchang (China), 2019.

5. Li-Hua Wen and Kang-Hyun Jo, LiDAR-Camera-Based Deep Dense Fusion for Robust 3D Object Detection, ICIC, Italy, 2020.

6. Li-Hua Wen, Xuan-Thuy Vo and Kang-Hyun Jo, 3D SaccadeNet: A Single-Shot 3D Object Detector for LiDAR Point Clouds, ICCAS, Korea, 2020.

7. Xuan-Thuy Vo, Li-Hua Wen and Kang-Hyun Jo, Bidirectional Non-local Networks for Object Detection, ICCCI, Vietnam, 2020.

8. Li-Hua Wen and Kang-Hyun Jo, Analysis of Various Point Clouds Feature Encoders for Two-Sensor-based 3D Object Detection, IWIS, Korea, 2020.

9. Li-Hua Wen and Kang-Hyun Jo, A Lightweight One-Stage 3D Object Detector Based on LiDAR and Camera Sensors, ISIE, Japan, 2021.

10. Li-Hua Wen and Kang-Hyun Jo, A Lightweight Attention Fusion Module for Multiple-Sensor 3-D Object Detection, ICIC, Shenzhen (China), 2021.

Bibliography

Charles, R. Q. et al. (2017). "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85.

Chen, X. et al. (2016). "Monocular 3D Object Detection for Autonomous Driving". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2147–2156.

Chen, X. et al. (2017). "Multi-view 3D Object Detection Network for Autonomous Driving". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6526–6534.

Fu, J. et al. (2019). "Dual Attention Network for Scene Segmentation". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3141–3149.

Fu, J. et al. (2020). "Scene Segmentation With Dual Relation-Aware Attention Network". In: *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14. DOI: 10.1109/TNNLS.2020.3006524.

Geiger, A., P. Lenz, and R. Urtasun (2012). "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361. DOI: 10.1109/CVPR.2012.6248074.

Girshick, R. (2015). "Fast R-CNN". In: *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448.

Graham, Benjamin and Laurens van der Maaten (2017). "Submanifold Sparse Convolutional Networks". In: *CoRR* abs/1706.01307. arXiv: 1706.01307. URL: http://arxiv.org/abs/1706.01307.

Han, X. et al. (2019). "Active Object Detection With Multistep Action Prediction Using Deep Q-Network". In: *IEEE Transactions on Industrial Informatics* 15.6, pp. 3723–3731. ISSN: 1941-0050. DOI: 10.1109/TII.2019.2890849.

He, C. et al. (2020). "Structure Aware Single-Stage 3D Object Detection From Point Cloud". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11870–11879. DOI: 10.1109/CVPR42600.2020.01189.

He, K. et al. (2016). "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

He, Kaiming et al. (2016). "Identity Mappings in Deep Residual Networks". In: *Computer Vision – ECCV 2016*. Cham: Springer International Publishing, pp. 630–645.

He, Tong and Stefano Soatto (July 2019). "Mono3D++: Monocular 3D Vehicle Detection with Two-Scale 3D Hypotheses and Task Priors". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33, pp. 8409–8416.

Hoang, V., D. Huang, and K. Jo (2020). "3D Facial Landmarks Detection for Intelligent Video Systems". In: *IEEE Transactions on Industrial Informatics*, pp. 1–1.

Hu, J., L. Shen, and G. Sun (2018). "Squeeze-and-Excitation Networks". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7132–7141. DOI: 10.1109/CVPR.2018.00745.

Jaderberg, Max et al. (2015). "Spatial Transformer Networks". In: *Advances in Neural Information Processing Systems*. Vol. 28, pp. 2017–2025.

Ku, J. et al. (2018). "Joint 3D Proposal Generation and Object Detection from View Aggregation". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1–8.

Lang, A. H. et al. (2019). "PointPillars: Fast Encoders for Object Detection From Point Clouds". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12689–12697.

Li, Bo, Tianlei Zhang, and Tian Xia (2016). "Vehicle Detection from 3D Lidar Using Fully Convolutional Network". In: *Proceeding of Robotics: Science and Systems*.

Li, Minle et al. (2019). "One-Stage Multi-Sensor Data Fusion Convolutional Neural Network for 3D Object Detection". In: *Sensors* 19.6, p. 1434. ISSN: 1424-8220. DOI: 10.3390/s19061434.

Li, P., X. Chen, and S. Shen (2019). "Stereo R-CNN Based 3D Object Detection for Autonomous Driving". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7636–7644.

Liang, M. et al. (2019). "Multi-Task Multi-Sensor Fusion for 3D Object Detection". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7337–7345.

Liang, Ming et al. (2018). "Deep Continuous Fusion for Multi-sensor 3D Object Detection". In: *Computer Vision – ECCV 2018*, pp. 663–678.

Lin, T. et al. (2017). "Feature Pyramid Networks for Object Detection". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944.

Lin, T. et al. (2020). "Focal Loss for Dense Object Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.2, pp. 318–327.

Liu, Wei et al. (2016). "SSD: Single Shot MultiBox Detector". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, pp. 21–37. ISBN: 978-3-319-46448-0.

Lv, N. et al. (2018). "Deep Learning and Superpixel Feature Extraction Based on Contractive Autoencoder for Change Detection in SAR Images". In: *IEEE Transactions on Industrial Informatics* 14.12, pp. 5530–5538.

Mozifian, Melissa Farinaz (2018). "Real-time 3D Object Detection for Autonomous Driving". Master Thesis. University of Waterloo. URL: http://hdl.handle.net/10012/13267.

Qi, C. R. et al. (2018). "Frustum PointNets for 3D Object Detection from RGB-D Data". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 918–927.

Qi, Charles Ruizhongtai et al. (2017). "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: *Advances in Neural Information Processing Systems 30*, pp. 5099–5108.

Redmon, Joseph and Ali Farhadi (2017). "YOLO9000: Better, Faster, Stronger". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525. DOI: 10.1109/CVPR.2017.690.

Redmon, Joseph et al. (2016). "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788. DOI: 10.1109/CVPR.2016.91.

Ren, S. et al. (2017). "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6, pp. 1137–1149.

Shi, S. et al. (2020). "PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10526–10535. DOI: 10.1109/CVPR42600.2020.01054.

Shi, W. and R. Rajkumar (2020). "Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1708–1716. DOI: 10.1109/CVPR42600.2020.00178.

Simonyan, Karen and Andrew Zisserman (2015). "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations*.

Sindagi, V. A., Y. Zhou, and O. Tuzel (2019). "MVX-Net: Multimodal VoxelNet for 3D Object Detection". In: *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7276–7282.

Szegedy, Christian et al. (2015). "Going deeper with convolutions". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594.

Tian, Zhi et al. (2019). "FCOS: Fully Convolutional One-Stage Object Detection". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9626–9635. DOI: 10.1109/ICCV.2019.00972.

Uijlings, Jasper RR et al. (2013). "Selective Ssearch for Object Recognition". In: *International Journal of Computer Vision* 104.2, pp. 154–171.

Wang, F. et al. (2017). "Residual Attention Network for Image Classification". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6450–6458.

Wang, J. et al. (2019). "MCF3D: Multi-Stage Complementary Fusion for Multi-Sensor 3D Object Detection". In: *IEEE Access* 7, pp. 90801–90814.

Wang, J. et al. (2020). "KDA3D: Key-Point Densification and Multi-Attention Guidance for 3D Object Detection". In: *Remote Sensing* 12.11, p. 1895.

Wang, X. et al. (2018). "Non-local Neural Networks". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7794–7803. DOI: 10.1109/CVPR.2018.00813.

Wang, Yue et al. (2018). *Dynamic Graph CNN for Learning on Point Clouds*. arXiv: 1801.07829 [cs.CV].

Wen, Li-Hua and Kang-Hyun Jo (2021). "Three-Attention Mechanisms for One-Stage 3D Object Detection Based on LiDAR and Camera". In: *IEEE Transactions on Industrial Informatics*, pp. 1–1. DOI: 10.1109/TII.2020.3048719.

Wen, Lihua and Kang-Hyun Jo (2019). "Fully Convolutional Neural Networks for 3D Vehicle Detection Based on Point Clouds". In: *Intelligent Computing Theories and Application*. Cham: Springer International Publishing, pp. 592–601.

Wen, Lihua and Jo Kang-Hyun (2020). "LiDAR-Camera-Based Deep Dense Fusion for Robust 3D Object Detection". In: *Intelligent Computing Methodologies*. Cham: Springer International Publishing, pp. 133–144.

Wen, Lihua, X. T. Vo, and Kang-Hyun Jo (2020). "3D SaccadeNet: A Single-Shot 3D Object Detector for LiDAR Point Clouds". In: *2020 20th International Conference on Control, Automation and Systems (ICCAS)*, pp. 1225–1230. DOI: 10.23919/ICCAS50221.2020.9268367.

Woo, Sanghyun et al. (2018). "CBAM: Convolutional Block Attention Module". In: *Computer Vision – ECCV 2018*, pp. 3–19.

Xu, B. and Z. Chen (2018). "Multi-level Fusion Based 3D Object Detection from Monocular Images". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2345–2353.

Xu, D., D. Anguelov, and A. Jain (2018). "PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 244–253.

Yan, Yan, Yuxing Mao, and Bo Li (2018). "SECOND: Sparsely Embedded Convolutional Detection". In: *Sensors* 18, 3337.

Yang, Bin, Wenjie Luo, and Raquel Urtasun (2018). "PIXOR: Real-time 3D Object Detection from Point Clouds". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7652–7660. DOI: 10.1109/CVPR.2018.00798.

Yang, Z. et al. (2020). "3DSSD: Point-Based 3D Single Stage Object Detector". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11037–11045. DOI: 10.1109/CVPR42600.2020.01105.

Yoo, Jin Hyeok et al. (2020). "3D-CVF: Generating Joint Camera and LiDAR Features Using Cross-view Spatial Feature Fusion for 3D Object Detection". In: *Computer Vision – ECCV 2020 - 16th European Conference, 2020, Proceedings*, pp. 720–736. DOI: 10.1007/978-3-030-58583-9_43.

Zhou, Y. and O. Tuzel (2018). "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4490–4499.

Zhou, Yin et al. (2020). "End-to-End Multi-View Fusion for 3D Object Detection in LiDAR Point Clouds". In: *Proceedings of the Conference on Robot Learning*. Vol. 100. Proceedings of Machine Learning Research, pp. 923–932.