Multi-level Fusion Network for 3D Object Detection from Camera and LiDAR Data

by

Zixuan Zhao

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Engineering
(Electrical Engineering)
in the University of Michigan-Dearborn
2020

Master's Thesis Committee:

Professor Yi Lu Murphey, Chair
Associate Professor Paul Watta
Assistant Professor Lu Wei

ACKNOWLEDGMENTS

It is imperative that I thank Professor Yi Lu Murphey, for her supervision and being so supportive throughout this thesis work. Special thanks to Zihao Zhao, who enlightened me to computer vision, for his knowledge, patience and so much more than that. I thank my parents, for supporting me both financially and psychologically, and for providing me insight and the whole fascinating world. Huge thanks to Tenghui Shao, Tian Qin and Wenjue Li, my mates, for everything and everyday yet to come and for letting me feel not alone through this journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

3DOD	3D Object Detection
2DOD	2D Object Detection
GNN	Graph Neural Networks
OD	Object Detection
MOT	Multiple Object Tracking
CNN	Convolutional Neural Networks
RPN	Region Proposal Network
IOU	Intersection Over Union

ABSTRACT

In 3D Object Detection (3DOD), the fusion of point cloud data and image data is of vital importance, because this can maximize the use of the high resolution information of RGB image and the rich 3D information of point cloud data. This thesis proposes a two-stage 3D object detection system, which takes input from the camera and LiDAR data, and outputs the localization and category of the 3D bounding box. The system uses a novel feature extractor to learn the full-resolution scale features while keeping the computation speed coupled with a multimodal fusion Region Proposal Network (RPN) architecture. The second stage detection network regresses the offsets between the 3D proposals generated by the RPN and the ground truth boxes using a 6-dimension encoding technique. Experiments conducted on the Kitti dataset showed the performance boost of the proposed algorithm over the state-of-the-arts on the 3D Object Detection tasks.

CHAPTER 1

Introduction

1.1 Overview

Autonomous driving is a subject undergoing intense research work towards achieving fully automated system. An accurate perception system is an essential condition which has considerable impact on the performance of an automated driving system that generally consists of 3 major modules: Object Detection (OD), Multiple Object Tracking (MOT), as well as action recognition. For instance, in order to make the correct decisions for path planning, the self-driving vehicle should be aware of the pedestrians and other surrounding objects, as well as their motions and trajectories. Evidence suggests that sensor technologies is an important factor for self-driving perception system, which could potentially make a great impact on traffic efficiency [6] [10].

Recent research works have contributed a significant number of breakthroughs in deep learning perception algorithms and exteroceptive sensor technologies, from which we are witnessing many accurate detection algorithms on 2D sensors such as monocular camera systems [11] [12] and fisheye camera systems [13]. However, as we are in a world composed of 3D geometry, it's essential to let the autonomous driving system to have an understanding of the 3D surrounding environment, which depends highly on depth information which's playing a vital role in speed computing, collision avoidance and etc. To this end, investigating 3D vision algorithm has yet been a fundamental research problem of great interest in a wide range of fields [14], which is the focus of this research.

3D vision requires 3D data, which has many different representations shown in Figure 1.1. Among them, 2D images taken from different angles that convey the 3D data are multi-view images representation . Volumetric representation measures the 3D object into a small 3D grid which is typical in 3D simulation and reconstruction (small 3D cubes). Point clouds are a set of points recording the 3D coordinates of many points on target's surfaces in three-dimensional space, typically created by some expensive sensors such as depth cameras, LIDARs, and other 3D scanners. Mesh, or representation of Polygon Mesh, is to represent 3D object with triangles or quads, and is

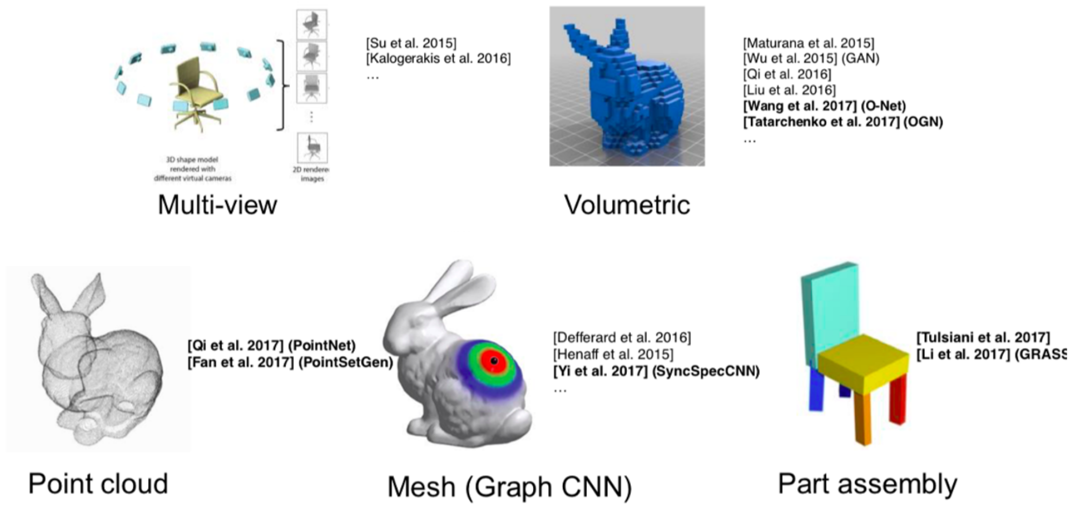


Figure 1.1: 3D data has many representations. Image credit: Su et al, "A Tutorial on 3D Deep Learning", in conjugation with CVPR 2017.

mostly used in 3D rendering and GPU modeling as GPUs are friendly with triangle geometry computing. Moreover, recent research works have applied Graph Neural Networks (GNN) on Mesh data and it is still a theme undergoing a significant number of research interests. Specifically, data in point cloud format is the most informative among these many representations, as it can provide high precision 3D geometry invariant to complex or poor lighting environments. On the other hand, point cloud is also the closest to the raw 3D sensor data and simplest in representation [15].

Since 2017, deep learning algorithms on raw point clouds data have made several significant progresses that overcame several technical challenges of LIDARs. For instance, in terms of data structure, point cloud is sparse, orderless and irregular [9]. In recent state of the art works, Qi et al in [2] proposed Pointnet to deal with the orderlessness and transformation variance of point cloud by introducing the Pointnet backbone and transform network, and this is still a remarkable work directly applying deep learning on raw point cloud data. There are also other works done to transfer point cloud data into other representations, such as Volumetric methods based [16] [17], Tree-like representation based [18] [19] [20] [21], and Geometric representation based [22] [23] [24] . These results marked the unreplaceable milestones in 3D deep learning.

However, pure point cloud data in the same time is constrained by the low-resolution and severe occlusion problems. This is because the resolution of the point cloud data captured by LiDARs in an autonomous driving application varies from distance, as the LiDAR send out certain amount of point towards each angle, making it unfriendly dealing with objects away from the sensor. Another standard in choosing sensors in a self-driving application is the cost issue, while monocular camera system is less expensive but high resolution and color features, and 3D sen-

Table 1.1: Comparison between image and point cloud data [9]

	IMAGE	POINT CLOUD
Data Structure	Regular	Irregular
Permutation	Ordered	Oderless
Data Type	Discrete	Continuous
Coordinates	Projective	Euclidian
Dimension	2D	3D
Resolution	High	Low

sors cost relatively higher, such as laser and LiDAR. To this end, the fusion of multiple sensors, combining the technical characteristics and the cost advantages of each, has become an emerging research theme [23]. Specifically, the sensor fusion of low-cost monocular camera image data and rich-feature point cloud data by LiDAR, is among the most popular and the most informative choice of combination.

However, it is not trivial to fuse the image data and point cloud data, due to their many differences referring to data structures and etc. (see Table 1.1). First of all, image data is ordered data, whereas point cloud is orderless data that it's hard to directly fuse the two without hand-crafted features. Secondly, image data is the projection of object in the image plane, whereas the point cloud is the record of real-world preserving 3D geometry [25]. Moreover, image data, due to its data format, can be applied by the CNN, etc. directly while it is hard to implement such technologies on point cloud data. Recent breakthroughs have been contributed in several attempts of fusion on result level, feature level and multi-level. However, there're still bridges need to be built between the camera and LiDAR data to reach a certain level of accuracy, and the fusion of the two is still an unsolved research theme need to be improved.

3DOD aims to locate and recognize the tight bounding boxes of objects in 3D space. Recent research works have witnessed a significant numbers of breakthroughs in 3DOD fusing 2D camera image data and 3D LiDAR data, which will be discussed in Chapter 2. Similar to 2D Object Detection (2DOD), there are two main approaches for 3DOD: two-stage models and one-stage model. This thesis will focus on the two-stage model that processes the 3D RPN and second stage detection network in a sequential order. Inspired by AVOD [5] among these state-of-the-arts, this thesis proposed a two-stage 3DOD system that fuses the image and LiDAR data in feature level, using a 6-dimensional encoding for 3D bounding box proposal regression, and reduced the weighted parameters in first stage RPN with a novel feature descriptor. The proposed model achieves 82.68% average precision based on 3D Intersection Over Union (IOU) 70% requirement in Kitti [7] benchmark.

1.2 Contributions and Thesis Outline

A new framework for 3DOD is proposed in this thesis, which is inspired by AVOD[5], but improved the model in many aspects. In summary, the key contributions of the proposed architecture are delivered as follows:

- This thesis proposes a novel feature extractor that can produce high resolution feature maps resulting in generating accurate proposals in the first stage region proposal network. The feature map is only of 16 dimension depth compared to 256 dimension of AVOD, which can solve the issue of memory overhead of the system computation.
- Though attempts have been made by many works [5] to obtain the most proper method for the 3D bounding box encoding for the regression, there are still redundant elements conformed by the cuboid geometry constraints. We propose an efficient 3D bounding box encoding for the second detection stage to regress only 6-dimensional offsets between the proposals and the ground truth boxes, allowing for a higher localization accuracy.
- The overall system is able to be processed at a higher speed while the precision can outperform many state of the arts by a large margin.

Contents for each chapter of the thesis are as follows:

In Chapter 2 we introduce the background and the literature survey on 3DOD. In Section 2.1, we present the foundations and the background of the LiDAR sensors and point cloud representations. In Section 2.2, we introduce the brief history for computer vision with deep learning. Specifically, we introduce the deep learning applications in the literature on point cloud data and the challenges of applying deep learning techniques. In Section 2.3, the common techniques in the literature are introduced focusing on 3DOD tasks fusing the camera and the LiDAR data, where the backbone architecture of this thesis AVOD [5] is discussed, along with other 2D and 3D proposal based methods.

Chapter 3 presents the methodology of the proposed framework. In Section 3.1, we described the scenario for the problem setting and the problem definition: the input, output and the data flow. We also present the overall system framework. In Section 3.2 to 3.5, the details of each module of the architecture is described. Specifically, we discuss the feature extractor we proposed and the 3D bounding box encoding techniques we introduced. The preprocessing of the inputs and the point cloud processing, the feature descriptor, the two stage detection modules as well as the fusion method are discussed in details. In Section 3.6. the training details and data augmentation techniques we used are introduced.

In Chapter 4, we introduced Kitti [7], the dataset used in this thesis, as well as the experiment settings, followed by the deep analysis of the system architecture by conducting the ablation study.

Next, to evaluate the performance of our system, experiments are conducted on the Kitti validation set in comparison with a series of other state-of-the-arts. The results show that the proposed framework outperforms its backbone AVOD [5] and other state of the arts..

Finally, in Chapter 5, we summarize the endeavor this thesis work has made, provide the insights on potential directions for the next step of research.

CHAPTER 2

Related Works

In this Chapter, we introduce the theoretical backgrounds that are of specific interest in the thesis and present a review of related research in two-stage 3DOD in the literature. The reader is thought to be comfortable with essential deep learning fundamentals. For a complete introduction of deep learning essentials, see for example [26].

This thesis focuses on the fusion of image and the LiDAR data, where we have already been familiarized with the image data based detection techniques. In Section 2.2, the point cloud data produced by LiDAR sensors is presented as well as its application in deep learning is discussed. One crucial backbone model proposed by Qi et al. [2] is discussed.

In Section 2.3, we present related backgrounds on the frameworks of the 3DOD model fusing camera and LiDAR data. And the training methods, i.e. loss functions, are covered in Section 2.3.3.

2.1 LiDAR and Point Clouds

LiDAR, also known as LADAR or Laser Radar, (light detection and ranging). LiDAR is self-evident as one of the most significant and necessary sensors in achieving fully autonomous driving and has a range of applications such as obstacle detection, road edge and lane detection, map creation, etc. LiDAR has a large number of characteristics. Firstly, compared to the camera sensor, it has wide range, high accuracy, rapid frequency and good environmental adaptability. Compared with stereo vision, it can directly return the distance from the measured object to the LiDAR, and the distance measurement is more direct and accurate. It can identify various kinds of obstacles in the surrounding environment, such as those that can be crossed (vegetation), those that cannot be crossed (rocks, trees, etc.), and potential negative obstacles (cliffs, mountain streams, etc.). As an active sensing device for perception system, LiDAR uses an excitation source to periodically drive a laser to emit laser pulses, shown as Figure 2.1. The beam controller controls the direction and number of laser lines. The detecter is responsible for receiving the echo of the laser beam reflected

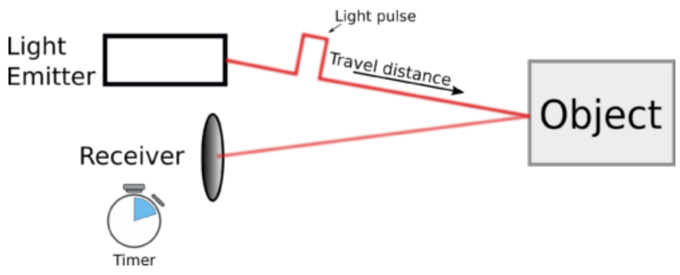


Figure 2.1: The working diagram of LiDAR sensors. Image credit: Luminar Technologies via The New York Times

by the target and generate the received signal. LiDAR uses a timer (stable quartz clock) to measure the difference between the transmission time and the reception time, and calculate and output the measured distance, angle, transmission intensity and other information through the information processing module. The typical data observed by LiDAR is an $N \times 4$ matrix $[x, y, z, w]$, where w is the reflection intensity, that is, the Point Cloud.

As a commonly used data expression, point cloud retains the original geometric information in the three-dimensional space, including three-dimensional coordinates, color, classification value, intensity value, time without any discretization. Therefore, it is the preferred representation for many applications related to perception systems, such as autonomous driving and robotics. A visualization of point cloud from Kitti dataset is shown in Figure 2.2. In recent years, deep learning technology has become a research hotspot in the fields of computer vision, speech recognition, natural language processing (NLP), and bioinformatics [27]. However, deep learning of 3D point clouds still are facing many challenges such as small data set, high dimensionality, and unstructured.

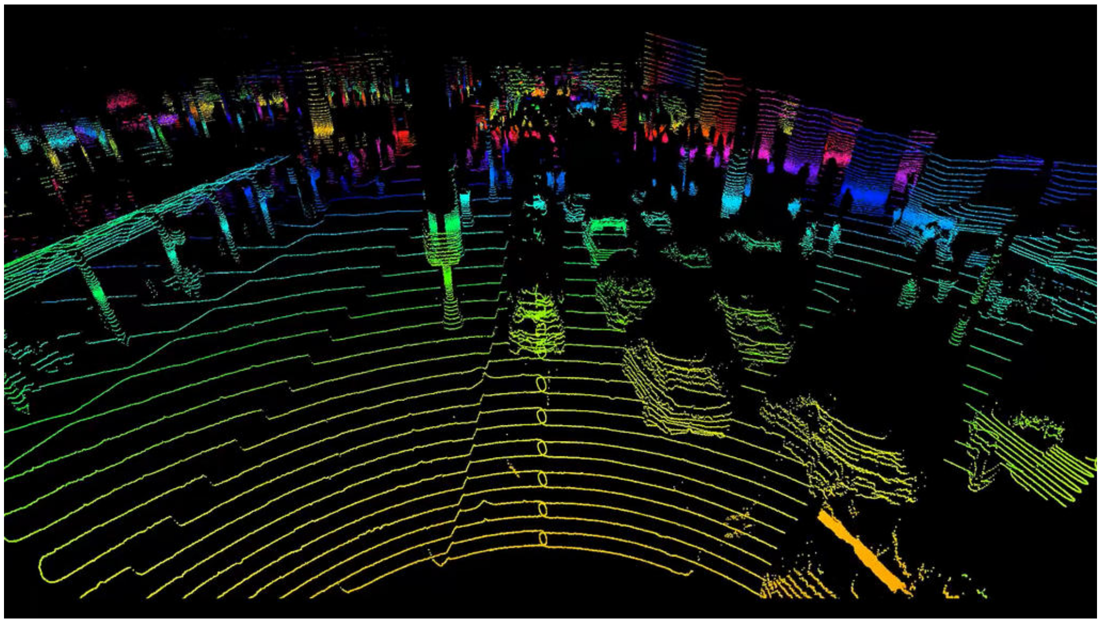


Figure 2.2: The visualization of point cloud data.

2.2 Deep Learning

2.2.1 Introduction

Deep learning is a research methodology served for certain data-driven tasks that has developed rapidly in recent years, and it has achieved great success in many sub-fields of artificial intelligence. From the root, deep learning is a branch of machine learning, which refers to a class of problems and methods to solve these problems.

2.2.2 Deep Learning and Computer Vision

In the field of computer vision, especially in image classification task, deep learning really began to be hot, and convolutional neural networks (CNN) began to become a household name, since Alex and his advisor Hinton's winning the first place (using deep neural networks) [28] in the 2012 ImageNet large-scale image recognition competition [1] (ILSVRC2012), crushing the second (using traditional computer vision methods) by 10 percentage points. As illustrated in Figure 2.3, we see that from AlexNet [28] in 2012 (83.6%) to the 2013 ImageNet large-scale image recognition competition champion (88.8%), to VGG [29] in 2014 (92.7%) and GoogLeNet [30] (93.3%) in the same year, and finally in 2015, when Microsoft proposed ResNet [31] which achieved a top 5 accuracy rate of 96.43%, exceeding the level of humans (human accuracy is only 94.9%), and deep learning really beats the human beings on the visual classification tasks.

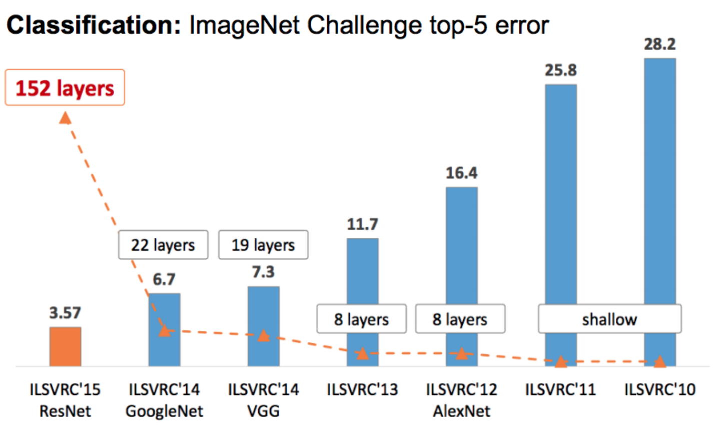


Figure 2.3: 2010-2015 ILSVRC competition image recognition [1] error rate trend.

It is worth mentioning, that these early-stage outperforming deep learning models and their thoughts are inspiring so many other disciplines in computer vision such as object detection and instance segmentation to represent the feature of an object. Table 2.1 presents the most recent state-of-the-arts for multi-discipline computer vision tasks published by peer reviewed conferences or articles that use the above mentioned models in image classification task as backbones of their models. Each task is defined as follows:

- Object Detection (OD): Given an image or a sequence of video, the system need to identify and locate objects.
- Instance Segmentation: The system identifies each instance of each object within the image at the pixel level.
- Re-identification (ReID): Person re-identification (Re-ID) is a task that aims to retrieve a person of interest across multiple cameras.
- Semantic Segmentation: The goal of semantic segmentation is to label each pixel of an image with a corresponding class of what is being represented.

from where we can observe that, although the tasks for computer vision are of many kinds and the breakthroughs leveraging deep learning are of rapid speed, the core ideas and backbones are still inspired deeply by the previous deep networks we mentioned above.

2.2.3 Challenges for Deep Learning on Point Clouds

Deep Learning methodologies on RGB images have been vastly exploited over the past few years with excellent performances for 2D vision tasks. Recently, 3D vision has aroused many re-

Table 2.1: State-of-the-arts of computer vision tasks with deep learning based backbones

TASK	MODEL	BACKBONE	CONFERENCE
Object Detection	DeformV2 [32]	ResNet [31]	CVPR 2019
	FasterRCNN [12]	GoogLeNet [30]	NIPS 2015
Instance Segmentation	MaskRCNN [33]	ResNet [31]	CVPR 2017
	MNC [34]	ResNet [31]	CVPR 2016
Semantic Segmentation	DPN [35]	VGG [29]	ICCV 2015
	CRF-RNN [36]	VGG [29]	ICCV 2015
Re-identification (ReID)	OSNet [37]	ResNet [31]	ICCV 2019
	AACN [38]	GoogLeNet [30]	CVPR 2019
	PSE [39]	ResNet [31]	CVPR 2018
	HA-CNN [40]	GoogLeNet [30]	CVPR 2018

searchers' and companies' interest, and the 3D data is growing in a rapid pace where we can see a significant amount of 3D dataset going public, such as Kitti [7], Lyft [41] and Waymo [42]. However, due to the properties of deep learning and Convolutional Neural Networks (CNN) architecture and the limitations of point clouds data structure, it is not simple to apply 2D based deep learning techniques directly on 3D point cloud data. And the main challenges are discussed as follows:

- Unstructured Data: Point clouds are a kind of unstructured data, as they are arbitrary data points of (X, Y, Z) coordinates in 3D space. And there is no structured grid for the CNN filters to slide.

- Invariance to permutations: One of the major difficulties that needs to be solved is the permutation invariance of point clouds. As the point clouds are a set of unordered point data representing the 3D geometry of an object invariant to the permutation of the matrix representation of the point cloud. However, as shown in Figure 2.4, different matrixes represent completely different inputs mathematically. This task can be formulated as:

$$f(x_1, x_2, ..., x_n) \equiv f(x_{\pi_1}, x_{\pi_2}, ..., x_{\pi_n}), x_i \in \mathbb{R}^D \tag{2.1}$$

- Uncertainty of the number of data. In 2D vision tasks, image data is represented through pixels, and the number of pixels input to the model can always be a fixed value, whereas point cloud data can conclude the umber of data points varying greatly. This also causes the difficulty of deep learning applying on point cloud.

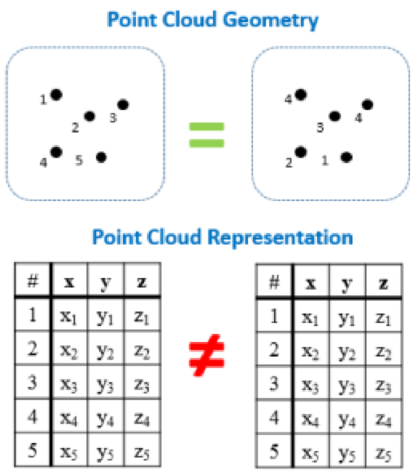


Figure 2.4: Permutation invariance problem.

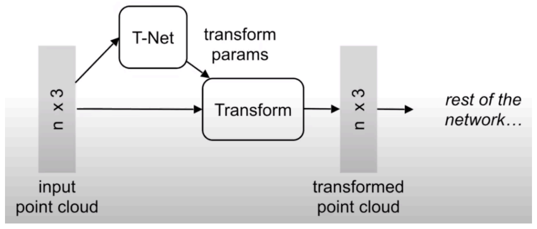


Figure 2.5: PointNet vanilla structure. [2].

2.2.4 PointNet and PointNet ++

As a mega for the 3D point cloud deep learning, PointNet and PointNet ++ [2] proposed a solution to solve challenges in deep learning on point clouds. PointNet is proposed as an end-to-end 3D point cloud classification baseline model, which is the first work that take the raw point clouds as the input and outputs the category scores. The PointNet vanilla architecture is shown in Figure 2.5, which is designed to tackle with permutation invariance by introducing a simple symmetric function and the transformation invariance by leveraging the spatial transform network with matrix multiplication. The PointNet network architecture is illustrated in Figure 2.6, whose input is the raw point cloud and the output are the classification scores. The work can also be extended to object detection and segmentation tasks as well, and the authors provided an direction on doing such tasks.

There are drawbacks of PointNet, however: PointNet learns the global feature of an object, which results only in one point or all points being learned; on the other hand, the network can fail in segmentation tasks due to the absence of the local context. PointNet++ was suggested

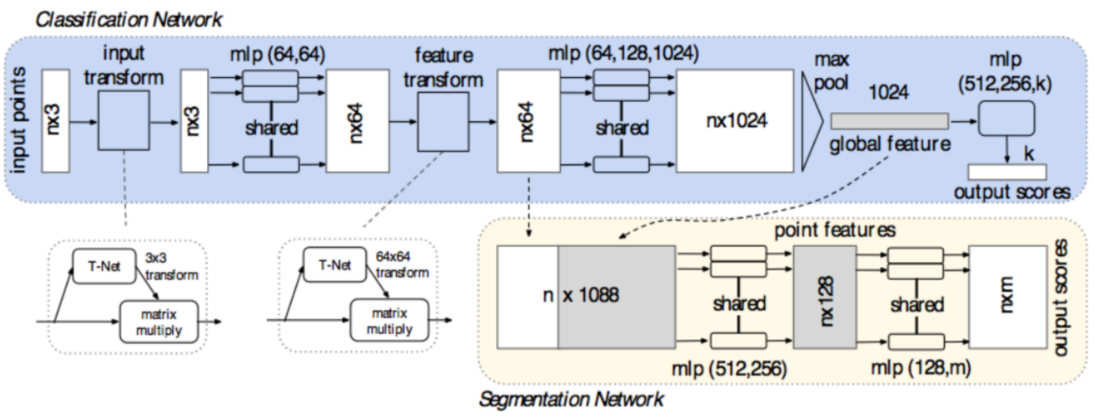


Figure 2.6: PointNet network architecture. [2].

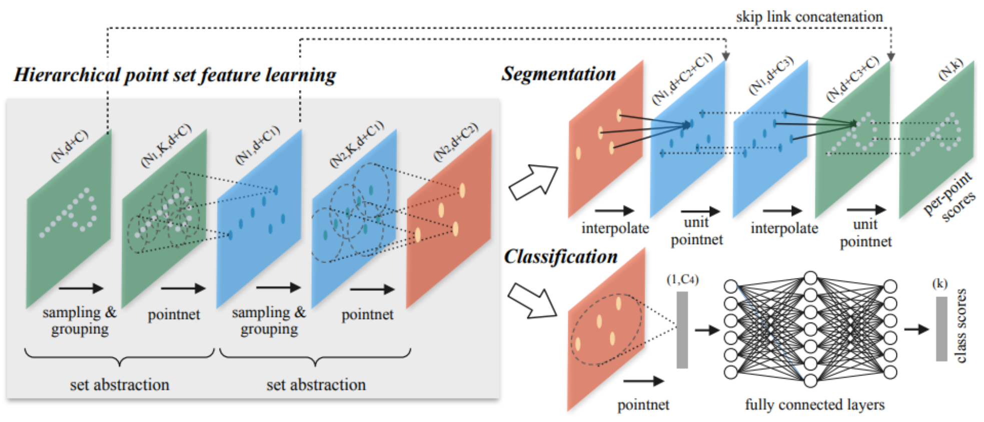


Figure 2.7: PointNet++ network [2].

first to use PointNet in local regions and secondly to aggregate local characteristics to learn the object's hierarchical function. Figure 2.7 demonstrates the method of hierarchical feature learning and its implementation for the tasks of 3D vision-object segmentation and classification using for example, points in 2D Euclidean space.

2.3 3DOD Fusing Camera and LiDAR Data

2.3.1 Introduction

3DOD aims to locate and classify the object embodied in 3D space and outputs the 3D bounding boxes. This section is devoted to delivering a literature review on 3DOD fusing image and LiDAR data. Figure 2.8 shows 3 typical architectures for 3DOD. For object detection, there are two main methods: the two-stage model and the one-stage model. We are primarily interested in the two-stage model in this dissertation. The two-stage model consists of a 3D bounding box proposal for the first stage and a network for second stage detection that regresses the bounding box in sequential order. The network proposes, at the proposed level, a series of regions which may contain an object of interest, which are moved to the second stage of detection and graded with a category ranking. We split the task into two key directions based on the approaches adopted by researchers in this task: 2D proposal-based and 3D proposal-based network, which is addressed in the following subsections with the typical state of the architectures of the arts that influenced our study.

2.3.2 2D Proposal-based

Today, we are witnessing the power and the excellent performance of the 2D based deep learning methodologies in image classification and object detection. A significant amount of research themes are trying to leverage these off the shelf state-of-the-arts to boost the performance on 3DOD. Typically, in a 3DOD task, these mature 2D image processing techniques are used in the proposal stage of 3DOD. Specifically, many state-of-the-arts leverage the effectiveness and the advantage of 2D object detection to generate the 2D regional proposals and project these proposals to the 3D space. There are two main approaches for the projection. The first method projects the 2D bounding box to a 3D plane. This often results in a frustum shape 3D region that contains the object of interest. Such methods are adopted by F-PointNet [3]. The other approach to deal with the projection is to project the 3D point clouds to the 2D BEV or FV image plane so that the point cloud can contain the rich color information with point-wise 2D semantic information.

In these methods, the core idea is to leverage the mature 2D detector such as MaskRCNN [33], to conduct object detection on BEV or FV image first, and narrow down the 3D region of interest through projection, thus, to reduce the computational complexity. One of the mega that adopted such methods is F-PointNet [3], whose architecture is shown in Figure 2.8 (a). And the core idea is to generate 2D bounding boxes on the FV image and then project the 2D bounding boxes to the 3D space resulting in a projected frustum shaped region in 3D space. These frustrum proposals are then fed in to a detector with backbone to be PointNet [2] to conduct further classification and

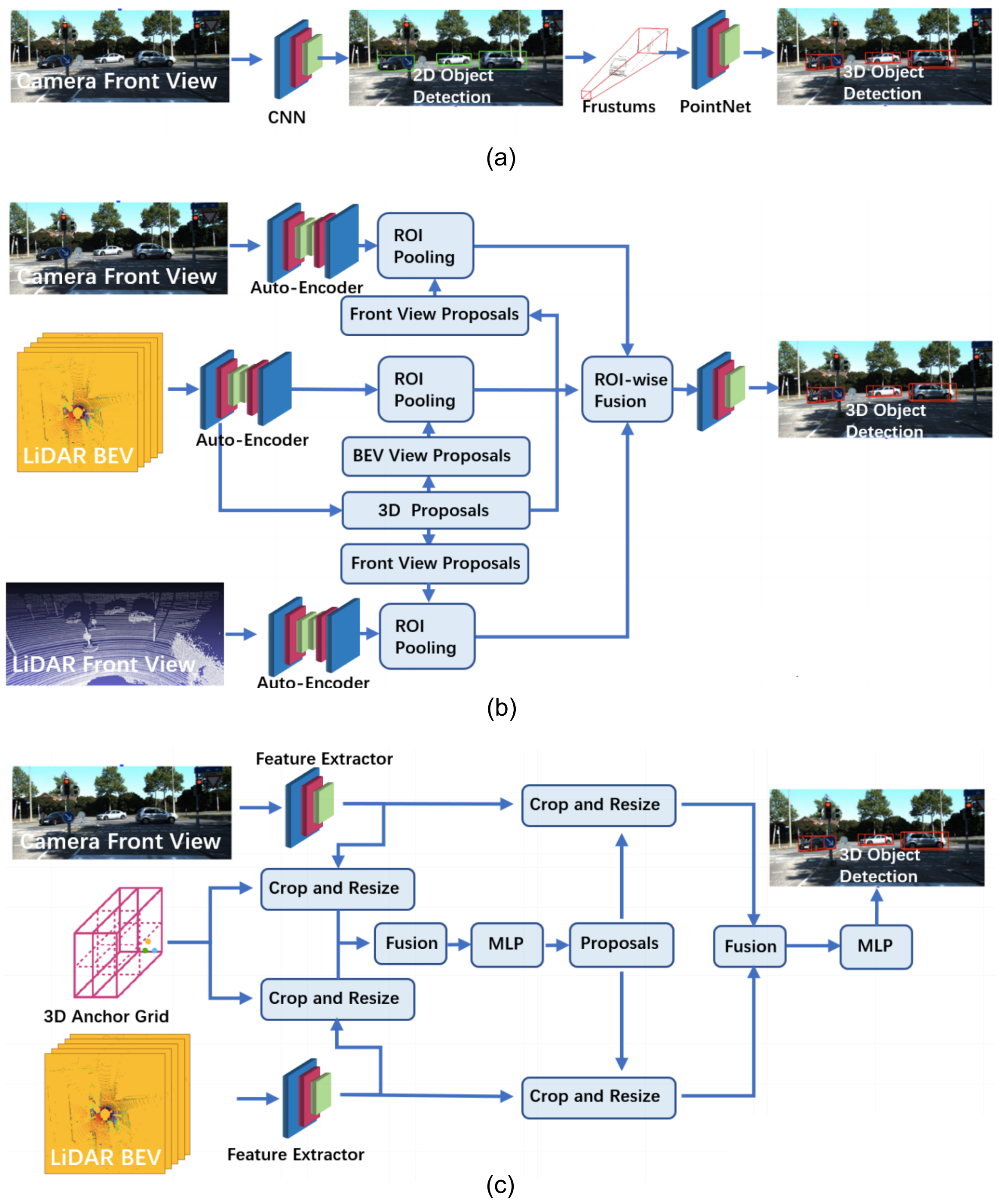


Figure 2.8: A comparison between three state-of-the-arts model architecture on 3DOD, where (a): F-PointNet [3], (b) MV3D [4], (c): AVOD [5].

regression work for 3DOD. The performance is still pretty competitive on Kitti benchmark for object detection.

However, it is computationally cost to consider every point cloud in the frustum proposal, as there are tons of points that belong to the background and noise. Inspired by this, Du et al. [43] proposed a system that extended the idea of F-PointNet [3] by introducing an additional refinement stage in the region proposal network that can detect background points in the frustum proposal and filter these points out before being fed into the nest stage of regression network. This reduces the search space and the computation cost further more on background points.

Another work proposed by Shin et al. [44] did the similar direction, but instead applied a neural network that can be trained to learn the invariance between the background pint clouds for the proposal refinement stage. First, the system processes the 2D image and generates the 2D bounding boxes using Faster-RCNN [45] and the 2D bounding boxes are used to generate multiple cylinder shaped proposals. These proposals are generally smaller than the frustum proposals introduced by Qi et al. in F-PointNet [3] but containing more precise points of interest. These proposals are then fed into the detection network of the second stage using PointNet [2] as the final refinement backbone. By a large margin, this technique outperforms many state-of-the-art methodology. However, based on the assumption that each of the frustum and or cylinder proposal only consists of one object that can be fed directly into a classification network, these methods are implemented, but it is not always valid because due to the occlusion and the size of the object problem, the predicted area can contain several small items. In that scenario, these strategies do not work very well.

One possible solution for the above-mentioned problems was given by the Intensive Point-based Object Detector (IPOD) [46] proposed by Yang et al. The main concept is to adjust the 2D object detector to a 2D semantic segmentation processor that allows the task point-wise, so that the proposal can deal more effectively with the problem of occlusion where the two bounding boxes detected may be strongly overlapped. To filter out the context points, the semantic segmentation processor is applied to the 2D images by projecting the 3D points to the image plane by pixel level to be identified with the corresponding labels. The IPOD considers the production proposals containing fine grained locations and semantic information as the most relevant session in the network of regional proposals.

2.3.3 3D Proposal-based

3D proposal-based methods use the proposals of 3D bounding boxes that are directly generated without additional projection activity from 2D and or 3D data. This method removes the cost of time and error produced from the projection and reduces the 3D search space without the unnecessary training and computational cost of 2D detection, making it more sensitive to the methodologies of pure 3D vision. The most popular technique for the 3D bounding boxes proposal is to manipulate the point cloud representation of the Bird's Eye View (BEV). BEV is the most acceptable representation of such a task because it prevents occlusion situations and preserves raw point cloud information for the full (X, Y) coordinates. These coordinates are of central importance as the transition BEVs and other points of view are required by the system.

One of the most classic and important works that leverages the BEV for the 3D proposal generation is Multiple View 3D Object Detection (MV3D) [4] proposed by Chen et al. in 2017. The architecture of MV3D is shown in Figure 2.8 (b). It combines RGB and LiDAR point cloud information. At the same time, unlike the previous voxel-based methods, it only uses the BEV and FV of the point cloud, which can reduce the amount of calculation without losing too much information. MV3D generate 3D candidate regions, fuse the features at the region of interest (ROI) pooling. The ROI can be formulated as:

$$\mathrm{ROI}_v = \mathbf{T}_{3D\rightarrow views}(P_{3D}), views \in \{\mathrm{BV}, \mathrm{FV}, \mathrm{RGB}\} \tag{2.2}$$

where $\mathbf{T}_{3D\rightarrow views}$ is denoted as the transformation function from 3D point clouds to 2D image space. The ROI-pooling can be formulated as:

$$f_{views} = \mathbf{R}(x, \mathrm{ROI}_{views}), views \in \{\mathrm{BV}, \mathrm{FV}, \mathrm{RGB}\} \tag{2.3}$$

MV3D has outperformed other 3DOD models by a great margin. However in the network architecture, there are limitations. Second, this approach is based on the assumption that there is no vertical overlap from the BEV point of view in the LiDAR sensor and that the target can be viewed entirely from the BEV without any occlusion, which is not possible since smaller objects such as pedestrians and cyclists can overlap the larger or upper objects completely. Second, knowledge loss occurs because during a series of convolution operations of neural networks, the small object needs to be down-sampled 8 times. During this process, small objects, such as pedestrians, may actually be sampled down to less than 10 points. It is also worth remembering that the proposal's repetitive operations that contribute to the need for computation are also a big downside.

To mitigate these flaws, Ku et al. proposed Aggregate View Object Detection (AVOD) [5] to boost the performance of 3DOD on small objects. The system framework of AVOD is illustrated in Figure 2.8 (c), where we can observe that AVOD first changed the input in MV3D [4] by replacing them to FV RGB image and BEV point clouds. Another improvement is the 3D anchor boxes. Furthermore, AVOD employs an auto-encoder based feature extractor inspired by the FPN network to upsample the feature maps down sampled by the consecutive CNN layers so that a full resolution feature map can be generated and used to detect smaller objects without the information loss. The proposed first stage region proposal network conducted the fusion at feature level by extracting the equal sized feature matrixes from both image data and the point cloud data and reduced the dimensionality by employing a 1×1 convolution layer, followed by a crop and resize operation and output a 3×3 equal sized feature vectors. The fusion happens at the next operation, where the two feature vectors are combined through an element-wise mean operation and generates the 3D bounding box proposals. The next step detection network further crops and resizes the feature maps from both image data and point clouds. The cropped feature maps are then combined into one feature vector through another element-wise mean operation that fuses the two. And the last few fully connected layers are deployed to conduct the bounding box regression and score classification tasks. One of the highlights authors stated is their way of 3D bounding box encoding. AVOD encodes the 3D bounding boxes with only 4 (X, Y) coordinates of corner points of the bottom space and two heights, resulting in 10 dimensional regression task that reduce the learnable parameters and boosted the performance.

Although AVOD did a mega work on fusing the image and the point clouds at multi stage and performs well on small objects, there are still a few flaws needed to be improved. First of all, the introduced 1×1 convolution layer deployed on the outputs of the feature extractor is a redundant operation, as the feature experiences the upsampling and the downsampling at the same time, which results in the computation redundancy. Second, the 3D encoding is not the most simple method to represent a 3D bounding box, as there're many physical alignment on the coordinates of the encoding due to the physical geometry of the cuboid. These physical constraints can result in learning redundancy, as they will be refined sooner or later. Third, there is no loss function designed to train the network properly so that the network can learn the imbalance of the fore-background invariance.

CHAPTER 3

Methodology

3.1 Overall Framework

As shown in Figure 3.1, our architectural diagram for 3DOD consists of three main modules: the feature extractor, the 3D region proposal network (RPN), and the second stage detection network. As a high level description, the feature extractor is composed of multiple CNNs that learns the features of the input, and applied both to the RPN and the second stage detection network. The 3D RPN aims to propose a set of 3D regions of interest, whose output is the feature crops which is then passed to the second stage detection network for the bounding box refinement and the classification.

Specifically, the 2D FV image and 3D BEV point clouds are preprocessed to generate the 2D channel maps representation as input to the system. The two identical feature extractors down samples the two inputs through encoder modeled after VGG-16 [29] and up samples through decoder resulting in full resolution feature maps. The output of the feature extractor is the 16 dimensional feature map with the same size as the input image or point cloud slices. The feature map is of high power of representing the high resolution and at the same time does have the high dimension as compared to [5] (256-dimension) that takes too much cost for the computation. A set of prior set anchor boxes are projected to the feature map of the two inputs and the feature crops are generated by the crop and resize operation. The output feature vector representing each feature feature crop is resized to the same size and fused through an element-wise mean operation in the 3D proposal generation network. The fused feature crops are then fed through 2 fully connected layers to regress the axis aligned 3D bounding box. The 3D proposals are then cropped on the feature map and then fused through another fusion operation. The final fully connected layers are then applied to these fused proposals to generate the classification score and the final bounding box regression. We will introduce each module in the following sections in more detail.

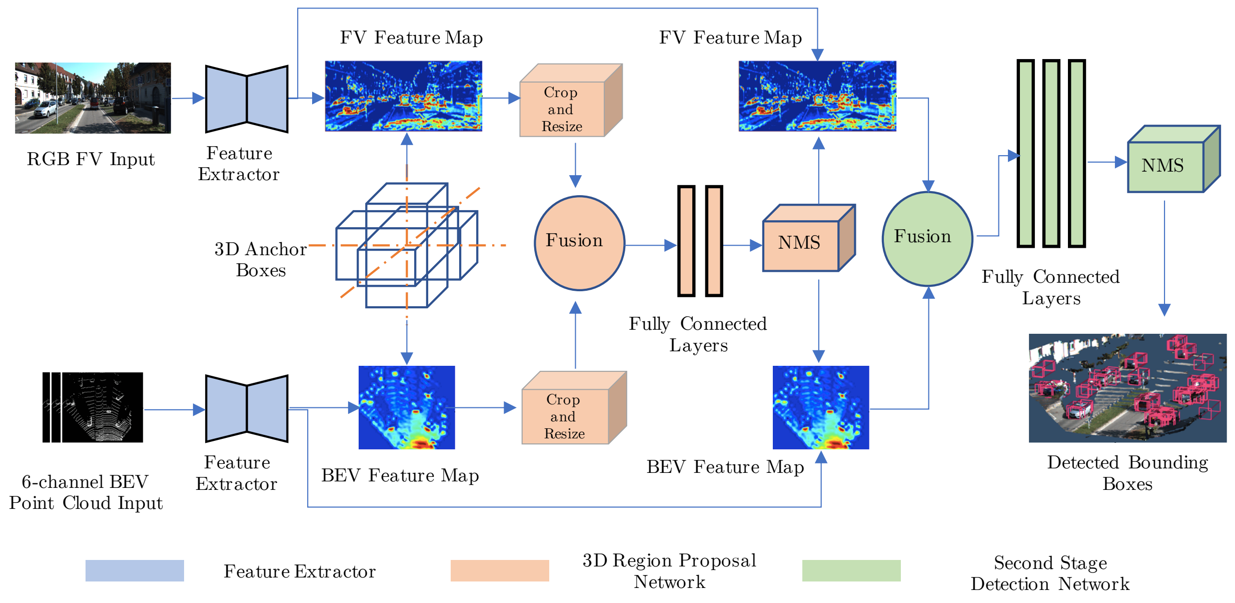


Figure 3.1: The overall framework of the proposed system. The system is composed of three main module: the feature extractor, the 3D RPN, and the second stage detection for bounding box regression and category classification. The feature extractors are shown in blue, the region proposal network in orange, and the second stage detection network in green.

3.2 The Inputs

The inputs to the system is the data from camera in front view (FV) and LiDAR from bird eye's view (BEV). Most existing works usually leverage 3D grid representation to embody the 3D point clouds. While the 3D gird representation retains much more raw 3D information, it usually requires much more computational cost for the complex feature extraction. To properly and efficiently encode the 3D point cloud inputs while not losing too much 3D information, we follow the work of MV3D [4] to preprocess the point cloud input visualized in Figure 3.3.

Another reason to process the input data representation to this is to make assure the point cloud data and the image data to be represented as the same data format while keeping the 3D depth information, allowing the both inputs can be applied by mature deep learning frameworks such as VGG [29] and ResNet [27] to extract the features, which were invented for 2D images.

3.2.1 The RGB Image Input

Due to the high resolution compared to the LiDAR point cloud input, the RGB image data is expected to reflect richer color information and the classification tasks for small items. We choose the front view image because it can provide the smaller objects with a rich function and is easier

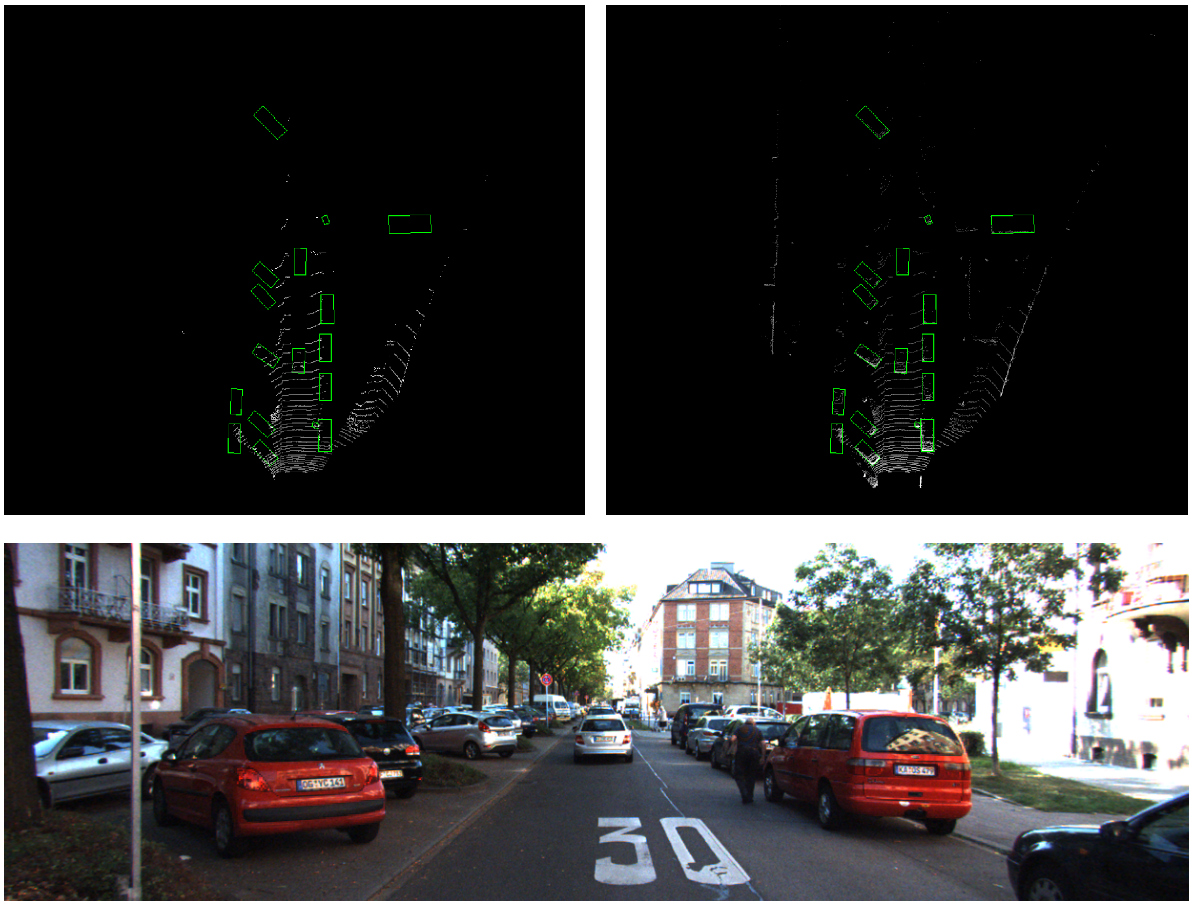


Figure 3.2: The visual representation of the inputs of the proposed 3DOD architecture. The representation of both RGB image and point cloud inputs, top representing the 2D RGB image from front view captured by camera, and the bottom represents the bird's eye view maps from point cloud data illustrating 1 height map and the density map.

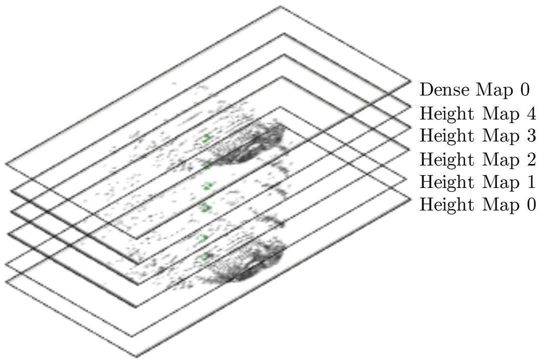


Figure 3.3: Input point cloud feature maps of proposed system.

to interact with the bounding box projection BEV point cloud input. The input image size is 360×1200 pixels and provided by the Kitti dataset captured by using a two color cameras sampled at 10 HZ. We normalize the input image before sending to the system, and the normalization of each image frame is conducted by subtracting the mean value of the RGB channels over the training set. Such example of an input RGB image is illustrated by Figure 3.2 at the bottom.

3.2.2 The Point Cloud Input

We use the LiDAR data provided by Kitti dataset collected by a Velodyne HDL-64E 3D laser scanner sampled at 10 HZ and emitted at 64 laser beams with $100m$ distance range.

The point cloud is processed to be represented by multi-channel 2D map. The mappings are parameterized by the height and density information and is cropped at a $[-40, 40] \times [0, 70]$ meters field of view (FOV) to make sure the point clouds contain the similar field of view with the camera. Point cloud are sampled by the 3D grid representation at a $0.1 \times 0.1 \times 0.1$ meter resolution. For each cell, the height information is computed as the highest point's height of the cell. For the detailed height representation, the 3D space is equally divided by M slices between $[0, 2.5]$ meters along the Z axis, where each slice contains the maximum height of the slice. To encode the density information, we compute the number of points in each cell, and normalized as:

$$\min(1.0, \frac{\log(N+1)}{\log 16}) \tag{3.1}$$

where N specifies the number of points in the cell. The density map is calculated for the whole point cloud and the height map is calculated through M slices resulting a $M + 1$ channel point cloud representation making it convenient to deal with 2D based CNN networks and etc. Figure 3.3 shows an example of the 6-channel mapping of the point cloud data in Kitti dataset [7].

3.3 The Feature Extractor

We are inspired by the methodology described in [5] to extract the feature maps from the corresponding input views. The feature extractor are composed of two stage: an encoder and a decoder, shown as Figure 3.4. The encoder is inspired by VGG-16 [29] architecture but made several modifications shown in Table 3.1 and Table 3.2. We adopt the first three convolution layers in VGG-16 [29] with the max pooling layers and use only half of the layers to extract high level feature representations for both the two inputs. The features has an higher capability of representation but in resolution level, they are $8\times$ lower as compared to the input.

The reason for designing the decoder in the feature extractor is discussed in following. In the BEV view, small objects such as pedestrian and cyclist are occupying very small space, usually in a $[0.4, 1.5]$ range. By down sampling the features of these small objects from BEV point cloud may result in only one or few points representing the objects. Therefore, inspired by Feature Pyramid Network (FPN) proposed by Ren et al. [12], a bottom up decoder is designed to upsample these low resolution feature vectors to a higher level, as shown in Figure 3.4. In AVOD-FPN [5], this is achieved by employing a set of transpose convolution layers and the concatenation of the corresponding feature map out from each layer of the encoder to the decoder, and finally fuse the two via a 3×3 convolution layer. However, the transpose convolution conducts the learning procedure of the upsampling operation while introducing the additional parameters and is occupying the computation memory. In our improved version, we replace the learnable transpose convolution layer by the single $2\times$ bilinear upsampling layer, which reduce the computing time by a significant margin, while keeping the same high resolution representational property by the concatenation of the corresponding convolution layer. As discussed in the ablation study in Chapter 4, there is no significant drop on the accuracy but the system achieved a faster computation speed by a large margin.

Table 3.1 and Table 3.2 illustrates an detailed information on each layer of the feature extractor for both RGB image input branch and the BEV point cloud input branch. The input size of the RGB image is $(360 \times 1200 \times 3)$ pixels representing the resolution and the RGB branch, and the output size of the RGB feature map is $(360 \times 1200 \times 16)$. For the BEV point cloud, the input size is $(700 \times 800 \times 6)$ representing the 6 channel point cloud maps as disscussed in Section 3.2.2. However, in order to allow the even divisions for the max pooling layers, the point clout input is appended with padding resulting in $(704 \times 800 \times 6)$ as the input. The output size of the feature extractor of point cloud branch is $(700(704) \times 800 \times 16)$.

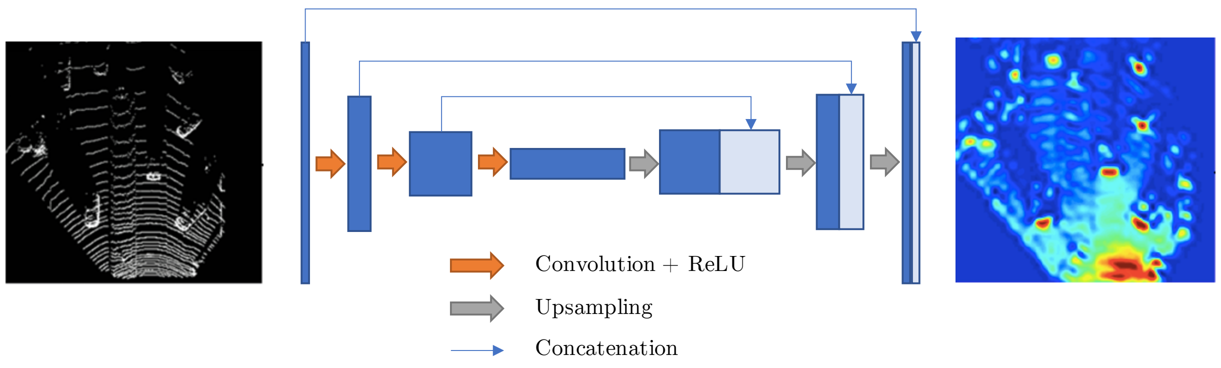


Figure 3.4: The structure of the proposed feature extractor. This shows an example of int BEV point cloud input branch.

Table 3.1: The architecture of the feature extractor with RGB FV image input.

Number of Layers	Operation	Output Size	Kernel Size
-	FV RGB Image	$360 \times 1200 \times 3$	-
2	Conv 1	$360 \times 1200 \times 32$	3×3
1	Maxpool	$180 \times 600 \times 32$	2×2
2	Conv 2	$180 \times 600 \times 64$	3×3
1	Maxpool	$90 \times 300 \times 64$	2×2
3	Conv 3	$90 \times 300 \times 128$	3×3
1	Maxpool	$45 \times 150 \times 128$	2×2
3	Conv 4	$45 \times 150 \times 256$	3×3
1	Upsample 3	$90 \times 300 \times 256$	2×2
1	Pyramid_fusion 3	$90 \times 300 \times 64$	3×3
1	Upsample 2	$180 \times 600 \times 64$	2×2
1	Pyramid_fusion 2	$180 \times 600 \times 32$	3×3
1	Upsample 1	$360 \times 1200 \times 32$	2×2
1	Pyramid_fusion 1	$360 \times 1200 \times 16$	3×3

Table 3.2: The architecture of the feature extractor with BEV point cloud input.

NUMBER OF LAYERS	OPERATION	OUTPUT SIZE	KERNEL SIZE
-	BEV Point Cloud	$700 \times 800 \times 6$	-
2	Conv 1	$704 \times 800 \times 32$	3×3
1	Maxpool	$352 \times 400 \times 32$	2×2
2	Conv 2	$352 \times 400 \times 64$	3×3
1	Maxpool	$176 \times 200 \times 64$	2×2
3	Conv 3	$176 \times 200 \times 128$	3×3
1	Maxpool	$88 \times 100 \times 128$	2×2
3	Conv 4	$88 \times 100 \times 256$	3×3
1	Upsample 3	$176 \times 200 \times 256$	2×2
1	Pyramid_fusion 3	$176 \times 200 \times 64$	3×3
1	Upsample 2	$352 \times 400 \times 64$	2×2
1	Pyramid_fusion 2	$352 \times 400 \times 32$	3×3
1	Upsample 1	$704 \times 800 \times 32$	2×2
1	Pyramid_fusion 1	$704 \times 800 \times 16$	3×3

3.4 3D Region Proposal Network

3.4.1 3D Anchor Boxes

In 3DOD, the region proposal network (RPN) aims to reduce the offset (differences) between the proposals and a series of preset 3D bounding boxes, similar to the 2D two-stage detector. These prior bounding boxes are referred to as 3D anchor bounding boxes, and is determined by the sensors and the training images. These 3D anchor boxes are parameterized by

$$\text{Anchor}_{3D} = \{(x, y, z), (l, w, h)\} \tag{3.2}$$

where (x, y, z) represents the center of the anchor bounding box, and (l, w, h) specifies the axis aligned dimentions (in meters) of the anchor bounding box as illustrated in Figure 3.5. For each 3D anchor box in BEV, the anchor encodings

$$\text{Anchor}_{3D_{BEV}} = (x_{BEV}, y_{BEV}, l_{BEV}, w_{BEV}) \tag{3.3}$$

can be obtained by discretizing (x, y, l, w). In order to generate the 3D anchors, the (x, y) coordinates are sampled with an interval of 0.5 meters in BEV. For the car detection, (l, w) of the anchor boxes takes values with in $\{(3.9, 1.6), (1.0, 0.6)\}$ and the height h is constrained to 1.68 meters

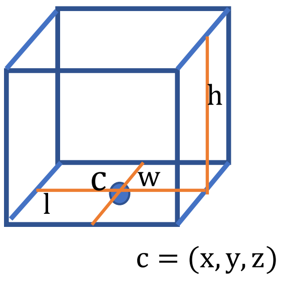


Figure 3.5: The encoding of the 3D anchor boxes.

which is determined by clustering the data samples in the training set of Kitti benchmark. If the anchor boxes does not contain any points or vey few 3D points, the anchors are then removed. These operations make each frame containing nearly $(80k, 100k)$ non-vacant 3D anchor bounding boxes.

3.4.2 Generating Feature Crops

Given an anchor box in 3D, for each input view (RGB and point cloud), the region of interest is obtained by projecting the 3D anchor box on the input views. Then, each region of interest are used for extracting the corresponding feature crops. During the crop and resize operation, these feature crops are then resized bilinearly to 3×3 to get the feature vectors of equal length so that they can conduct fusion in a feature vector element-wise operation.

To configure the input to the RPN, which is to extract the feature crops for each anchor box from the specific feature maps in the feature extractor, we adopt the Input Size Configuration [47] (crop and resize) to conduct the crop and resize operation to generate feature crops shown as Figure 3.6. If we denote the crop and resize operation as $\mathbf{CR}()$, then the feature crops is obtained by:

$$f_V = \mathbf{CR}(\text{Anchor}_{3D}, V), V \in \{\text{BEV}, \text{RGB}\} \tag{3.4}$$

where f_V of size 3×3 is denoted by the feature crops generated from input view V through the projection of Anchor_{3D} to the corresponding input view.

As the feature crops generated by the projection of anchor boxes to the image and BEV point cloud are bilinearly resized to 3×3 of equal length feature vectors, the fusion operation here fuses the two, shown as Figure 3.6 and can be defined as:

$$f_{\text{RPN}} = \frac{1}{2}(f_{BEV} + f_{RGB}) \tag{3.5}$$

which is fused through an element-wise operation, resulting in a 3×3 dimensional feature vector

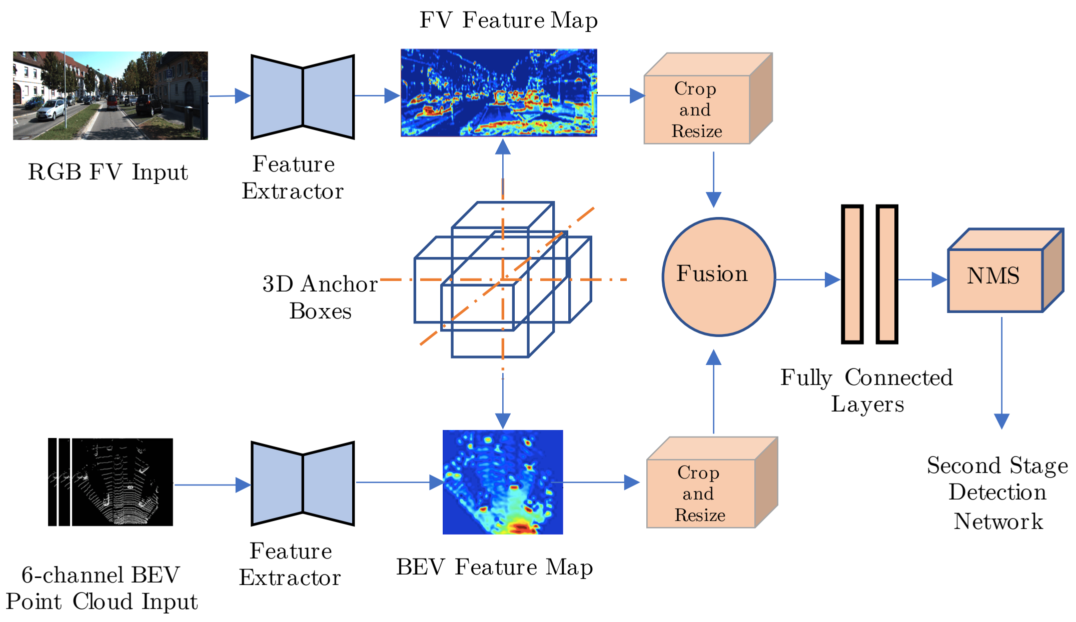


Figure 3.6: Dimensionality reduction and generation of feature crops in region proposal network.

crops generated as the output.

3.4.3 3D Proposal Generation

The outputs of the crop and resize operation are equal-sized feature crops from both views, which are fused via an element-wise mean operation. Two task specific branches [9] of fully connected layers of size 256, use the fused feature crops to regress axis aligned object proposal boxes and output an object/background "objectness" score. 3D box regression is performed by computing the difference in centroid and dimensions between anchors and ground truth bounding boxes. Smooth L1 loss is used for 3D box regression, and cross-entropy loss for "objectness". Similar to [6], background anchors are ignored when computing the regression loss. Background anchors are determined by calculating the 2D IoU in BEV between the anchors and the ground truth bounding boxes. For the car class, anchors with IoU less than 0.3 are considered background anchors, while ones with IoU greater than 0.5 are considered object anchors. For the pedestrian and cyclist classes, the object anchor IoU threshold is reduced to 0.45. To remove redundant proposals, 2D non-maximum suppression (NMS) at an IoU threshold of 0.8 in BEV is used to keep the top 1024 proposals during training. At inference time, 300 proposals are used for the car class, whereas 1024 proposals are kept for pedestrians and cyclists.

We do 3D box regression by regressing to

$$t = (\delta x, \delta y, \delta z, \delta l, \delta w, \delta h) \tag{3.6}$$

which is similarly to RPN [12]. $(\delta x, \delta y, \delta z)$ are the center offsets normalized by anchor sizes, and $(\delta l, \delta w, \delta h)$ are computed as

$$\delta s = log(\frac{s_{\text{gt}}}{s_{\text{anchor}}}), s \in \{l, w, h\} \tag{3.7}$$

We use a multi-task loss to classify object/background simultaneously and do regression of the 3D box. In particular, for the ”objectness” loss, we use class-entropy and for the 3D box regression loss, Smooth L1 [11]. When calculating the loss of box regression, context anchors are ignored. We compute the IoU overlap between anchors and ground truth bird's eye view boxes during preparation. If its overlap is above 0.7, and negative if the overlap is below 0.5, an anchor is considered to be positive. Anchors with an intermediate overlap are overlooked. Since the LIDAR point cloud is sparse, which results in several empty anchors, during both training and research, we eliminate all empty anchors to decrease computation. By computing an integral image over the point occupancy map, this can be accomplished. The network generates a 3D box for each non-empty anchor at each location on the last convolution function diagram. We enforce Non-Maximum Suppression (NMS) on the bird's eye view boxes to decrease redundancy.

3.4.4 Loss Function

Smooth L1 loss is used for 3D box regression, and cross-entropy loss for “objectness”. Background anchors are ignored when computing the regression loss. Back- ground anchors are determined by calculating the 2D IoU in BEV between the anchors and the ground truth bounding boxes. For the car class, anchors with IoU less than 0.3 are considered background anchors, while ones with IoU greater than 0.5 are considered object anchors. This way, a single ground-truth box may assign positive labels to multiple anchors. Anchors that are neither positive or negative do not contribute to the training objective. To remove redundant proposals, 2D NMS at an IoU threshold of 0.8 is used to keep the top 1024 proposals during training. For NMS, tensorflow API is used which selects a subset of bounding boxes based on scores, by pruning away boxes that have high IoU overlap with previously selected boxes. At inference time, 300 proposals are used for the Car class.

We minimize an objective function following the multi-task loss in Fast R-CNN [45]

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{3.8}$$

In the above loss formulation, i is the index of an anchor in a mini-batch and p_i is the predicted probability of anchor i being an object. The ground-truth label p_i^* is 1 if the anchor is positive and is 0 if the anchor is negative. t_i is a vector representing the 6 parameterized coordinates of the predicted bounding box, and $t_i^*$is the ground-truth box associated with a positive anchor. The classification loss L_{cls} is log loss over two classes (object vs. non-object). For the regression loss, we use $L_{reg}(t_i, t_i^*) = R_{reg}(t_i, t_i^*)$ where R is the robust loss function as define in the equation. The term $p_i^* L_{reg}$ means the regression loss is activated only for positive anchors $p_i^* = 1$ and is disabled otherwise $p_i^* = 0$. The output of the cls and reg layers consist of p_i and t_i respectively. The two terms are normalized by N_{cls} and N_{reg} and weighted by a balancing hyper-parameter λ. In the case of the cls term, it is normalized by the anchor mini-batch size and the reg term is normalized by the number of positive anchors. By default we set λ to 5 for reg which provides the best performance in our experiments.

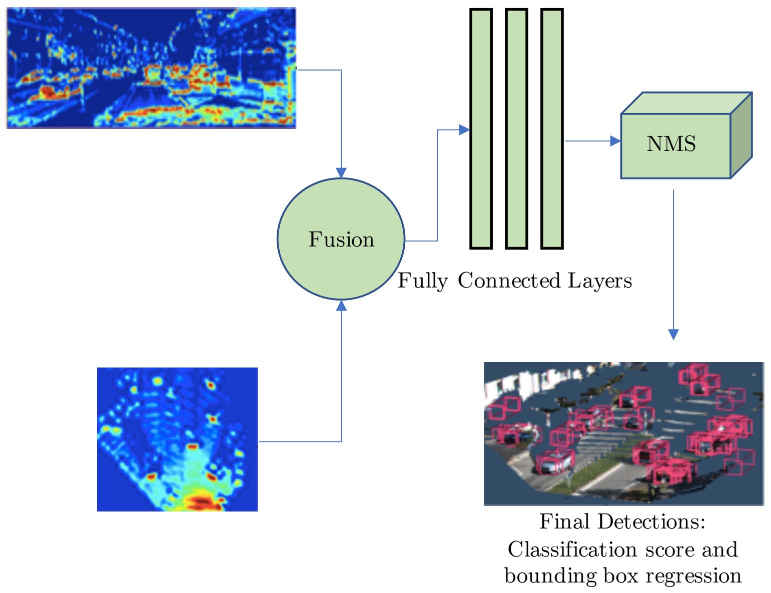


Figure 3.7: The architecture of the second stage detection network.

3.5 The Second Stage Detection Network

3.5.1 3D Bounding Box Encoding

In [6], Chen et al. claim that 8 corner box encoding provides better results than the traditional axis aligned encoding previously proposed in [7] However, an 8 corner encoding does not take into account the physical constraints of a 3D bounding box, as the top corners of the bounding box are forced to align with those at the bottom. To reduce redundancy and keep these physical constraints, AVOD [5] propose to encode the bounding box with four corners and two height values representing the top and bottom corner offsets from the ground plane, determined from the sensor height. The regression targets are therefore $(\delta x_1 ... \delta x_4, \delta y_1 ... \delta y_4, \delta h_1, \delta h_2)$.

However, the 4 corner encoding does not take into account the physical constraints of a 3D bounding box, as the top corners of the bounding box are forced to align with those at the bottom. To reduce redundancy and keep these physical constraints, we propose to encode the bounding box with only the (X, Y) coordinates of two corners and two height values representing the top and bottom corner offsets from the ground plane, determined from the sensor height. The regression targets are therefore $(\delta x_1, \delta x_2, \delta y_1, \delta y_2, \delta h_1, \delta h_2)$.

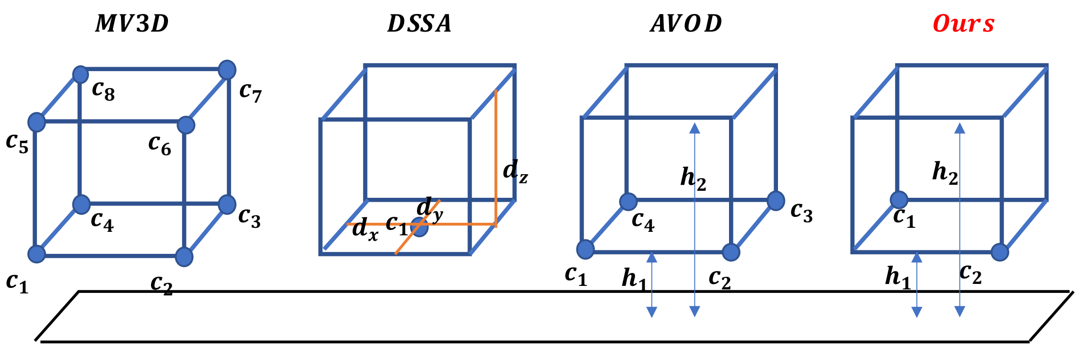


Figure 3.8: A visual comparison between the 8 corner box encoding proposed in [6], the axis aligned box encoding proposed in [4], the 4 corner encoding proposed in [5] and out two corner encoding.

3.5.2 Dimentionality Reduction

As shown in Figure 2.8 (c), in the region proposal stage, AVOD [5] leverages one 1×1 convolution layer employed on the feature map generated by feature extractor from both two inputs, stating that this can reduce the dimensionality of the further computation complexity. However, in the second stage detection network, AVOD uses the full size feature map $(M \times N \times 256)$generated by the feature extractor, because the number of proposals is in an order of magnitude lower than the the number of anchors. This can result in a large memory overhead per input view. For example, if the system is running with 32 bit floating representation, extracting 7×7 feature crops from the feature map with 256 dimension for 100K proposals would require 5 gigabytes[1] of memory. In our proposed system, the output of the feature extractor is a feature map with only 16 dimension for each input view, allowing the system to process the fused feature crops with 100K proposal boxes with megabytes.

3.5.3 Generating Final Detections

Similar to the RPN, the inputs to the multi-view detection network are feature crops generated from projecting the proposals into the two input views, shown in Figure 3.7. As the number of proposals is an order of magnitude lower than the number of anchors, the original feature map with a depth of $D = 32$ is used for generating these feature crops. Crops are resized to 7 x 7 from both input views and then fused with an element-wise mean operation. For each proposal, a single perceptron module of three fully connected layers of size 2048 process the fused feature crops to output box regression, orientation estimate, and category classification. Like the RPN, for the bounding box and orientation vector regression tasks, we employ a multi-task loss combining two Smooth L1

[1] $100,000 \times 7 \times 7 \times 4 \times 256 bytes$

losses, and a cross-entropy loss for the classification task. In the assessment of regression failure, ideas are only considered if they have at least a 0.65 or 0.55 2D IoU in BEV with ground truth boxes for the car and pedestrian/cyclist classes, respectively. NMS is used at a threshold of 0.01 to eliminate overlapping detections.

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*) \tag{3.9}$$

3.6 Training

We train trained the system for the car class for 3D object detection tasks. In an end-to-end fashion, the RPN and the detection networks are trained jointly using mini-batches containing one image with 512 region of interests, respectively. Using an ADAM optimizer[39] with an initial learning rate of 0.0001, the network is trained for 120K iterations and the learning rate decays exponentially every 100K iterations with a decay factor of 0.1. We use dropouts with a probability of 0.5 in all the fully connected layers in the perceptron module for regularization. On all fully connected layers, Batch Normalization is also applied. Normalization of batches leads to convergence changes thus eliminating the need for other types of regularization.

3.6.1 Path Drop

By applying the following, path-drop training is implemented. We create 3 probabilities randomly at each iteration, where the first decides the probability of preserving the image branch, the second determines maintaining the BEV branch, and the third makes the final decision in the event that both branches are dropped off. So if our random probability is greater than the hyper-parameter probability set, the branch is dropped. For example, we decide to drop either the BEV or the image branch at each iteration, but we never drop both. While it may be an ideal approach to drop one input during training to make the network resilient to missing inputs such as lack of image or BEV, our experiments overall showed that the results were insensitive to path-drop. The probability of losing either branch was therefore set at 0.9 in all our experiments, suggesting that both branches were maintained most of the time during training.

3.6.2 Data Augmentation

For the training data, data augmentation techniques such as flipping and PCA jittering methods are applied to boost the training performance. Augmentation of data tends to improve training instances and in turn, decreases overfitting. In order to match the flipped object position, both image and point cloud data can be flipped horizontally where the corresponding labels are also flipped. In the training images, PCA jittering consists of altering the intensity of the RGB channels. This is achieved by computing PCA in the training data on all RGB values, and then adding to the images multiple of the found concept components.

CHAPTER 4

Experiment

4.1 Dataset

The proposed model is trained using a dataset created from KITTI Vision Benchmarking Suite [7]. KITTI datasets are captured by driving a car around the city of Karlsruhe in Germany. The recording platform is a Volkswagen Passat B6, equipped with the following sensors: (also shown as in Figure 4.1)

- One Inertial Navigation System (GPS/IMU): OXTS RT 3003
- One Laser scanner: Velodyne LiDAR HDL-64E
- Two Grayscale cameras, 1.4 Megapixels: Point Grey Flea 2 (FL2-14S3M-C)
- Two Color cameras, 1.4 Megapixels: Point Grey Flea 2 (FL2-14S3C-C)
- Four Varifocal lenses, 4-8 mm: Edmund Optics NT59-917

The LiDAR laser scanner spins at 10 frames per second, capturing approximately 100k points per 360 degree scan, which is known as a point cloud. The camera images are cropped to 1382 x 512 pixels. The cameras are triggered at 10 frames per second by the laser scanner (when facing forward) with dynamically adjusted shutter time [7].

For object groups such as cars, buses, trucks, pedestrians, cyclists and trams, the 3D object KITTI benchmark offers 3D bounding boxes that are manually labeled based on the camera details in 3D point clouds. Objects outside the camera plane are unlabeled to prevent false-positive detection. The KITTI 3D object dataset consists of 7481 clouds of training points (and images) with labels used for training and validation and 7518 clouds of points (and images) without labels used for testing. KITTI also offers three levels of detection evaluation: simple, moderate and hard, depending on the size of the target, state of occlusion and level of truncation. For easy items, the minimum pixel height is 40px, which roughly corresponds to vehicles within 28m. There are 25px for moderate and difficult level objects, corresponding to a minimum distance of 47m.

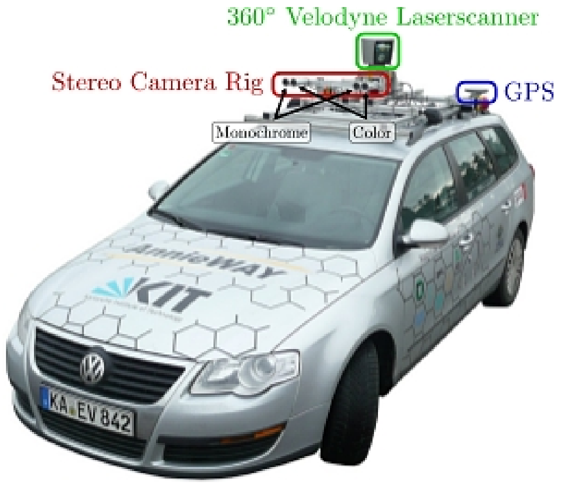


Figure 4.1: Volkswagen Passat B6, KITTI sensor setup. [7]

4.2 Evaluation Metrics

This section discusses the evaluation metrics used to evaluate the proposed 3DOD system. For the overall system performance, the average precision (AP) is used to conduct the evaluation introduced in the Subsection 4.2.2.

4.2.1 Precision & Recall

Precision and recall are the most common performance measures for classification and object detection tasks. Precision can be defined as:

$$Precision = \frac{TP}{TP + FP} \tag{4.1}$$

where TP is acronym for True Positive, which indicates a correct detection with existing corresponding ground-truth. FP is False Positive and indicates that an object was detected but it does not have a corresponding ground-truth, i.e. false detection. FN is False Negative, and indicates that an object was in the ground-truth but not detected by the system. The precision metric measures how many of the predicted objects that were true positive. High precision relates to a low false positive rate. A precision score of 1.0 for a class indicates that every object that predicted to be in that class, is classified correctly.

Recall metric is the fraction of correct positive predictions among the all positive observations in actual class given by:

$$Recall = \frac{TP}{TP + FN} \tag{4.2}$$

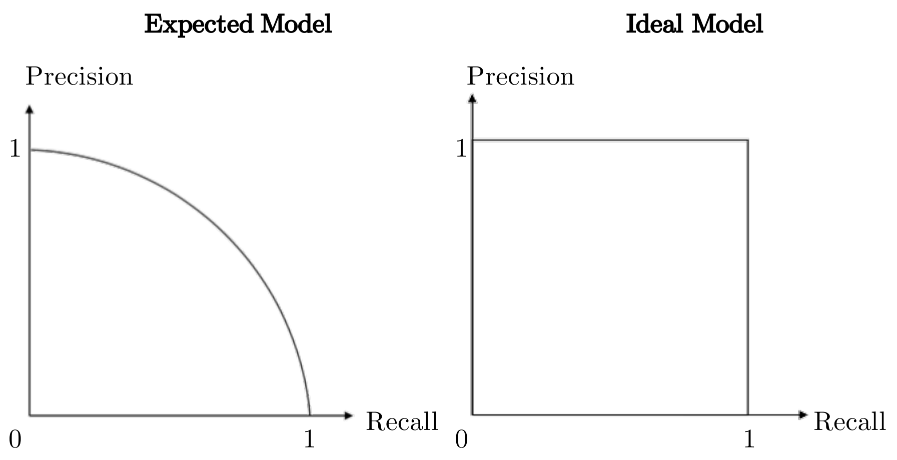


Figure 4.2: The precision-recall curve. Precision-recall curve shows the trade-off between precision and recall. The closer the curve to (1,1), the higher the performance.

The recall metric measures how many of the true positives that were found. High recall relates to low false negative rate. A recall score of 1.0 for a class indicates every object in that class has been found and predicted correctly.

4.2.2 Average Precision

The accuracy and recall of a model with zero errors is equal to 1, which is challenging to attain. The accuracy-recall curve is used in practice to observe what the optimal balance between accuracy and recall is. It shows the trade-off between both metrics for different thresholds, as shown in Figure 4.2. The AP summarizes the shape of the precision-recall curve and is defined as the mean precision at a set of eleven equally spaced recall levels $[0, 0.1, ..., 1]$:

$$AP = \frac{1}{11} \sum_{r \in \{0,0.1,...,1\}} p_{interp}(r) \tag{4.3}$$

The precision at each recall level r is interpolated by taking the maximum precision measured for which the corresponding recall exceeds r:

$$p_{interp}(r) = \max_{\hat{r}:\hat{r} \geq r} p(\hat{r}) \tag{4.4}$$

where $p(\hat{r})$ is the measured precision at recall $\hat{r}$. The objective of interpolating the precision-recall curve is to decrease the effect of the ”wiggle” induced by minor variations in the ranking of examples on the precision-recall curve. The detections are iteratively allocated, determined by bounding box intersection over union, to ground truth labels beginning with the largest overlap. A detection method must have accuracy at all stages of recall in order to achieve a high score. Therefore this metric penalizes techniques that recover only a subset of high-precision detections.

4.3 Evaluation Results

4.3.1 Training and Test Setting

In this section we present the performance of our proposed system by conducting the evaluation on 3D car detection task of the Kitti 3D Object Detection Benchmark [7]. Kitti dataset has 7,481 training and 7,518 test point clouds (frames), comprising a total of 80,256 labeled objects. The ground-truth labels of the test set is not available, but it is possible to evaluate it over the Kitti server [7]. KITTI also provides a common evaluation protocol for object detection that is used in many research papers, where the performance is measured by the Average Precision (AP) and the IoU threshold is 0.7 for Car class (at least 70% overlap with ground truth). Since the test server access is limited, we evaluate the performance of our method on a validation set. We split the training frames into a training set (3,712 frames) and a validation set (3,769 frames), and ensure that frames from training and validation set do not come from the same video sequences.

4.3.2 Results

3D detection results are evaluated using the 3D AP at 0.7 IoU threshold for the Car class. We implemented AVOD [5], MV3D [4], and F-PointNet [3]. The results of Deep3DBox [10] are publicly provided by their work. Other results are tested with a 3D IOU threshold of 0.7 on Kitti [7] validation set (3769 examples) and trained on Kitti train set (3712 examples). A detailed comparison of the performance are presented in Table 4.1. On the validation set, our proposed system outperforms its base network AVOD-FCN [5] by 1.39% on the easy setting, 0.24% on the moderate setting. AVOD-FCN did a better job than us on the hard setting, because they leverages more learnable layers in the feature extractor when conducting the upsampling operation where we only uses the bilinear upsampling. However, by doing this we reduced a significant number of learnable weights that require computation and memory, making it running at a 0.07 seconds per frame as compared to AVOD-FPN's 0.1 seconds, resulting in closer to the real-time application. Our system doesn't outperform F-PointNet [3]. This is because F-PointNet uses the 2D detection to generate the 2D proposals, where 2D detection has already been proved to be a mature technique. A precision-recall curve is also presented in Figure 4.3 of our system compared to AVOD [5].

Table 4.1: A comparison of the 3D-AP (Average Precision) of our proposed system with state-of-arts. We implemented AVOD [5], MV3D [4], and F-PointNet [3]. The results of Deep3DBox [10] are publicly provided by their work. Other results are tested with a 3D IOU threshold of 0.7 on Kitti [7] validation set (3769 examples) and trained on Kitti train set (3712 examples).

METHOD	FUSION LEVEL	PROPOSAL	EASY	MODERATE	HARD	RUN TIME
F-PointNet	Result Level	2D based	83.26%	79.28%	62.56%	0.1
MV3D	Feature Level	3D based	73.59%	65.78%	58.39%	0.1
Deep3DBox	-	-	5.97%	5.06%	3.87%	-
AVOD	Feature Level	3D based	77.26%	73.28%	59.56%	0.08
AVOD-FPN	Feature Level	3D based	80.29%	**76.58%**	63.27%	0.1
Ours	Feature Level	3D based	**81.68%**	76.32%	**64.59%**	**0.07**

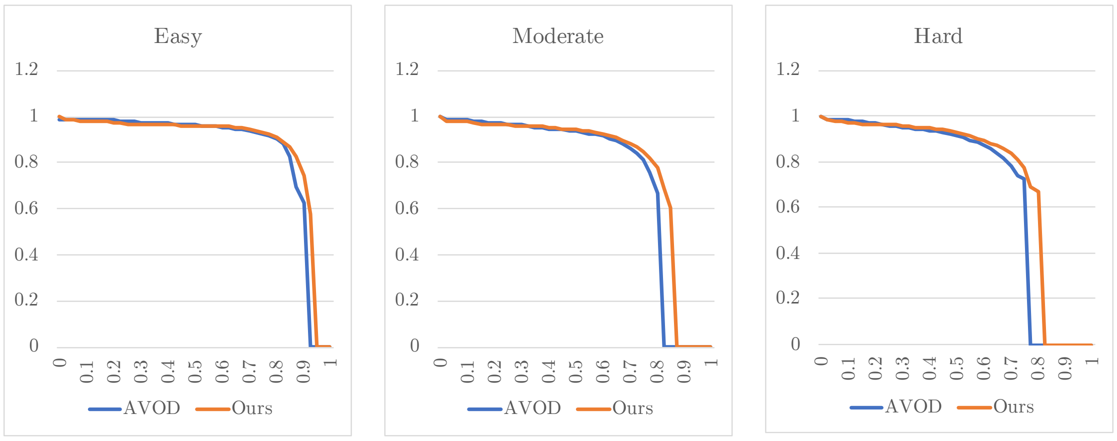


Figure 4.3: A comparison of recision recall curve of the proposed system and AVOD [5].

4.4 Ablation Study

Table 4.2 shows the effect of varying different hyper-parameters on the performance measured by the AP and the number of model parameters of the proposed architecture. The original network uses hyper-parameter values described throughout the thesis up to this point, along with the feature extractor of MV3D. We study the effect of the RPN's input feature vector origin and size on both the proposal recall and final detection AP by training two networks, one using BEV only features, and the other using feature crops of size 1×1 as input to the RPN stage. We also study the effect of different bounding box encoding schemes shown in Figure 3.8.

4.4.1 Bounding Box Encoding

We study the effect of different bounding box encodings shown in Figure 3.8 by training two additional networks. The first network estimates axis aligned bounding boxes, using the regressed orientation vector as the final box orientation. The second and the third networks use our 4 corner and MV3D's 8 corner encodings without additional orientation estimation as described in Section 3.1.5. As expected, without orientation regression to provide orientation angle correction, the two networks employing the 4 corner and the 8 corner encodings provide a lower AP than the original network for the Car class. This phenomenon can be attributed to the loss of orientation information as described in Section 3.1.5.

4.4.2 The Feature Extractor

We trained two feature extractor for the networks, one using ResNet [31] in the FC7 feature, and one using the FPN proposed by Ku et al. [5] to study the ablation and the effectiveness of our proposed feature vector. The results are performed at a 0.7 3D AP threshold on the validation set. Our experiments shows that the proposed feature extractor outperforms ResNet feature extractor, but almost the same as the AVOD-FPN feature extractor. This is because the FPN feature extractor uses the learnable layers to operate the upsampling for the resolution representation. However, we didn't find the learnable layers out performs us by a large margin, while on the easy and hard setting, our feature outperforms FPN. Also, the proposed feature extractor gives a faster running time making the system close to real time application. Thus, we state that our feature extractor is of high representational power for the task.

Table 4.2: Ablation Study. A comparison of the performance of different variations of hyper-parameters, evaluated on the validation set at moderate difficulty. The effect of variation of hyper-parameters on the FLOPs and number of parameters are measured relative to the original network. The results are tested with a 3D IOU threshold of 0.7 on Kitti [7] validation set (3769 examples) and trained on Kitti train set (3712 examples).

Architecture	Number of Parameters	Easy	Moderate	Hard
Base Network	28,127,536	80.24%	75.40%	61.54%
Axis-aligned Box Encoding	-8,196	80.21%	74.41%	61.39%
4 Corners Box Encoding	-4,098	80.28%	73.56%	60.37%
8 Coners Box Encoding	+24,638	79.21%	73.62%	59.82%
2 Corner & 2 Offsets Box Encoding	-8,196	80.63%	75.41%	61.37%
ResNet [29] Feature Extractor	+8,700,298	80.07%	74.37%	58.62%
FPN [5] Feature Extractor	-4,113,536	80.29%	**76.58%**	63.27%
Our Feature Vector + 2 Corner Encoding	**-6,391,116**	**81.68%**	76.32%	**64.59%**

4.5 Qualitative Results

Figure 4.4 presents a qualitative output sample visualized through the work of Ku et al. [8]. The red bounding boxes are representing the ground truth and the green boxes represents the final detection outputs.

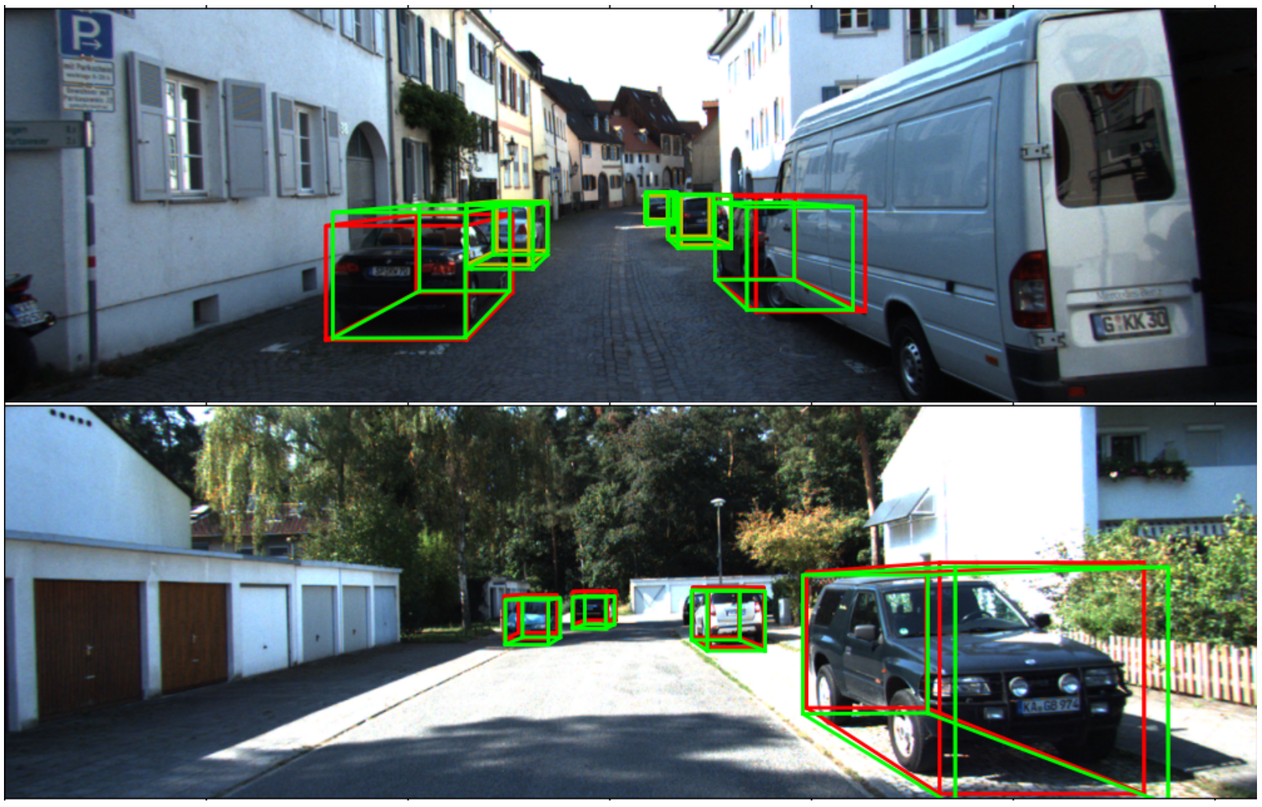


Figure 4.4: A qualitative output of the proposed system relative to KITTI's ground truth on a sample in the validation set. Visualization using [8].

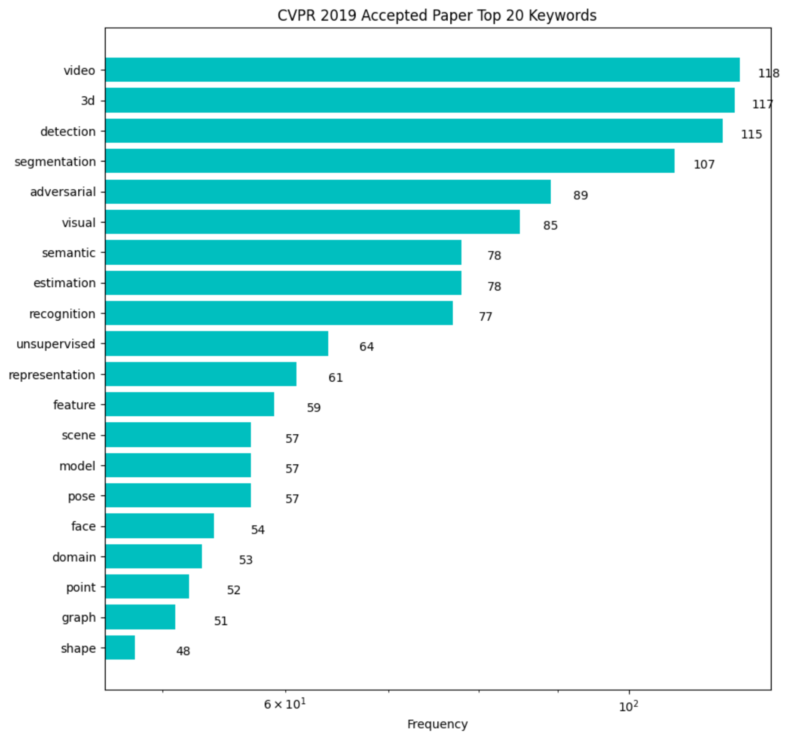
CHAPTER 5

Conclusion and Future Works

A two-stage 3D object detector for autonomous driving scenarios was presented in this thesis. The proposed architecture is distinguished from the state-of-the-art by using a high resolution feature extractor coupled with an integrative fusion RPN framework with less learnable weights. In turn, the device is able to generate precise region proposals and detections at a low computational cost. The performance boost of the method over the state-of-the-art on the 3D Object Detection tasks has been shown by experiments on the KITTI data set.

5.1 Limitations

Although the proposed system's accuracy performance is similar compared to many other systems that directly process LIDAR points, the system still relies on the BEV view. F-PointNet outperforms our system. This may be due to two factors, the reality that F-PointNet uses an off-the-shelf pre-trained 2D detector that can detect smaller and far away objects more accurately leveraging much more 2D information from high resolution representation, and the fact that compared to the BEV view, raw 3D data has much richer 3D embodied information. Ideally, all the available information such as image, BEV and point cloud data should be directly fused by a more acceptable technique. This will however, substantially improve the processing time. Networks such as F-PointNet can also achieve acceptable speed-time because they depend on the outcomes of a 2D detector that produces a limited number of candidate proposals compared to the RPN and thus greatly reduces the processing time. To be able to function efficiently, these approaches often involve a pre-trained 2D network.



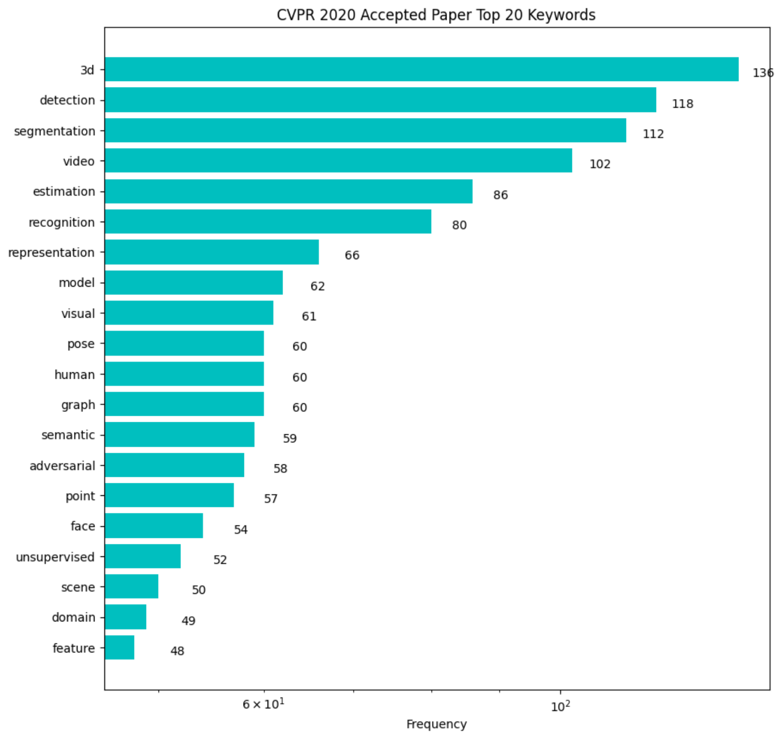


Figure 5.1: Keywords of accepted paper in CVPR 2019 and 2020.

5.2 Future Works

Figure 5.1 shows the top 20 keywords of the accepted papers submitted to CVPR 2019 and 2020, from where we can observe that 3D vision and 3D deep learning are ones of the most popular research themes in the computer vision field. However, take 3DOD as example, the 3D AP ranked top 1 is only 82.83%, which is far from the performance on 2DOD. Among the most successful research on 3DOD, few of them are really exploiting the complete 3D information of the 3D data. Exploiting and fusing the 2D image and 3D data are still considered as a future endeavor.

The most recent 3D detectors process only single frames, one explanation being the overhead computing of 3D data processing. In the case of autonomous driving, however a lot of data can be collected through sequential frames. The network may also be injected with information from the previous frame. The use of recurrent neural networks (RNN) to process video sequences is one potential future task to explore. A joint pipeline for the identification and monitoring of objects is also possible, where tracking information can also be of help in determining the detection performance of objects and likewise.

REFERENCES

[1] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. S., Berg, A., and Fei-Fei, L., "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, Vol. 115, 2015, pp. 211–252.

[2] Charles, R. Q., Su, H., Kaichun, M., and Guibas, L. J., "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 77–85.

[3] Qi, C. R., Liu, W., Wu, C., Su, H., and Guibas, L. J., "Frustum PointNets for 3D Object Detection from RGB-D Data," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 918–927.

[4] Chen, X., Ma, H., Wan, J., Li, B., and Xia, T., "Multi-view 3D Object Detection Network for Autonomous Driving," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6526–6534.

[5] Ku, J., Mozifian, M., Lee, J., Harakeh, A., and Waslander, S. L., "Joint 3D Proposal Generation and Object Detection from View Aggregation," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–8.

[6] Duarte, F., "Self-driving cars: A city perspective," *Science Robotics*, Vol. 4, No. 28, 2019, pp. eaav9843.

[7] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R., "Vision meets robotics: the KITTI dataset," *The International Journal of Robotics Research*, Vol. 32, No. 11, 2013, pp. 1231–1237.

[8] Ku, J., Harakeh, A., and Waslander, S. L., "In Defense of Classical Image Processing: Fast Depth Completion on the CPU," *2018 15th Conference on Computer and Robot Vision (CRV)*, 2018, pp. 16–22.

[9] Cui, Y., Chen, R., Chu, W., Chen, L., Tian, D., and Cao, D., "Deep Learning for Image and Point Cloud Fusion in Autonomous Driving: A Review," *arXiv:2004.05224*, 2020.

[10] Guo, J., Kurup, U., and Shah, M., "Is it Safe to Drive? An Overview of Factors, Metrics, and Datasets for Driveability Assessment in Autonomous Driving," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 21, 2020, pp. 3135–3151.

[11] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., "You Only Look Once: Unified, Real-Time Object Detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[12] Ren, S., He, K., Girshick, R., and Sun, J., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 39, No. 6, 2017, pp. 1137–1149.

[13] Peng, X., Murphey, Y., Stent, S., Li, Y., and Zhao, Z., "Spatial Focal Loss for Pedestrian Detection in Fisheye Imagery," *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019, pp. 561–569.

[14] Huang, P., Cheng, M., Chen, Y., Luo, H., Wang, C., and Li, J., "Traffic Sign Occlusion Detection Using Mobile Laser Scanning Point Clouds," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 18, No. 9, 2017, pp. 2364–2376.

[15] Qi, R., *Deep learning on point clouds for 3D scene understanding*, Ph.D. dissertation, Elect. Eng., Stanford Univ., Stanford, CA, 2018.

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

[17] Maturana, D. and Scherer, S., "Voxnet: A 3d convolutional neural network for real-time object recognition," *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2015, pp. 922–928.

[18] Riegler, G., Ulusoy, A. O., and Geiger, A., "OctNet: Learning Deep 3D Representations at High Resolutions," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6620–6629.

[19] Garcia, D. C., Fonseca, T. A., Ferreira, R. U., and de Queiroz, R. L., "Geometry Coding for Dynamic Voxelized Point Clouds Using Octrees and Multiple Contexts," *IEEE Transactions on Image Processing*, Vol. 29, 2020, pp. 313–322.

[20] Zeng, W. and Gevers, T., "3DContextNet: K-d Tree Guided Hierarchical Learning of Point Clouds Using Local and Global Contextual Cues," *ECCV Workshops*, 2018.

[21] Lei, H., Akhtar, N., and Mian, A., "Octree Guided CNN With Spherical Kernels for 3D Point Clouds," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9623–9632.

[22] Chen, S., Duan, C., Yang, Y., Li, D., Feng, C., and Tian, D., "Deep Unsupervised Learning of 3D Point Clouds via Graph Topology Inference and Filtering," *IEEE Transactions on Image Processing*, Vol. 29, 2020, pp. 3183–3198.

[23] Li, C., Qin, X., Xu, X., Yang, D., and Wei, G., "Scalable Graph Convolutional Networks With Fast Localized Spectral Filter for Directed Graphs," *IEEE Access*, Vol. 8, 2020, pp. 105634–105644.

[24] Boscaini, D., Masci, J., Melzi, S., Bronstein, M. M., Castellani, U., and Vandergheynst, P., "Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks," *Computer Graphics Forum*, Vol. 34, No. 5, 2015, pp. 13–23.

[25] Bracci, F., Drauschke, M., Kühne, S., and Marton, Z., "Challenges in fusion of heterogeneous point clouds," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. 42, No. 2, 2018, pp. 155–162.

[26] Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning*, Cambridge, MA: MIT Press, 2016.

[27] Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., and Bennamoun, M., "Deep Learning for 3D Point Clouds: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020, pp. 1–1.

[28] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, Vol. 60, No. 6, 2017, pp. 84–90.

[29] Simonyan, K. and Zisserman, A., "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556*, 2015.

[30] Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A., "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[31] He, K., Zhang, X., Ren, S., and Sun, J., "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[32] Zhu, X., Hu, H., Lin, S., and Dai, J., "Deformable ConvNets V2: More Deformable, Better Results," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9300–9308.

[33] He, K., Gkioxari, G., Dollár, P., and Girshick, R. B., "Mask R-CNN," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 42, 2020, pp. 386–397.

[34] Dai, J., He, K., and Sun, J., "Instance-Aware Semantic Segmentation via Multi-task Network Cascades," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3150–3158.

[35] Liu, Z., Li, X., Luo, P., Loy, C. C., and Tang, X., "Semantic Image Segmentation via Deep Parsing Network," *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1377–1385.

[36] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P., "Conditional Random Fields as Recurrent Neural Networks," *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1529–1537.

[37] Zhou, K., Yang, Y., Cavallaro, A., and Xiang, T., "Omni-Scale Feature Learning for Person Re-Identification," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 3701–3711.

[38] Xu, J., Zhao, R., Zhu, F., Wang, H., and Ouyang, W., "Attention-aware compositional network for person re-identification," *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2119–2128.

[39] Saquib Sarfraz, M., Schumann, A., Eberle, A., and Stiefelhagen, R., "A pose-sensitive embedding for person re-identification with expanded cross neighborhood re-ranking," *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 420–429.

[40] Li, W., Zhu, X., and Gong, S., "Harmonious attention network for person re-identification," *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2285–2294.

[41] Kesten, R., Usman, M., Houston, J., Pandya, T., Nadhamuni, K., Ferreira, A., Yuan, M., Low, B., Jain, A., Ondruska, P., Omari, S., Shah, S., Kulkarni, A., Kazakova, A., Tao, C., Platinsky, L., Jiang, W., and Shet, V., "Lyft Level 5 Perception Dataset 2020," https://level5.lyft.com/dataset/, 2019.

[42] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., and Anguelov, D., "Scalability in Perception for Autonomous Driving: Waymo Open Dataset," 2019.

[43] Du, X., Ang, M. H., Karaman, S., and Rus, D., "A General Pipeline for 3D Detection of Vehicles," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3194–3200.

[44] Shin, K., Kwon, Y. P., and Tomizuka, M., "RoarNet: A Robust 3D Object Detection based on RegiOn Approximation Refinement," *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2019, pp. 2510–2515.

[45] Hsu, S., Wang, Y., and Huang, C., "Human Object Identification for Human-Robot Interaction by Using Fast R-CNN," *2018 Second IEEE International Conference on Robotic Computing (IRC)*, 2018, pp. 201–204.

[46] Yang, Z., Sun, Y., Liu, S., Shen, X., and Jia, J., "IPOD: Intensive Point-based Object Detector for Point Cloud," *arXiv:1812.05276*, 2018.

[47] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., and Murphy, K., "Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3296–3297.