# VE581 Group9 Final Report

3D Object Detection Using Point Cloud

Zhang Ting
UM-SJTU Joint Institute
515370910023

Shi Jiecheng
UM-SJTU Joint Institute
515370910022

## Abstract

*3D object detection and localization are becoming more and more important in daily life including the autonomous driving and traffic condition control. Our project aims to make prediction and outline the 3D estimate box about the object in an image containing car, pedestrian and cyclist. Also, in our work, we design and modify several models to compare their performance and figure out our final pipeline. The dataset comes from KITTI and the results are shown in mayavi UI view.*

### I. Introduction

As the popularity of autonomous driving grows, multiple methods are figured out including sensor detection, sonar detection, image real-time detection. Due to the development of the computer vision, 3D detection becomes more and more useful in these fields. By estimating the 3D box outline of the car and other traffic related objects, the control of the autonomous driving and traffic can be easier. 3D model of the vehicle and pedestrian can indicate out their rough location, outline and directions, which can show us a more visualized sight of the traffic around.

#### 1.1 Problem

Autonomous driving and traffic control require the machine to intelligently recognize the class of the object and the location it lies in. With the popularity of the 3D sensor development on the vehicles, more and more 3D images are recorded and processed. The resources should not be wasted and the condition also needs to be improved and developed. The problem requires us to preprocess the data first and do both the classification and localization procedures

#### 1.2 Resources

The related resources include the datasets from the KITTI and some former designed neural network. Also, we have learned from several benchmarks to improve our project work. From them, we get our idea about how to create our pipeline and make prediction.

### II. Environment

Before we begin our project, we will set up our environment first. The project is running under python 3.6 environment and using the package like the TensorFlow, numpy, opencv, mayavi and so on. The training is running under Tensorflow-gpu 1.4.0, Cuda 8.0 and cudnn 6.0.

### III. Data

The introduction of data mainly includes the resource of dataset, what it is and how to preprocess it.

#### 3.1 Data set

The dataset comes from KITTI. The dataset includes 7481training 2D images and 7518 test images as well as the corresponding point cloud data. It includes a total of 80256 labeled objects in the images. We download the data resource including left color images of object data set (12 GB), training label on these images (5 MB), camera calibration matrices of object data set (16 MB) and velodyne point cloud data (29 GB) to the corresponding 2D left color RGD images. We mainly work on the point cloud data to make the prediction and localization.

#### 3.2 Point Cloud

The velodyne point cloud data provided by KITTI describes the 3D space behind each 2D image, which means that point cloud gives the 2D image depth and help them become visualized in 3D space. Point cloud is a set of unordered points. Points include the 3-dimensional RGB information and the coordinate information. The 3-dimensional RGB and coordinatge information are discussed with regard to the specific original point where the camera sets.


Figure 1. Point Cloud Sample

As we can see in the picture, the 3D-space rabbit object is formed by multiple point cloud data. The information of these point cloud data is provided by the

scanner---velodyne. The sample velodyne point cloud data input is like below:

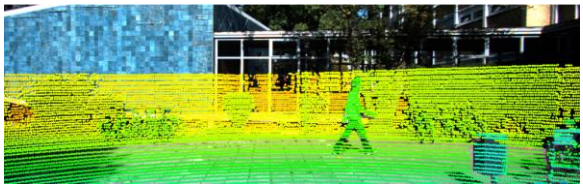


Figure 2. Point Cloud of Pedestrian

Mayavi is package used in python to help us have a detailed and visualized view of these point cloud data sets. The sample point cloud data in mayavi is like:

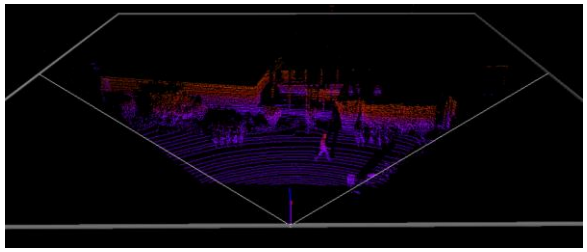


Figure 3. Point Cloud in mayavi

### 3.3 Frustum Proposal

After setting the coordinates of the camera, we can project the 2D proposal region in the 2D image to the 3D point cloud space. In other word, with a known camera projection matrix, a 2D bounding box can be lifted to a frustum that defines a 3D search space for the object. The result of projection cutting by 2D proposal region forms a frustum. We only need to analyze the point cloud data inside this frustum. The forming process is shown as below:

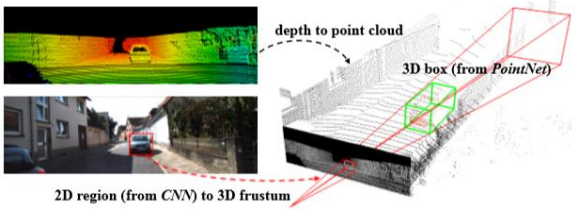



Figure 4. Diagram of forming a frustum

What we need to do is analyzing the point cloud data in this frustum and do the job following this data preprocessing procedure.

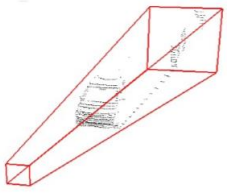


Figure 5. Frustum

## IV. Model Design

In this part, we will first introduce the neural network PointNet which is suitable for the point cloud preprocessing, and then utilizae it and its improvement PointNet++ into our model design. The model design includes 3D instance segmentatino in frustum and 3D bounding box estimation in 3D space. Also, the model has two versions, one using PointNet and another using PointNet++.

### 4.1 PointNet

When we are analyzing the data in the 3D frustum proposal, it is hard to train the model base on these unordered point cloud, since this model can only predict the objects in the specific location where it gets trained. Consider the problem below:

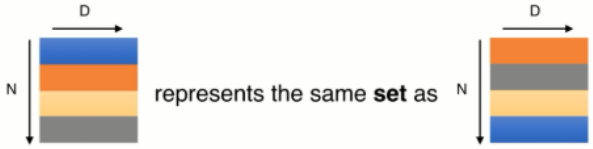



Figure 6. Unordered Point Cloud

One solution to predict these two seeming-different vectors as the same one is doing something symmetric on these vectors like max pooling. However, max pooling could probably cause the necessary information loss in point cloud data set. Thus, we could initially lift the data into high dimensional layers and then do the max pooling operation, which could keep the important information and maintain the symmetric characteristic. PointNet uses MLP (Multilayer Perceptron) lifting the information matrices in former 3 dimensions to higher dimension space.

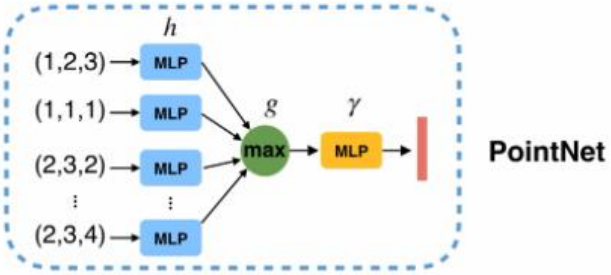



Figure 7. Schematic Diagram of PointNet

### 4.2 PointNet ++

In PointNet, the projection from low data dimension to high dimension is done unifomrly in the frustum. However, when facing the non-uniform density condition, it can be a great challenge for PointNet since the operations at the low-density area can be wasted and the operations at the high-density area is far from enough. Thus the concept of PointNet++ is introduced. The concept of the PointNet++ is that recursively applying PointNet in the high-density regions and extract out their fetures. The methods of PointNet++ include Multi-scale grouping (MSG) and Multi-res grouping (MRG). MSG applies PointNet on the regions with different radius and same circle center. MRG applies PointNet on the regions with different circle centers and same radius.

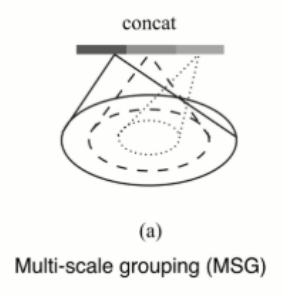



Figure 8. MSG

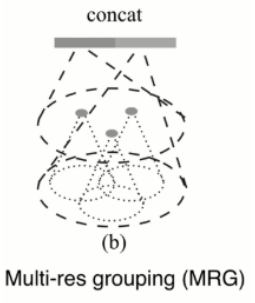



Figure 9. MRG

4.3 3D Instance Segmentation

Because objects are naturally separated in physical space, segmentation in 3D point cloud is much more natural and easier than that in images where pixels from distant objects can be near-by to each other. Based on this fact, we decide to do the instance segmentation in 3D space.

4.3.1 PointNet

The network takes a point cloud in frustum and predicts a probability score for each point that indicates how likely the point belongs to the object of interest. Each frustum contains only one kind of object, thus one point cloud in the frustum can only be or not be that object. Similar as the 2D instance segmentation, our PointNet is learning the clutter patterns as well as recognizing the geometry for the object of a certain label.

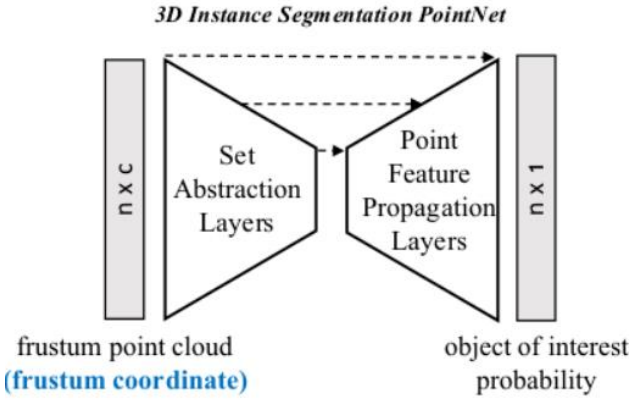



Figure 10. 3D Instance Segmentation PointNet

We also modify some specific parameters in MLP layers in the original PointNet. Here we set the MLP as five convolution layers with kernel szie 1*1. The detailed structure is shown as below:

| | |
|---|---|
| Inputs | N x 4 |
| MLP | (64,64) |
| Conv | N x 64 |
| MLP | (64,128,1024) |
| Conv | N x 1024 |
| Max pooling | Global feature |
| Conv | N x (1088+k) |
| MLP | (512,256,128,128,2) |
| Conv | N x 2 |
| Outputs | N x 1 |

Table 1. 3D Instance Segmentation in PointNet

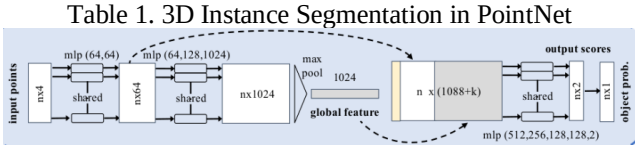



Figure 11. 3D Instance Segmentation in PointNet

The numbers in the brackets mean the amount of neurals in the hidden layer of MLP.

4.3.2 PointNet++

We also implement the PointNet++ in 3D Instance Segmentation by adding some MSG and MRG into the former network.

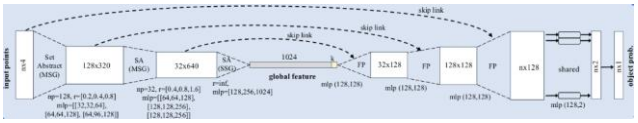



Figure 12. 3D Instance Segmentation in PointNet++

4.4 3D Bounding Box Estimation

Our module estimates the object's amodal oriented 3D bounding box by using a box regression PointNet(++) together with a preprocessing transformer network T-Net.

4.4.1 T-Net

We use T-Net to transfer the center of segmentation region to the estimation of the center of 3D bounding box. The abstract architecture is shown as below:

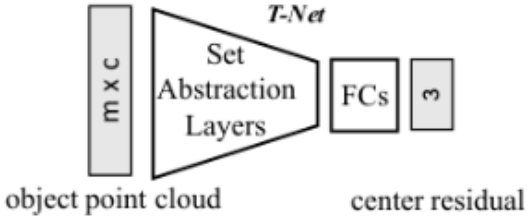



Figure 13. T-Net Architecture

This T-Net is light-weight PointNet which could help transform the coordinate of predicted center by regression.

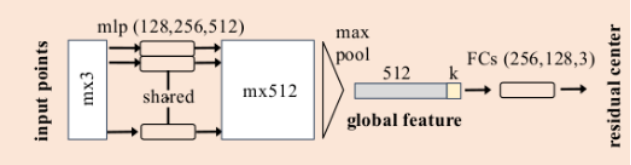



Figure 14. T-Net Implementation

4.4.2 3D Box Estimation by PointNet

For PointNet version, the network is designed as:

| | |
|---|---|
| Inputs | M x 3 |
| MLP | (128,128,256,512) |
| Conv | M x 512 |
| Max pooling | Global feature (512 + k) |
| FCs | |
| Outputs | Box parameters |

Table 2. 3D Box Estimation by PointNet

4.4.3 3D Box Estimation by PointNet++

For the PointNet++ version, we add some SSG into the former architecture.

| | |
|---|---|
| Inputs | M x 4 |
| SSG(MLP) | (64,64,128) |
| Conv | 128 x 320 |
| SSG(MLP) | (128,128,256) |
| Conv | 32 x 640 |
| SSG(MLP) | (256,256,512) |
| Max pooling | Global feature (512 + k) |
| FCs | |
| Outputs | Box parameters |

Table 3. 3D Box Estimation by PointNet++

The PointNet++ version doesn't need the T-Net since the local features extraction in PointNet++ is not so sensitive to rotating the camera visual angle and thus no need to transfer the center position.

## V. Training & Results

5.1 Training Environment and Hardware

We train both the PointNet version and PointNet++ versions on the cloud computing platform hpc.ai-research provided by Professor Jiajia Luo. The environment and training parameters are shown as below:

| | V1 PointNet | V2 PointNet++ |
|---|---|---|
| GPU | NVIDIA GTX 1080ti | NVIDIA GTX 1080ti |
| Batch Size | 32 | 24 |
| Decay Step | 800000 | 800000 |
| Decay Rate | 0.5 | 0.5 |
| Epoch | 200 | 200 |

Table 4. Training parameters

Our model v1 runs for 1 day and our model v2 runs for about 3 days. Then we make prediction mainly on three classes: car, pedestrian and cyclist. The difficulty of the test is also be divided into three levels: easy, moderate and hard. In the easy level, there is only one single object in the image. In the moderate level, there can be some multiple objects occurring in the image. In the hard level, there can be more different kinds of objects showing in the image and they can probably be overlapped with each other.

The results come out like below:

For v1:

```
car_detection_3d AP: 83.886032 69.448318 62.808960

pedestrian_detection_3d AP: 68.117783 58.301006 51.048351

cyclist_detection_3d AP: 72.054031 53.010910 49.345177
```

| | Easy % | Moderate % | Hard % |
|---|---|---|---|
| car | 83.89 | 69.45 | 62.81 |
| pedestrian | 68.12 | 58.30 | 51.05 |
| cyclist | 72.05 | 53.01 | 49.35 |

Table 4. result of version v1

For v2:

```
save train/detection_results_v2/plot/car_detection_3d.txt
car_detection_3d AP: 83.670593 70.894257 63.638203

save train/detection_results_v2/plot/pedestrian_detection_3d.txt
pedestrian_detection_3d AP: 66.527534 58.461678 51.382637

save train/detection_results_v2/plot/cyclist_detection_3d.txt
cyclist_detection_3d AP: 77.974770 56.860680 53.606236
```

| | Easy % | Moderate % | Hard % |
|---|---|---|---|
| car | 83.67 | 70.89 | 63.64 |
| pedestrian | 66.53 | 58.46 | 51.38 |
| cyclist | 77.97 | 56.86 | 53.61 |

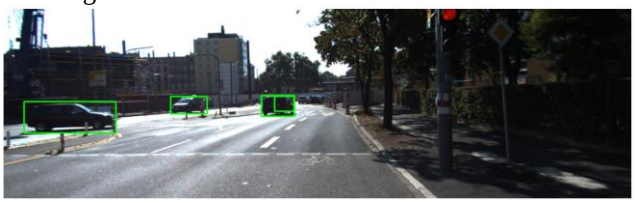
Table 5. result of version v2

From these results, we can see that version PointNet++ performs better than version PoinNet, especially when the difficulty level increases. The recursively apply of PointNet at specific region proves to be a better solution for the non-uniformly region regression and multiple object detection.

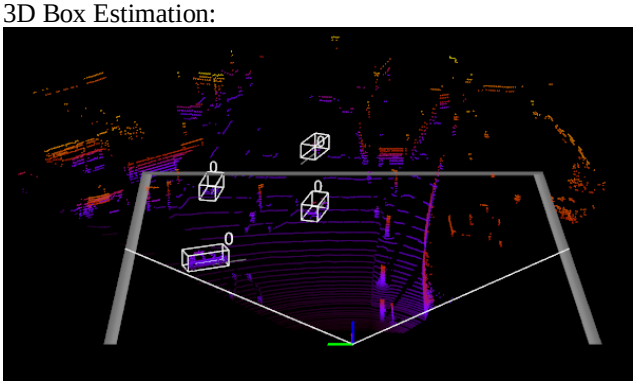
Also, we use mayavi module to help us get a more visualized view of the 3D box estimation of the detected objects.

3D Box Estimation Result in mayavi:

2D image label:

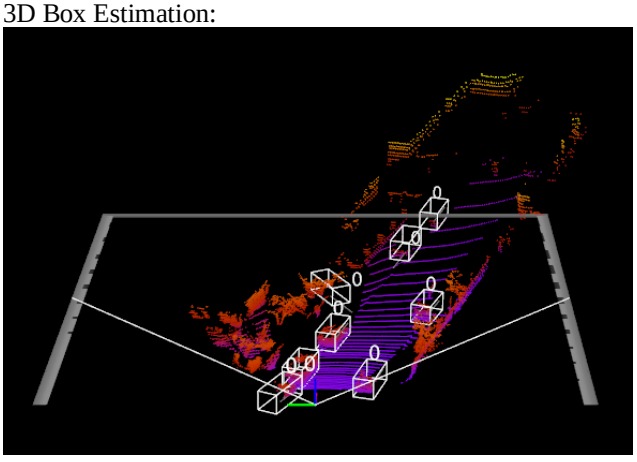


3D Box Estimation:

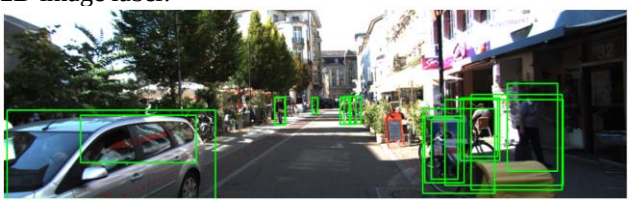


2D image label:

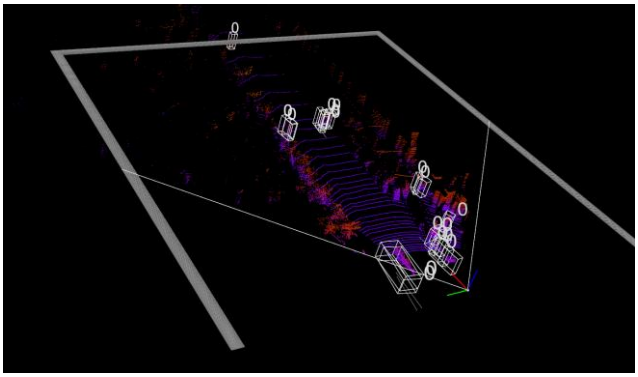
3D Box Estimation:



2D image label:

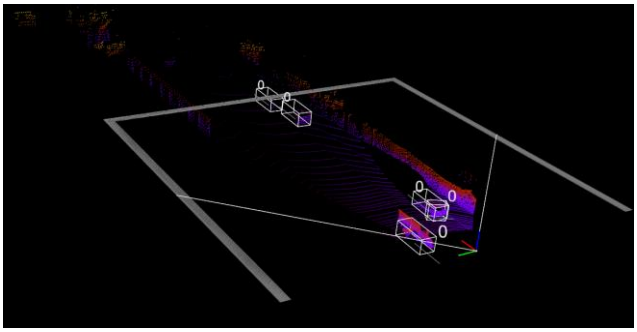


3D Box Estimation:



2D image label:

3D Box Estimation:



## VI. Comparison and Discussion

6.1 Alternative State-of-the-art Method Frustum ConvNet

After releavant litreture research and referring to KITTI evalution record we found a modified version of Frustum PointNet called Frustum ConvNet (F-ConvNet), which has better performance than original Frustum PointNet. Then we tried to analyze the differences between F-PointNet and F-ConvNet and what made F-ConvNet's performance looked like better on KITTI benchmark.

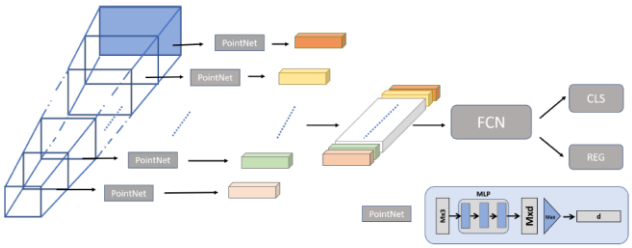



Figure 15. The whole framework of F-ConvNet

F-ConvNet pipeline is based on F-PointNet that defines a 3D frustum through the 2D proposal and then detects 3D instance box in the frustum. However, the author of F-ConvNet indicates that local grouping of point cloud is a challenge. And they think that F-PointNet uses FPS to group local points which ensures coverage of point cloud, but FPS is not useful for object detection, which is a limitation of F-PointNet. Nevertheless, we think this criticism is not inaccurate and will discuss about it in the later part. To overcome the limitation, they propose Frustum Convolution Net. The pipeline will first divide the whole frustum from projection of 2D proposal to a sequence of small frustums with an equal stride. And then they use PointNet to extract frustum-level feature for each frustum, which will be processed and concatenated in the following FCN network. Finally, they use detection header as the classification and regression branches to deal with FCN outputs.

6.2 Contrast Analysis of F-PointNet & F-ConvNet

To understand crucial differences between two pipelines and what make sense we should contrast them part by part.

6.2.1 Key Characteristics of F-PointNet

As mentioned above, F-PointNet in fact has two networks respectively undertaking segmentation and estimation. Indeed, this pipeline uses FPS in PointNet++ to segment point cloud and FPS can efficiently cover masked point cloud we interest. And FPS doesn't undertake any position or detection in this part. Besides, PointNet++ with MSG and MRG can effectively mitigate nonuniformity of point cloud data due to LIDAR probe. After the segmentation, another PointNet designed to position and detect objects will finish the estimation task. In conclusion, F-PointNet properly uses FPS to sample point cloud as far as possible, and meanwhile this design doesn't weaken the detection ability of the whole pipeline due to these two separated networks.


Figure 16. Deconstructed Diagram of F-PointNet

6.2.2 Key Characteristics of F-ConvNet

With regard to F-ConvNet, this pipeline has two main parts: frustum-level feature extraction and concatenation. Because it will divide the frustum with equal stride it doesn't need to consider how to group local points and in which scale. And partitioned frustum-level feature extraction can partly mitigate nonuniformity of point cloud, but fixed stride without density-adapting and multi-stage learning limits this ability. In feature concatenation part, the core idea is abstracting frustums to 2D feature maps and then learning features via FCN like 2D image. And feature concatenation from partitioned frustums also provides a hierarchical granularity. However, this kind of abstraction may lose some information. In conclusion, F-ConvNet is simple and easy to adjust but may not perform great on nonuniform point cloud data.

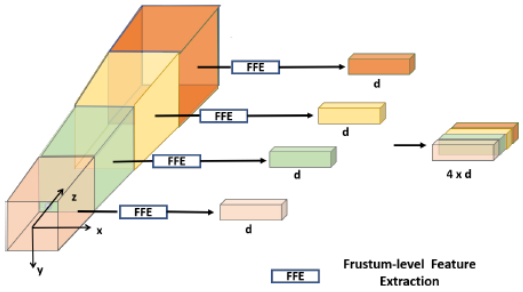



Figure 17. Deconstructed Diagram of F-ConvNet

6.3 Results Comparison and Discussion of Pros & Cons

According to the comparison, we can obviously find F-ConvNet doesn't overcome the shortcoming of FPS in F-PointNet they assert, actually we don't think F-PointNet has such shortcoming. Furthermore, F-ConvNet has even weaker generalization ability on extremely nonuniform point cloud data. But, why F-ConvNet has better performance like higher accuracy than F-PointNet on KITTI datasets? We think it because F-ConvNet has simpler hyperparameters while PointNet++ in F-PointNet has more complicate mechanism about MSG and MRG to generically deal with nonuniformity problem. And KITTI datasets don't have a lot of extreme data so that F-ConvNet can easier to adjust hyperparameters to get greater accuracy on KITTI. In fact, after adjusting hyperparameters of F-PointNet we get a result as well as F-ConvNet especially on small object Pedestrians, which is shown in the table below (E=Easy, M=Moderate, H=Hard). Hence, we still choose F-PointNet as our backbone and improve it.

| Method | Cars | | | Pedestrians | | | Cyclists | | |
|---|---|---|---|---|---|---|---|---|---|
| | E | M | H | E | M | H | E | M | H |
| Original F-PointNet [1] | 81.20 | 70.29 | 62.19 | 51.21 | 44.89 | 40.23 | 71.96 | 56.77 | 50.39 |
| F-ConvNet [2] | **85.88** | **76.51** | **68.08** | 52.37 | 45.61 | 41.49 | **79.58** | **64.68** | **57.03** |
| Ours Improved F-PointNet | 83.67 | 70.89 | 63.64 | **66.58** | **58.46** | **51.38** | 77.97 | 56.86 | 53.61 |

Table 6: 3D object detection AP (%) on KITTI test set

## VII. Challenge and Future Effort

The critical challenge is that when two same category objects are totally overlapped on 2D image (like two persons standing by), we will only get on proposal and then there are two similar instances in a frustum. But current pipeline only assumes a single object of interest in each frustum, which is more like semantic segmentation instead of instance segmentation. To solve the problem, we should improve precision of 3D segmentation PointNet and then apply advanced instance segmentation thoughts like TensorMask and Mask R-CNN on 3D box estimation pointNet. Hence, we are still trying to get better performance on our improved F-PointNet.

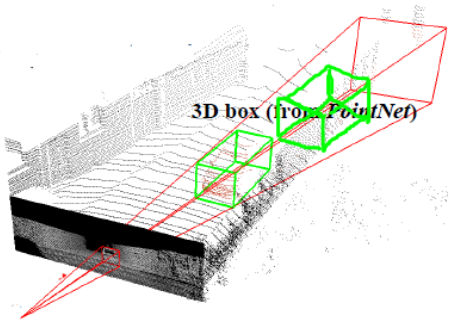



Figure 18. Schematic Diagram of Overlapped Problem

## Reference

[1] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018

[2] Z. Wang, K. Jia, "Frustum ConvNet: Sliding Frustums to Aggregate Local Point-Wise Features for Amodal 3D Object Detection," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2019.