EE599 Deep Learning – Final Report

3D object detection system for self-driving vehicle.

Jingbang Zhong, Weizhong Jin, Xinyi Tao.

May 7, 2019

1. Abstract

Currently 3D target detection is in a period of rapid development. For self-driving vehicle, 3D object detection can help to indicate the distance between other object (cars, people etc.) and our car, and give us a warning signal if there is any possibility of collision. In this report, we propose to realize 3D object detection using 2 different models, an excellent example model from KITTI and another self-created model. Through the input images and ground truth object label with corresponding bounding box, we train a Convolutional Neural Network to localize objects in 2D image, and categorize it with corresponding class, such as pedestrian and vehicle. Firstly, we started the project from learning an excellent example form KITTI, called Point-RCNN. We trained this model with our dataset and finally get a good performance. And then, we try to develop our own model, and it is our novelty point too. The new model is a combination of 2D Faster-RCNN and 2D-3D transformed model. By applying the transformed model, we can get 3D bounding box of object from 2D bounding box generated by Faster-RCNN model. We can also get distance between object and car by calculating 3D information.

2. Introduction

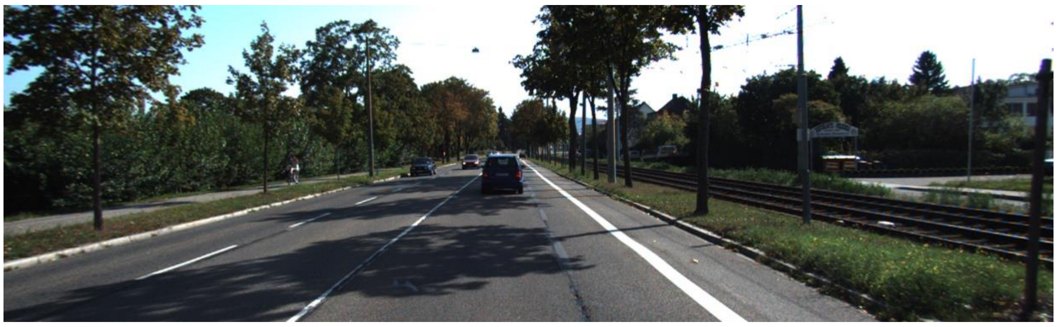
2.1 Problem Statement and Goals

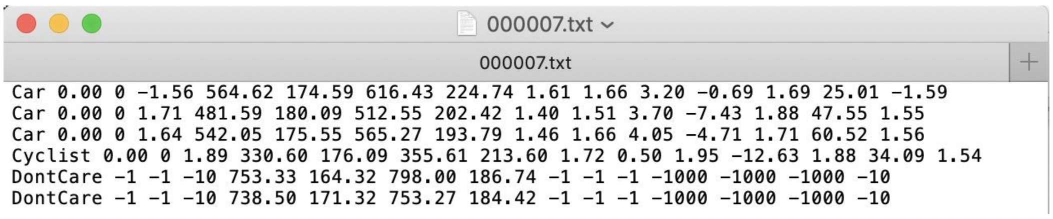
Camera and LIDAR are two types of sensors that are often used by autonomous vehicles to sense their surroundings. RGB images are generally used for target detection. The output object class and the minimum bounding box on the image are called 2D target detection. The output information such as object type, the length, width, height, and rotation angle in three-dimensional space by training the RGB image, the RGB-D depth image, and the laser point cloud is called 3D object detection. With the advent of Faster-RCNN, 2D target detection has reached unprecedented prosperity, and new methods are emerging. However, in the application scenarios of driverless, robotic, and augmented reality, ordinary 2D detection does not provide all the information needed to perceive the environment. 2D detection can only provide the position of the target object in the two-dimensional image and the confidence of the corresponding category. However, in the real three-dimensional world, objects have three-dimensional shapes, and most applications require information such as the length, width, height, and deflection angle of the target object. Therefore, it is easier to get a 2D bounding box of the object on the image than 3D bounding box. At present, 3D target detection is in a period of rapid development. And monocular cameras, binocular cameras, and multi-line laser radars are mainly used for 3D target detection. Monocular camera detection is of lowest cost, and lowest accuracy. However, in this project we developed a deep learning model by combining Faster-RCNN and 2D-3D transformed model. This new model is of low cost since only 2D image captured by monocular camera is required, and higher accuracy.

What we are doing is to construct a 3D object detection system for driverless vehicle to detect dynamic objects. The first step is to use our model to draw a 3D bounding box of the object needed to be recognized, such as car and people. The second step is to classify different objects with corresponding labels. The third step is to calculate the distance from detected object to our car. As we already get the 3D bounding box of detected object, we can get the center point coordinate of this object and calculate the distance. Furthermore, we can set an alert that if the distance is smaller than the safe distance, the system will give us a warning signal to avoid collision.

2.2 Data collection

We collected 7481 images generated by driving recorder as our raw data, including day and night view. The images also come with corresponding bounding box and labels. What we have to do is to evaluate the data and their classification label to see if this dataset and its size meet the requirement. And here is an example of input image and its ground truth label.





```
Car 0.00 0 -1.56 564.62 174.59 616.43 224.74 1.61 1.66 3.20 -0.69 1.69 25.01 -1.59
Car 0.00 0 1.71 481.59 180.09 512.55 202.42 1.40 1.51 3.70 -7.43 1.88 47.55 1.55
Car 0.00 0 1.64 542.05 175.55 565.27 193.79 1.46 1.66 4.05 -4.71 1.71 60.52 1.56
Cyclist 0.00 0 1.89 330.60 176.09 355.61 213.60 1.72 0.50 1.95 -12.63 1.88 34.09 1.54
DontCare -1 -1 -10 753.33 164.32 798.00 186.74 -1 -1 -1 -1000 -1000 -1000 -10
DontCare -1 -1 -10 738.50 171.32 753.27 184.42 -1 -1 -1 -1000 -1000 -1000 -10
```

Pic 2.1 data: image and label

The data package we used includes RGB images with label, velodyne point cloud and camera calibration matrices downloaded from the website The KITTI Benchmark Suit, under the 3D object column.[1]

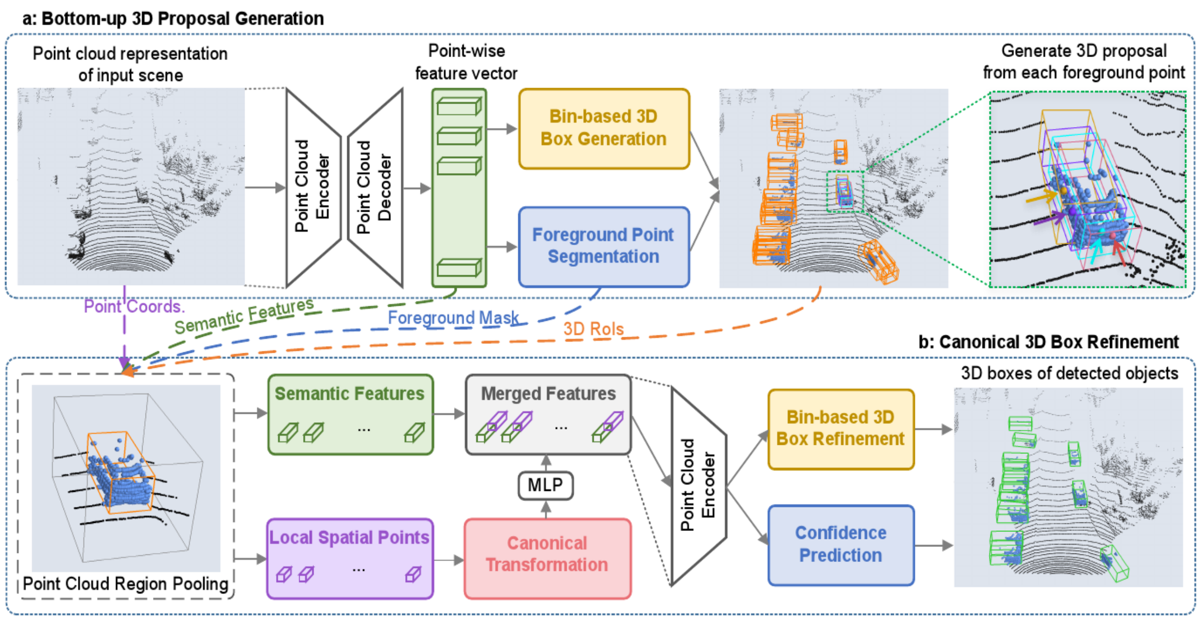
We don’t only use original label of KITTI. Because the Faster-Rcnn frame requires data in Pascal VOC form, we have to transfer the original label to Pascal VOC form which only contain 2D bounding Box information.

3. Starting point

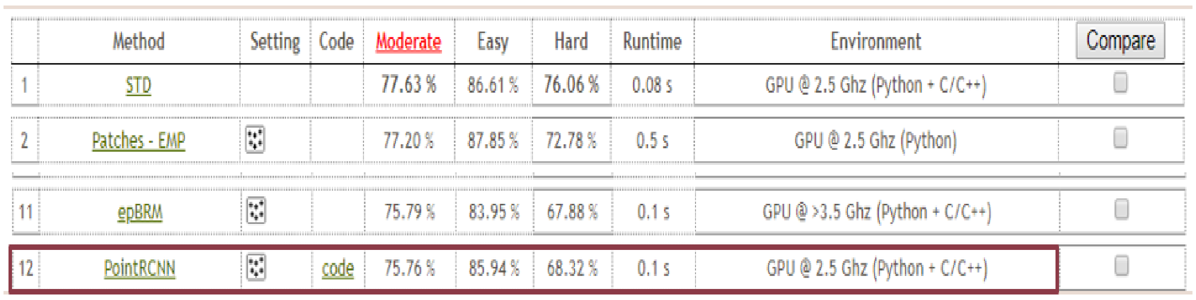
3.1 Model Introduction

At the first beginning, all of us have no clue about what 3d object detection is, so we started the project from learning an excellent example. KITTI has the most famous dataset for 3D object detection. Therefore, we chose a model from KITTI as our starting point. By learning the structure of this model, as shown in pic3.1, we hope we can get some inspiration for the later self-created model. The model we chose is called PointRCNN. The pic3.2 is a ranking of performance on the KITTI 3D object detection leaderboard. The PointRCNN ranked 12th, which shows its good performance and high detection accuracy.
The whole framework is composed of two stages. Stage 1 is the bottom-up 3D proposal generation. In this stage, instead of using RGB images as input to generate 3D bounding box which is the same as previous methods, we directly use point could as the input to generates a small number of high-quality 3D proposals, in a bottom-up manner. Specifically, it means that it segments the point cloud of the whole scene into foreground points and background points. All points inside the 3D ground-truth boxes are considered as foreground points. The other points are treated as background points. During training, we ignore background points near the object boundary, so that we can reduce the computation complexity. The stage 2 is for refining the 3D proposals from stage 1 to obtain the final bounding box locations and orientations. It transforms the pooled points of each proposal to canonical coordinates to learn better local spatial features. It is also combined with global semantic features of each point learned in stage 1 for accurate 3D bounding box refinement and confidence prediction. [2]

Platform:
Pytorch, AWS 3.2xlarge instance, OpenCV


a: Bottom-up 3D Proposal Generation
Point cloud representation of input scene
Point Cloud Encoder
Point Cloud Decoder
Point-wise feature vector
Bin-based 3D Box Generation
Foreground Point Segmentation
Generate 3D proposal from each foreground point
Point Coords.
Semantic Features
Foreground Mask
3D RoIs
b: Canonical 3D Box Refinement
Point Cloud Region Pooling
Semantic Features
Merged Features
MLP
Local Spatial Points
Canonical Transformation
Point Cloud Encoder
Bin-based 3D Box Refinement
Confidence Prediction
3D boxes of detected objects

Pic 3.1 Point-RCNN



	Method	Setting	Code	Moderate	Easy	Hard	Runtime	Environment	Compare
1	STD			77.63 %	86.61 %	76.06 %	0.08 s	GPU @ 2.5 Ghz (Python + C/C++)	☐
2	Patches - EMP	▦		77.20 %	87.85 %	72.78 %	0.5 s	GPU @ 2.5 Ghz (Python)	☐
11	epBRM	▦		75.79 %	83.95 %	67.88 %	0.1 s	GPU @ >3.5 Ghz (Python + C/C++)	☐
12	PointRCNN	▦	code	75.76 %	85.94 %	68.32 %	0.1 s	GPU @ 2.5 Ghz (Python + C/C++)	☐

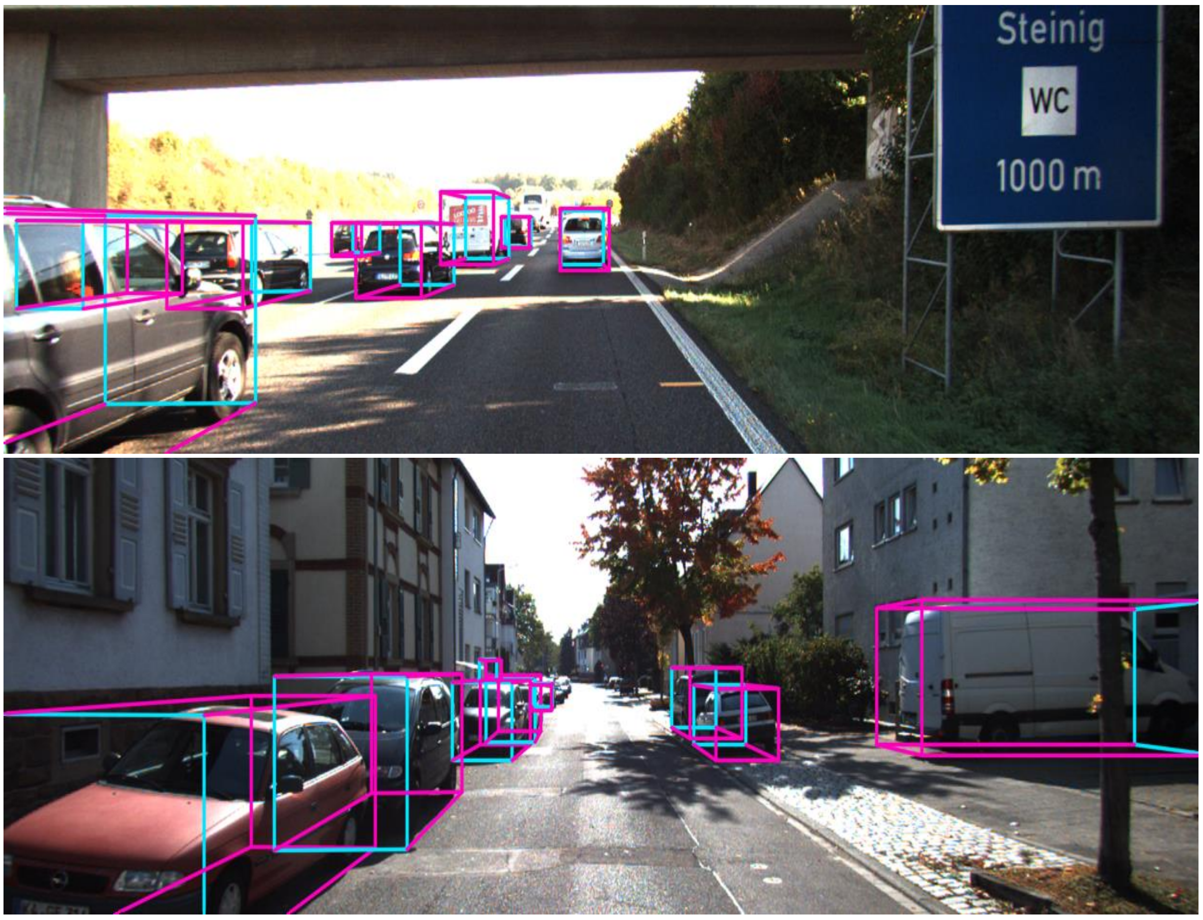
Pic 3.2 accuracy of Point-RCNN

3.2 *Training analysis*

In the training step, the two stages of PointRCNN are trained separately. For each 3D point-cloud scene in the training set, we subsample 16,384 points from each scene as the inputs. The stage-1 sub-network is trained for 200 epochs with batch size 16 and learning rate 0.002. It took around 15 hours. After we have a well-trained first stage model, then we started to train the second stage. The stage-2 subnetwork is trained for 50 epochs with batch size 256 and learning rate 0.002. And it took around 20 hours. During training, we randomly augment the 3D proposals with small variations to increase the diversity of proposals.

3.3 *Result*

The 3D object detection result we got are shown as follow. We randomly select a few images from the database, and input the images to this model. As we can see, it correctly detects the location of every vehicle in this image, and give us their 3D location. The experiments show that PointRCNN outperforms previous state-of-the-art methods with remarkable margins on the challenging 3D detection benchmark of KITTI dataset.



Pic 3.3 results of Point-RCNN

4. Self-created model

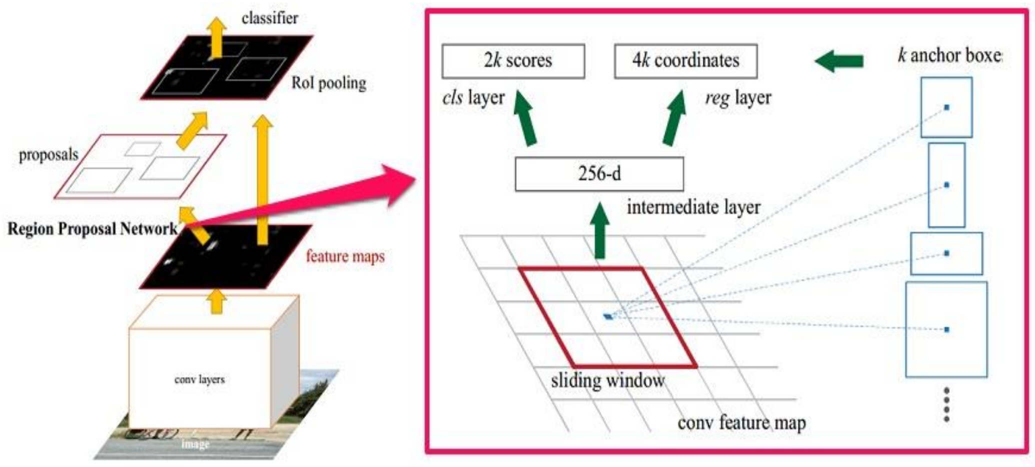
4.1 *Model Introduction*

The model Faster-3D-RCNN is combination of Faster-RCNN (traditional object detection deep learning network) and a 2D-3D network.

4.1.1 Faster-RCNN

Faster R-CNN has two networks: region proposal network (RPN) for generating region proposals and a network using these proposals to detect objects as showed in pic 4.1 . Faster-RCNN will first extract image's feature map by a CNN. Based on this CNN, the RPN will use anchor technique to propose a lots of region in Proposal layer and Anchor Target layer. Then, by computing RPN Loss, the proposal target layer will give out a bunch of proposal region that likely contain objects. Then here will be a ROI pooling layer, this layer will crop original feature map according to proposal regions. Finally, the cropped feature map will send to final classifier (FCN) to classify and compute the Classification Loss. Faster-RCNN have really good performance on detection accuracy and faster than Fast-RCNN and RCNN.

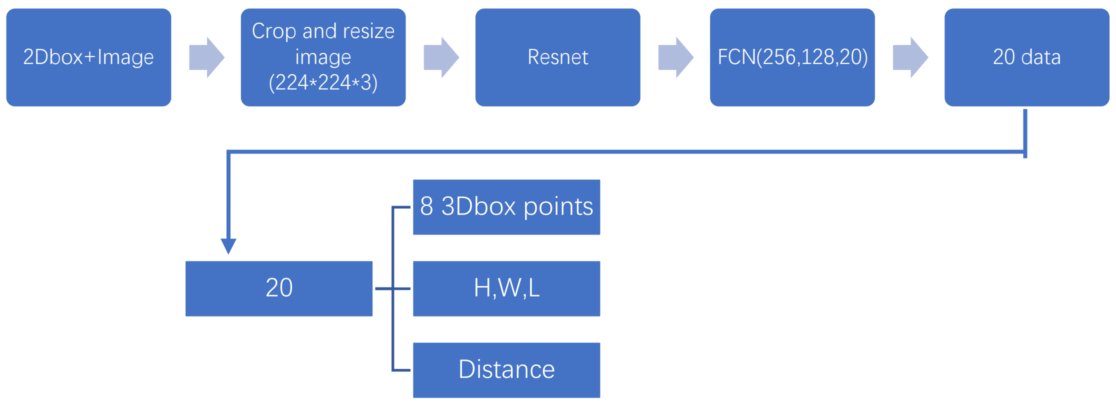
But it's still slower than YOLO or SSD, such one stage network. There will be five outputs of Faster-RCNN. The first four will be left, top, right, bottom pixel coordinate of bounding box. The last one will be the score of object.[3]



Pic 4.1 Faster-RCNN model

4.1.2 2D-3D model

We will use 2D bounding box to crop the original image and resize it to 224*224*3 image. This will be the input of our 2D-3D model. The first part of this model will be ResNet34, the part will extract the feature of cropped 2D bounding box image. The dimension of output of Resnet34 will be 512, the here will be a FCN with RELU activation layer. The final layer will have no activation and the output of final layer will be a vector of 20 elements. The first 16 will be pixel coordinates of the 8 vertices on the box. The next 3 is height, width and length of box. The last one is the distance between camera and object.[4]



Pic 4.2 2D-3D model

Platform:

Pytorch, AWS 3.2xlarge instance, OpenCV

4.2 Model Analysis

The Faster-3D-RCNN is really a new experiment. We combined two model to implement 3D object Detection without point cloud information. The model is pretty large because of two complete networks so that this system cannot achieve real time detection. But this model also has its special advantages:

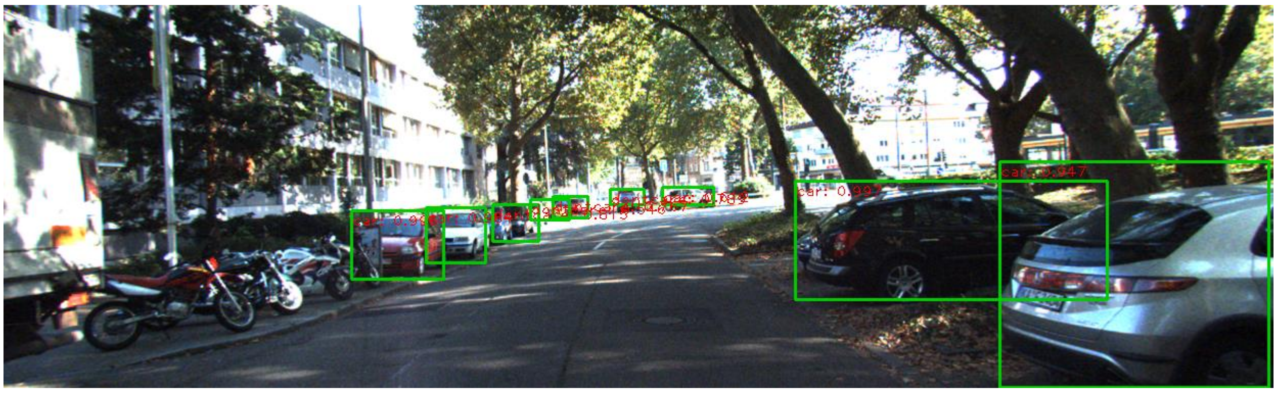
1. The accuracy of detection is really good, because we use only 2D object detection to finish the work of detection, which is easier to implement than 3D object detection. For car detection, most of existing training data only come with 2D bounding box. With these data, we can get a really good model of 2D object detection. The 2D to 3D transformation's performance is better than we have expected, because of all method using KITTI's Data. And we can improve the 3D detection performance in two stage, which means we can use much more data to train the model than method like Point-RCNN.
2. We use only one camera to finish 3D detection and Distance estimation. Now, most 3D detection are based on point cloud but not all car can be equipped with the machines to collect point cloud data. What's more, the most distance estimation method are based on double-camera because we can only build 3D coordinate system with double cameras. But in this project, we try to use only one camera to estimate the distance. This application I think will be a further goal for us to pursue.

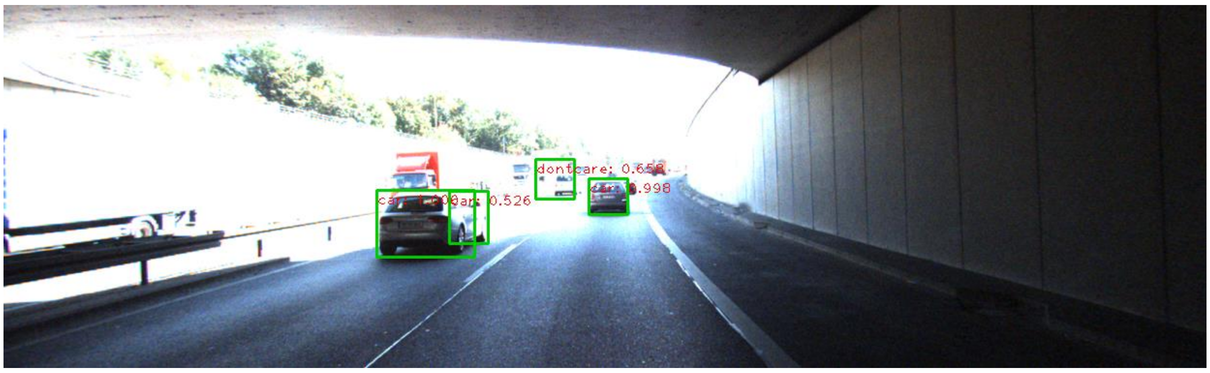
4.3 Training analysis

We trained faster-rcnn (vgg16) model in p3.2xlarge instance for 15 hours and the final loss of model after 20 epoch is 0.02 and accuracy of KITTI test data is 0.80. The 2D-3D model took us 7 hours to train but we didn't test it, just put it into real world application, because there is no other models can be used to compare.

4.4 Result

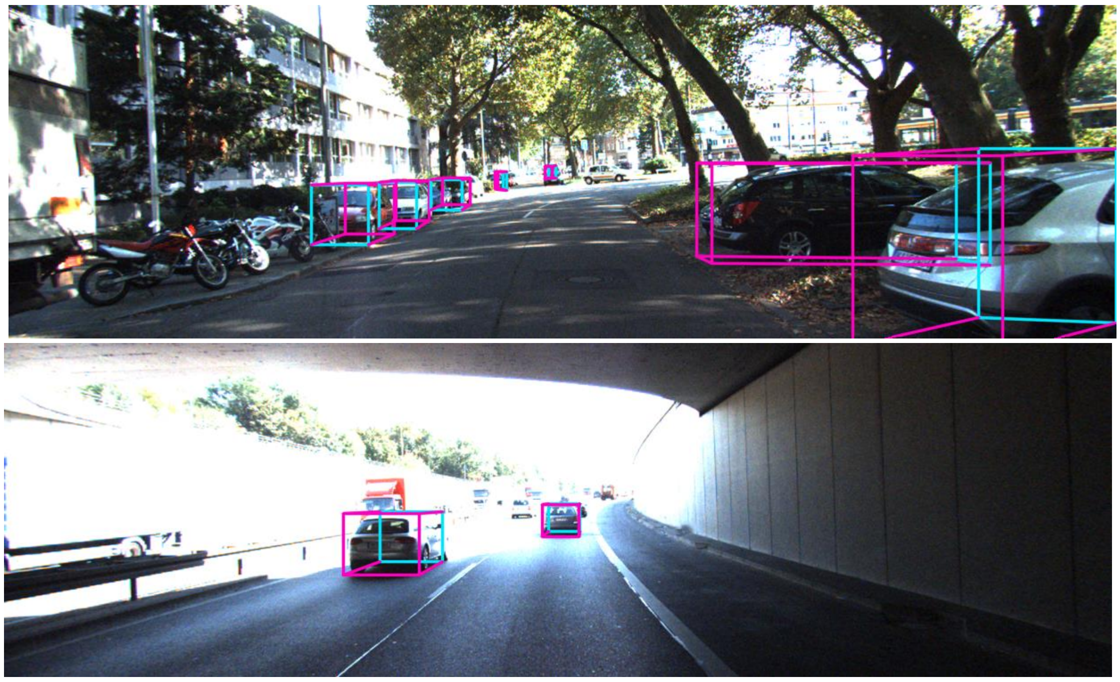
Faster-RCNN result:





Pic 4.3 results of faster-rcnn

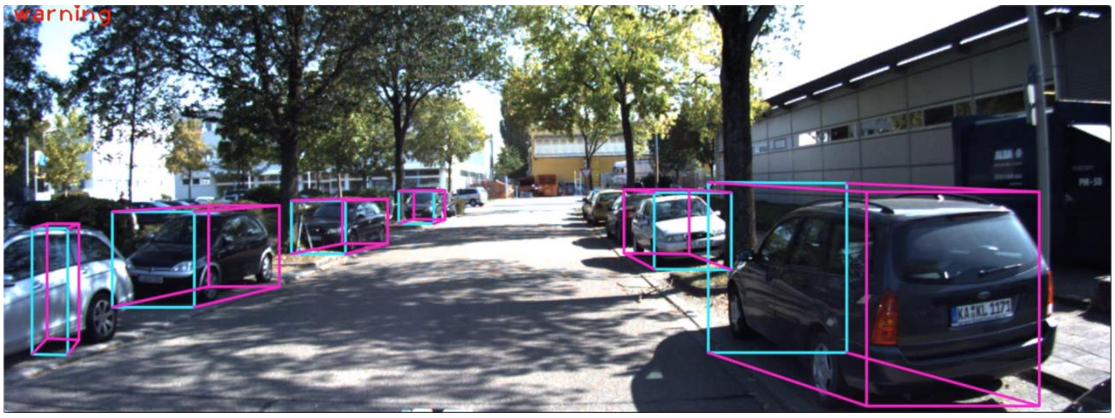
Faster-3D-RCNN result:



Pic 4.4 results of faster-3D-rcnn

As these output pictures, we can find Faster-RCNN correctly detect and classify the objects and our model successfully transfer 2D bounding box to 3D bounding box.

Out model can also give the distance information that we can utilize to give the warning signal when distance between car and object is too short. And the final model we use will spend 0.2s on each input image.



Pic 4.5 final-result

5. Conclusion

We have finished two methods for 3D object detection and successfully utilize it on driverless warning system (though the method is simple). The Point-RCNN works really well due to the point cloud data which can provide the detailed distance information. The Faster-3D-Rcnn as our final novelty method, really worked as we expected. Although this Faster-3D-RCNN cannot satisfy real time working, it's really a good method that has big space to improve:

1. Using More Data

   We only use KITTI this time, and here is still a lot of driverless car dataset, such as Berkeley Deep Drive, CitySpace, Oxford RobotCar, scr.

2. The model size can be much smaller! Here are several solutions:

   a) Use YOLO or SSD instead of Faster-RCNN:

   Faster-RCNN is really big network when compared with YOLO of SSD. Next time we can improve our network by combine 2D-3D model with other 2D object detection network.

   b) Get out the ResNet34 in 2D-3D model:

   We use 2D-3D model to extract the feature map of cropped image. But such feature map has once extracted by the first part of Faster-RCNN. We can just utilize this existed feature map and 2D bounding box (Proposal region) from RPN directly to decrease the processing time.

3. Improve the 2D-3D model's performance:

   This model is still underdeveloped. We need more data to train this model and we can also fine tune its structure.

4. Working on Monocular distance measurement model:

   Because of works and researches of this project we did, we first realized that using deep learning to implement Monocular distance measurement model is really a meaningful topic. We will try to do further research on this direction.

5. Compare with other 3D object detection network.

   We will make a comparison between our novel network and other 3D object detection network, the KITTI website posts a lot of open source method, we can implement them and make a detailed comparison to find the essence of 3D detection.

Reference

[1] Andreas Geiger and Philip Lenz and Raquel Urtasun "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite", Conference on Computer Vision and Pattern Recognition
[2] Shi, Shaoshuai et al. "PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud." CoRR abs/1812.04244 (2018)
[3] Ren, S., He, K., Girshick, R.B., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39, 1137-1149.
[4] http://www.telesens.co/2018/03/11/object-detection-and-classification-using-r-cnns/ "Object Detection and Classification using R-CNNs".