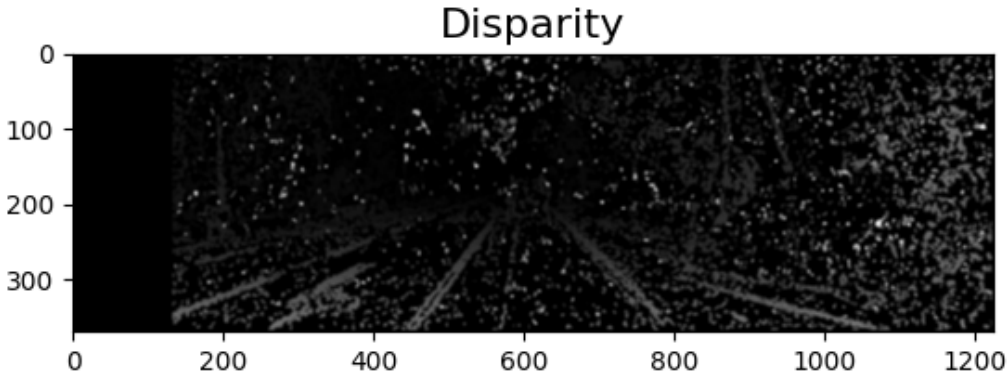
CSC 420 Project2 - Autonomous Driving

Introduction

In recent years, with the rapid development of artificial intelligence technology, the traditional automobile has combined with information technology, and research on automatic driving technology has made a lot of progress. In the existing automatic driving based on traditional features, target recognition is the core task. One, which includes road and road edge recognition, lane line detection, vehicle recognition, vehicle type recognition, non-motor vehicle recognition, pedestrian recognition, traffic sign recognition, obstacle recognition and avoidance, etc. The target recognition system uses computer vision to observe the traffic environment and automatically identify targets from real-time video signals, providing a basis for real-time autonomous driving operations such as starting, stopping, turning, accelerating and decelerating. In this project, we mainly did tasks related to vehicle recognition and road recognition.

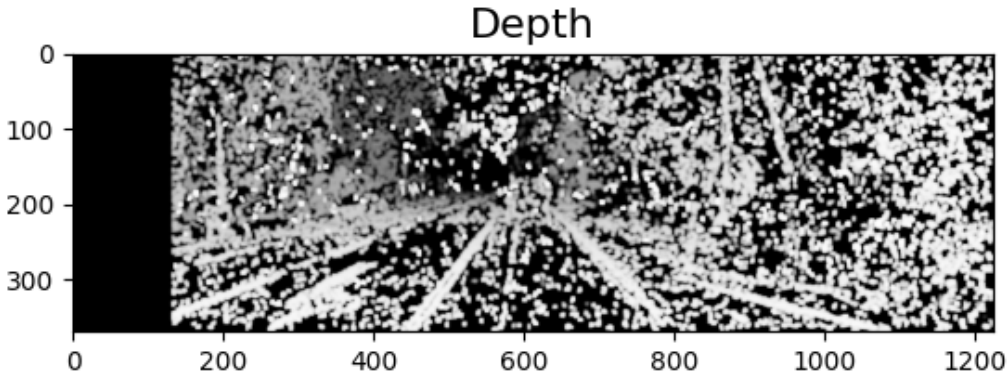
Task 1 Disparity Map

In task1 and task2, we compute the disparity map and depth map from the left view and right view. For disparity map we used the **block matching algorithm** to find the pixel correspondence in the left and right views, and then use the abscissa difference of the corresponding pixels to get disparity map. We used the API `cv2.StereoBM_create()` help us to implement this.



Task 2 Depth Map

From disparity map, we can easily get depth map by formulate: depth = (baseline * focal_length) / disparity

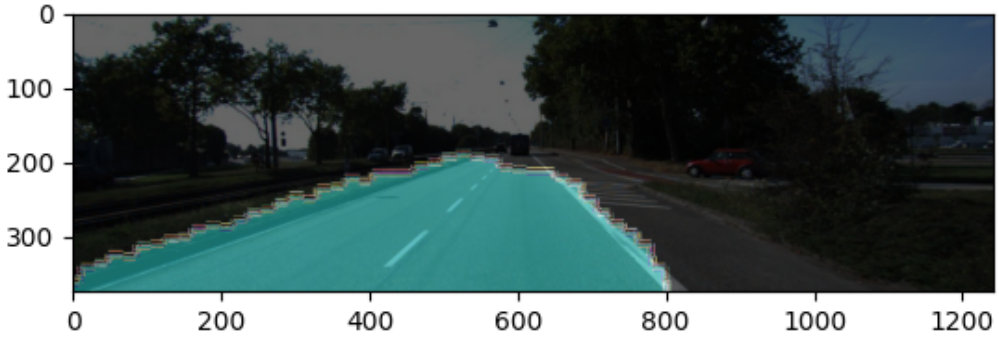


Task 3 Road Classifier

In this task, we user the architecture from [1] Road Segmentation Using CNN and Distributed LSTM. This model has very good performance on KITTI road segmentation challenge:

Method	F1	AP	PRE	REC	FPR	FNR	Para	FLOPs	Runtime
ours	89.08%	91.60%	88.12%	90.06%	6.69%	9.94%	0.35M	6.9B	16 ms
ours without LSTM	81.84%	73.27%	81.66%	82.02%	10.15%	17.98%	0.36M	7.0B	16 ms
MAP [11]	87.80 %	89.96%	86.01%	89.66%	8.04%	10.34%	457.43M	7.15B	280ms
StixelNet [9]	89.12%	81.23%	85.80%	92.71%	8.45%	7.29%	6.82M	43.0B	1s
Up-Conv-Poly [10]	93.83%	90.47%	94.00%	93.67%	3.29%	6.33%	19.44M	31.5B	83ms

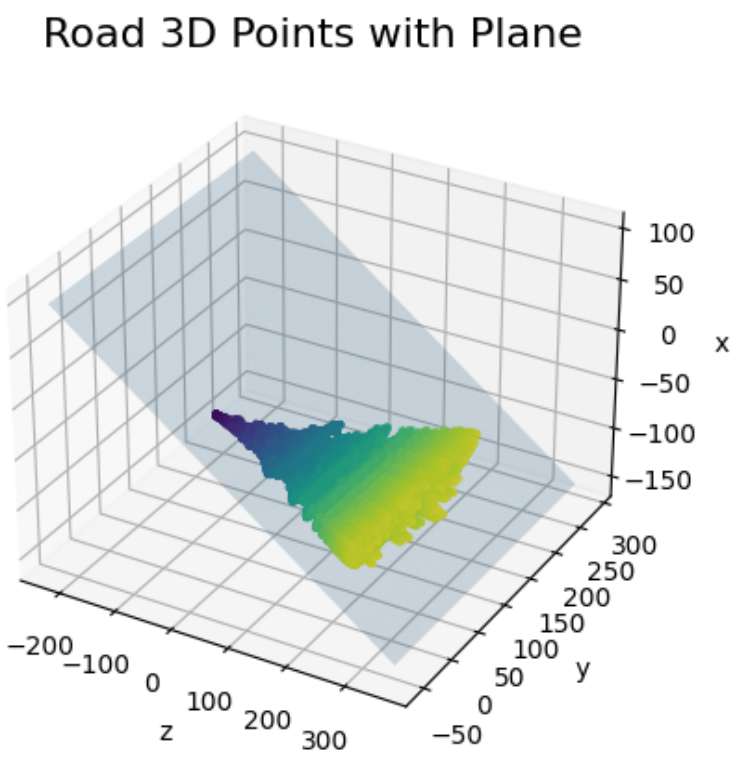
For the feature selection, we used the 2D feature, ground truth pixel of road as the training feature. Train this model is quiet challenging cause it will take a long time by using CPU. To solve this problem, we install CUDA and pytorch for GPU version to train this model. The following images is the ground trueth segmentation of road, and the output of the model .



Task 4 Road Plane

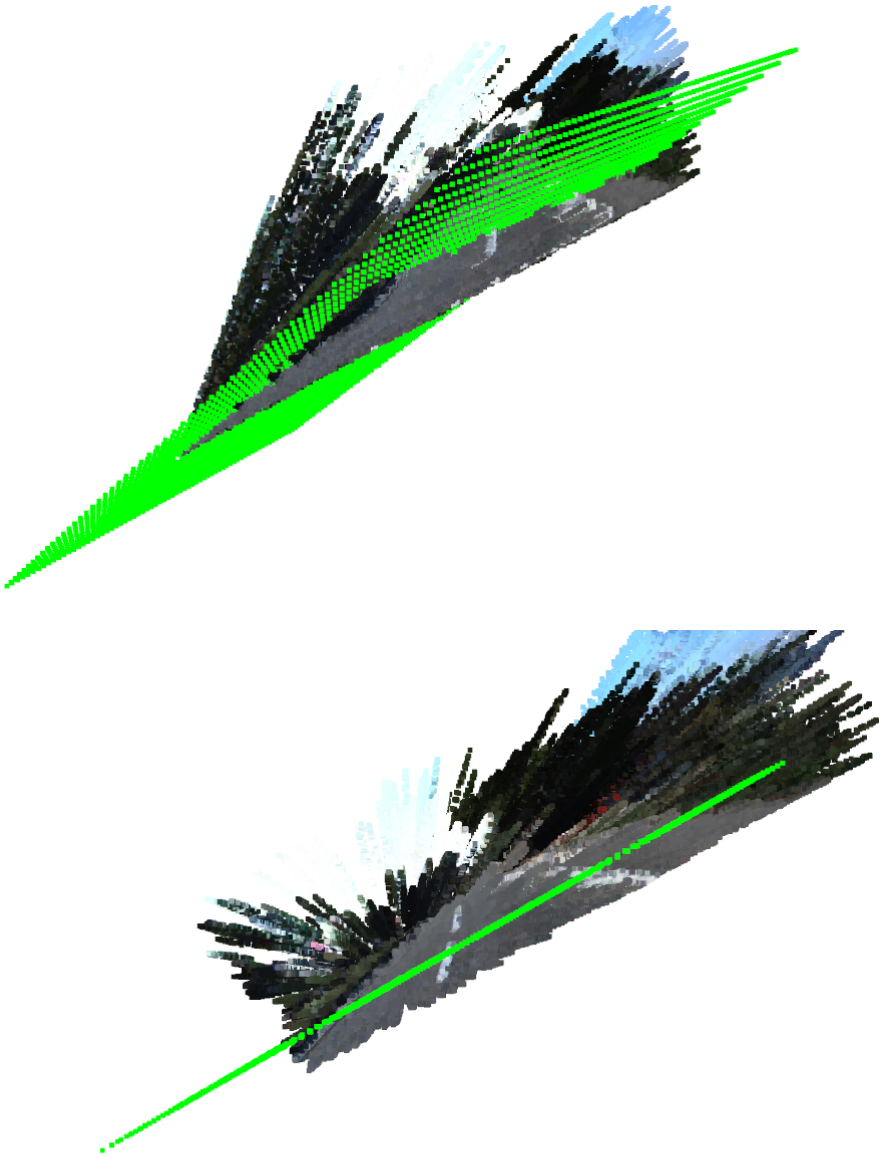
In this task, we used depth map and ground truth mask from previous Subtask. Firstly we get 3D location based on depth map. We used the formula X = Z−(x∗Px) f , Y = Z−(y∗Py) f , where Z is the depth in the depth map, x and y is the image coordinate, then get the 3D location. Then for all the 3D location, we used ground truth mask to filter out points that are not road.

The last step is to fit a plane. Here we used a technique called Least Square to find the best fit plane. The algorithm can find a plane that minimize the least square distance to every point. In order for our algorithm to be robust to outliers, we applied RANSAC over Least Square.



Task 5 3D Point Cloud Road Plane

In this task, we used `open3d` python package help use to convert the road plane and image to 3D point cloud.



The green point is the 3D point cloud of road that we calculate. We can see that it in the bottom of the image and fit the road plane well.

Task 6 Detect Cars

In this task, we used [2]Fast-RCNN: `torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)` help us to detect cars. We used the pretrained model, the performance if good:

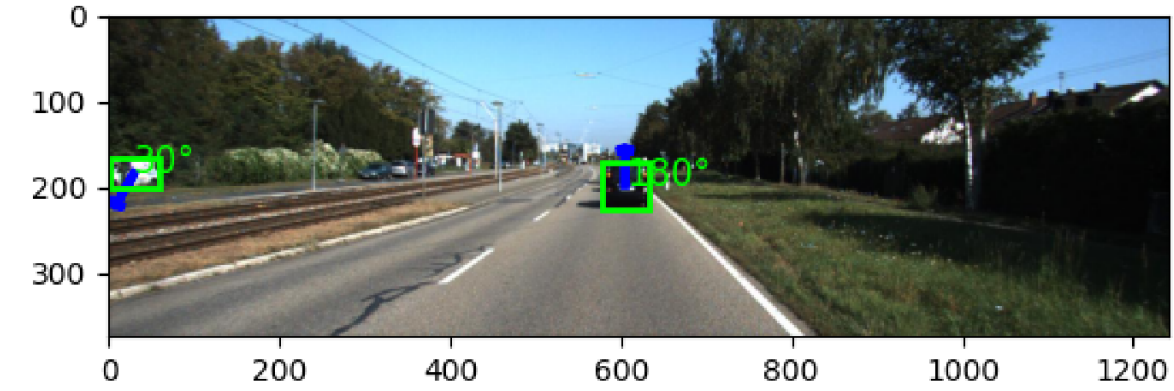
Task 7 Train a Classifier Predicts Viewpoint

For this task, we use the same network architecture in task 3, with some modification, to do viewpoint classification. task. For the training labels, we used one-hot encoder to represents that. Totally, there are 12 viewpoints, so the output should be 1*12 matrix. The last layer, we used softmax, to make each element of the output represent the probability of the prediction for each view. The criterion function we choose binary cross entropy.

For the data process, we user the car pixels as the training data. We filter the car image that is too small to train.

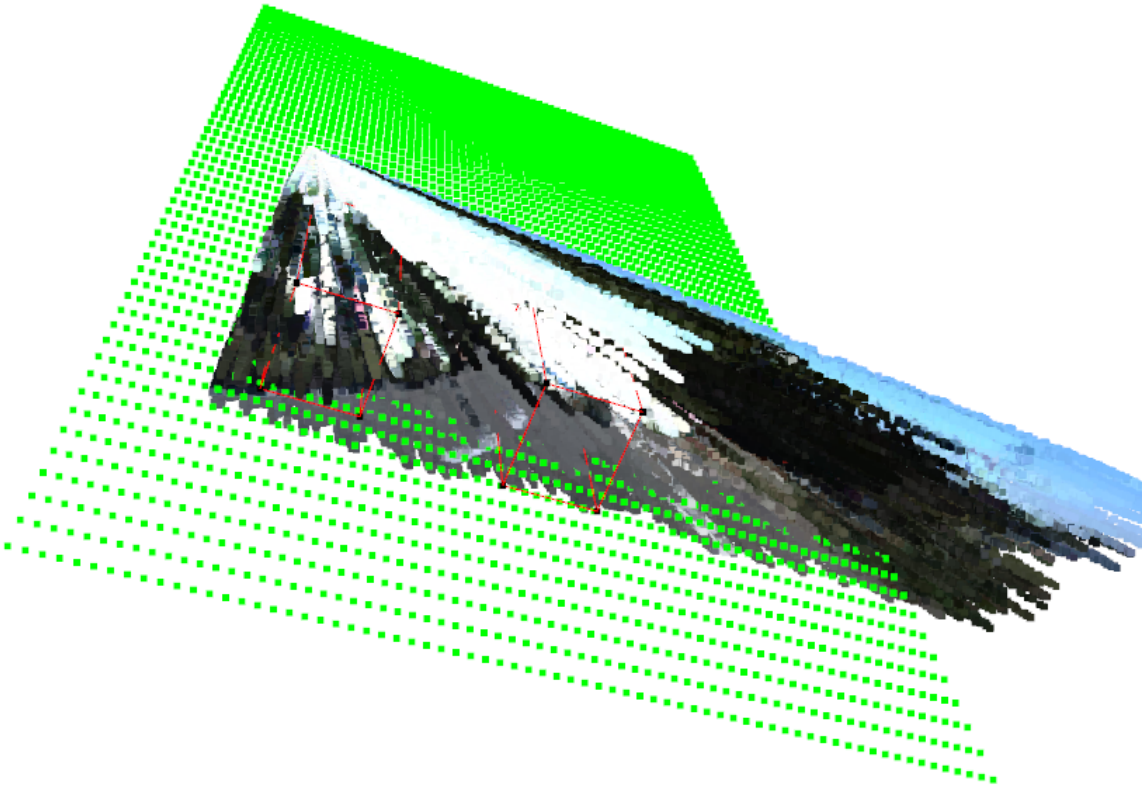
Task 8 Predicts Viewpoint

Here is the performance. The black car is away from us so the direction is 180, the white car is towards us but a little offset.



Task 9 3D Bounding Box of Car

In this task, I used car detector in task 6 to get the left bottom pixels of the car, then I use the depth map in task 2 to get the 3D location of the cars. I used hard code to define the size of the car, and to see the location clearly, I make the size larger. The red lines are the BBOX of cars, I used the same image in task 8.



Reference

[1]Yecheng Lyu, Lin Bai and Xinming Huang. *Road Segmentation Using CNN and Distributed LSTM* https://arxiv.org/pdf/1808.04450.pdf

[2] Ross Girshick. *Fast R-CNN* https://arxiv.org/pdf/1504.08083.pdf