CSC420 Final Project

Autonomous Driving

Kyle Zhou
Yihan Wang
Yining Lin

December 2017

Introduction

Autonomous driving has become one of the most popular topics in machine learning area. It is expected that it will bring up an intellectual revolution to the automobile industry by changing the way we use cars, enhancing the level of safety of driving as well as improving traffic conditions. Moreover, it brings a brand new way for people to interact efficiently and boundlessly. In this project, we have attempted to implement a few techniques that are vital to the functionality of self-driving cars, including road detection and car detection.

For road detection, our task is to detect the area of the road in the image. In order to achieve this task, in the preparation period, we read through some related articles to help us enrich the background knowledge on how to conduct road detection, such as Simple Linear Iterative Clustering (SLIC) for generating superpixels to bring up efficiency of computation, Support Vector Machine (SVM) to train the binary classification model for identify the road area, and RANSAC to fit the plane in 3D for displaying the result of detection.

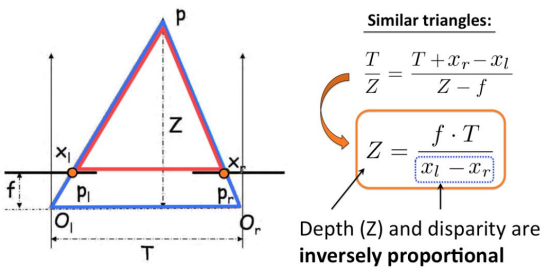
Methods

Road detection

We use SLIC to compute superpixel for clear illustration. The speed of computation has shown to be significantly faster by using superpixel. Meanwhile, we use SVM to train the binary classification model to verify whether each superpixel belongs to road or not. After generating all results using corresponding algorithms, we use pitfitplane, a function that applying MSAC to find the plane. Eventually, we plot all points into 3D cloud to visualize the result of the detection.

Compute disparity
First of all, we need disparities to compute depth from images. So we use pairs of stereo images from two parallel-calibrated cameras to compute disparities for each image by using getDisparity function from spSStereo.

Compute depth
By using focal length from camera intrinsics and baseline from provided data, as well as the two image points from left and right camera, we are able to compute the depth of the whole image. We use the similar triangles rule as a fundamental rule of guidance, and by dividing the product of focal length and baseline using disparity we obtain the result of the depth.



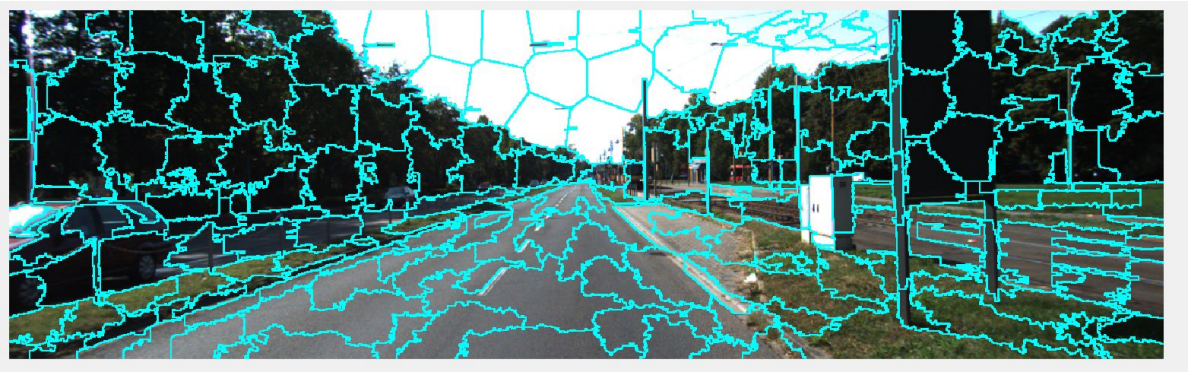
Code:
```
depth = (f .* baseline) ./ disp1;
```

Segmentation

Superpixel is one of the technique that merge similar pixels into a region. This technique can bring up the speed of computation significantly since the 'pixels' that algorithm need to handle will be way more less than the original image. Superpixels are generated using function "superpixels" which uses simple linear iterative clustering (SLIC). 300 is the number of superpixels the algorithm will try to match, and we use low compactness so that it will try to fit the boundaries rather than try to keep superpixels rectangular. Method is set to slic so the compactness is constant during clustering.

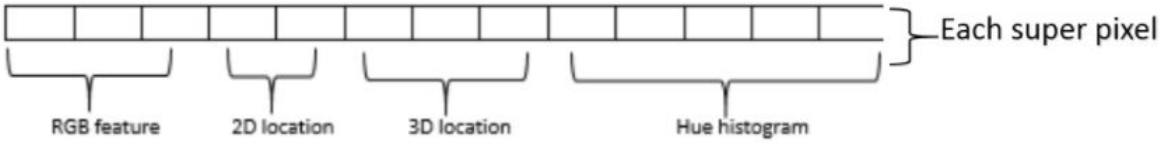
Code:
```
[L,numSP] = superpixels(im,300, 'compactness', 1, 'Method', 'slic');
```

Visulization:



Train a classifier

After segmentation, extract the following features for each superpixel. Concatenating them into a numSP x numFeatures feature matrix.



Features:

For RGB feature, we use the averange RGB value among all pixels within the superpixel.

Foe 2D location feature, we normalize them by dividing by the width and height of the image so that it won't be influenced by image size.

```
features(sp,4:5) = [centerX./M,centerY./N];
```

For 3D location feature, we computed them into camera coordinate system using similar triangle theory.

```
location = [(z.*(m - pX) ./ fX),(z.*(n - pY) ./ fY),z];
```

For Hue, we extract them from image using rgb2hsv, which returns hue, saturation and value matrix.

Then construct a label column extracted from the road identifiers from the gt_left folder. Use the segmentation index matrix on the gt_road_image, and extract the label details by setting each superpixel with more purple pixels than red pixels (controlled by a threshold) will be set

to one. Label column is numSP x 1 matrix which only contains 1 or 0. We use function fitcsvm(all_features, all_labels) to train the model where all_features and all_labels are the concatenation of features and labels of all image's superpixels in the training set.

Apply classifier to find the road
We use function from MATLAB called 'predict' to get the label of superpixel, which indicate specific superpixels.

Car Orientation Detection

To do Car Orientation detection, we decided to train a set of 12 SVM classifiers using the KITTI Object Detection data set's training set. These 12 classifiers represent angles (from the driver's PoV) from -180º to 180º in 30º increments. The KITTI Object Detection data is convenient because object detection labels are provided. We decided to use the following points of interest: 2D bounding box coordinates, 3d bounding box coordinates, and how many corners above a threshold were detected within the car's 2D bounding box.

The rationale for using a corner detector is because different orientations of the car show different numbers of corners. For example, a car pointing forward would have a different amount of corners shown than a car pointing sideways, due to the shape of the car. The corner detectors we tried are the Harris corner detector and SIFT.

After training the 12 classifiers, we used a DPM model to find the locations of cars in the left images of the test set. To find the 3D locations, we ran spsstereo on the provided left and right images, and calculated the depth (see the formulas in Road Detection). We also ran a corner detector on each of the left images, and found the interest points that were within each bounding box. After the 3D locations were found and the corner interest point count was found, we fed this into each of the SVM models. We chose the model which returned both a 1, and the highest score, and plotted the direction on the left image, along with the 2D bounding box of the car.

Lane Change Detection

For the extra work, initially we decided to use train a model for that but unfortunately there are not enough data to train a good performance model. Instead, we used the gt_lane images as predicted label to write a pseudocode to detect lane change action.

Since the left camera's position is located at the origin of the world coordinate system (by KRT) and the car moves in the center of a lane, the bottom (closest to camera) of the lane should be positioned in the middle of the image. With that we can easily compute if the car is changing lane or not by checking if the center of the lane bottom's distance is within a threshold distance to the bottom center of the image.

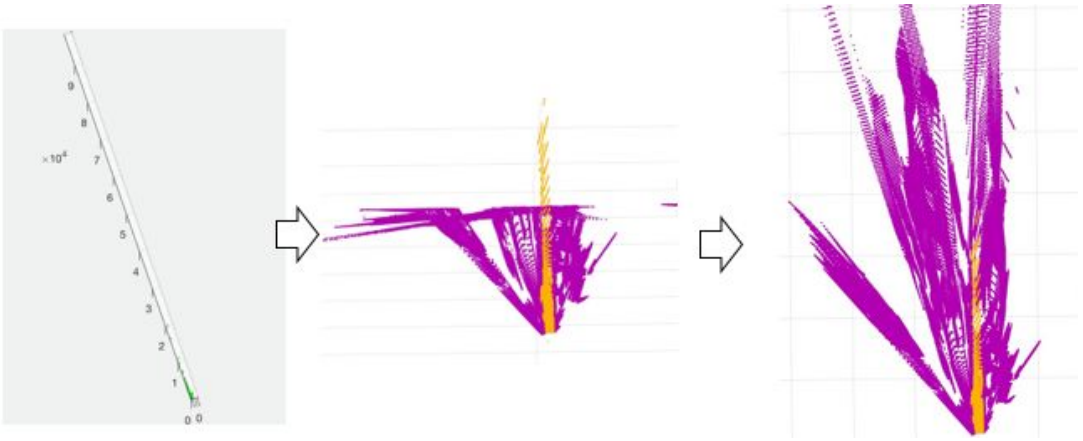
```
if abs(bot_center(2) - img_bot_center(2)) > 50
```

The code for P3 is stored in Part1 file of the code folder.

Main Challenges

Road detection

Deciding which training algorithm and features to be used is a major challenge for us. We solved that by looking at the slides, researching online and asking on Piazza.
In the early state the model kept having false positives that includes the trees in it. It is solved by normalized the 2D location feature, causing it to be influenced largely by different image sizes.
For the 3D point cloud visualization, how to display the point cloud without infinite depth make it hard to identify. That is solved by adding restrictions to x, y and z values while computing the 3D location matrix.



Car Orientation Detection

The main challenges mainly surrounded getting the SVM models to return accurate results. We considered adding more feature points, and using different corner detection algorithms, but sometimes none of the models would return true. Thus, the final set of interest points we ended up with included the 2D bounding box coordinates, the 3D coordinates, and the count of the number of interest points returned by the Harris corner detector. This gave us the best results, without any inconclusive results. We also had to exclude data where the cars were significantly truncated. This was marked by a field in the training set labels. This was necessary because this would skew the results, since the detector we were using would not likely be able to detect partial cars anyways.

Results and Discussions

Road detection

Present the result
Result in 2D will be presented in video.

For displaying the result of detection in 3D, we first fit a plane in 3D. (Figure1)

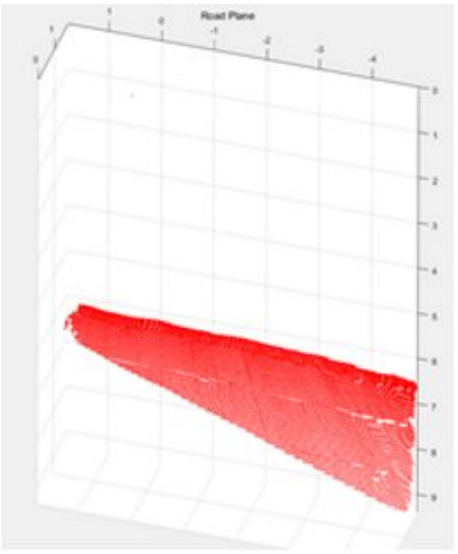


Figure 1

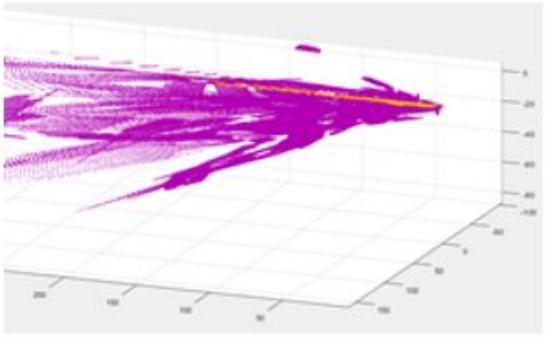


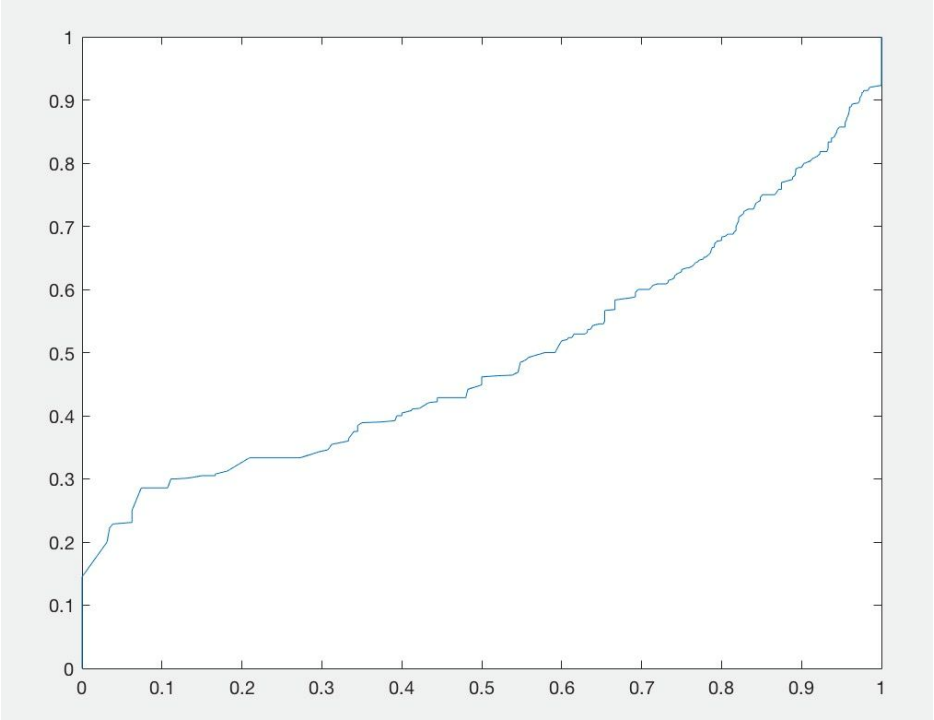
Figure 2

And plot it into a 3D cloud that comes from 3d location of the image. (Figure2)

The result video:
https://www.youtube.com/playlist?list=PL83jtmznhCpnfkNyhCE2E-nFeb8u3FNmL

along with the car orientation detection result. Form the vedio it's clear that the result is not ideal. Several missing superpixels are missing close to the boundaries and some false positives on the side of the road on the trees. Also, it's noticeable that when there's multiple lanes (>2) it's harder for the algorithm to detect the road superpixels. It should be the result of not using shape-related features such as interest points, gradients, 3D gradients or HOG. Roads with similar colors to the training set are generally detected better since we use hue histogram as a feature. Roads in different color or light conditions are harder to be detected because our features rely on color more than shapes. The reason why the shadow is causing false negatives might because of the compactness of the superpixel is too low and some superpixels are mainly shadow and is very hard to be detected.

Precision-Recall Curve



Car Orientation Detection

The results we ended up with were mixed. On some images, like the following scene with two full cars, the results look quite promising.

However, on other images, the results were more mixed:

In this case, the likely cause is that the detected car's bounding box does not encompass the entire car, and so the models are left guessing the orientation. In this case, the confidence score was also very low. Others had the following result:

This is the result of the way the classifiers were implemented. The rotation values were specified in the intervals, [0º, 30º), [30º, 60º), …, [-180º, -150º), etc. During the visualization, I snap the arrow to the left bound of each of these intervals, thus resulting in the arrows being slightly less accurate than they otherwise could be. There were also a bunch that were pointing in the completely wrong direction:

The two cars in the distance are pointing in the completely wrong direction. This may be due to the fact that the models don't properly detect cars that are facing either forward or backwards, especially when they are farther away. Since the front and back of some cars look the same as the left and right, especially for cars that are so far away, the detectors are left guessing. As for the car in the corner, that is likely due to an insufficient number of features in the training set.

For more details on the accuracy of our model, refer to the following confusion matrix: (only first 100 images in training set)

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
1	1	26	21	2	2	1	14	18	2
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Conclusion and Future Work

Road Detection

For the road detection, the result is not ideal when there are strong-contrast shadows on the road, or when the road has a different color. In future training of road classification, we learned that we could also use that using HOG features, gradients, 3D gradients, 2D & 3D SIFT interesting points and corners. If we use more shape-related features as much as our color-related features the algorithm will perform better for detecting road boundaries, and detect road in different color or light conditions. Also, using 3D segmentation rather than 2D should give us a better boundary to reduce false positives.

Car Direction Detection

As for the Car Direction Detector, the results were not as good as we were hoping for. While some of the results were excellent, the combination of an imperfect car detector, and models trained with feature points that may not sufficiently describe all of the features of a car in all lighting conditions, resulted in some arrows that were pointing nowhere near where the car seems to be going.

Future developments include determining some extra features that would more accurately describe the shape of the car, such as using light direction information, and using the colour data to determine the orientation of the car. The width of the bounding box can also be a factor in what direction the car is pointing. Furthermore, using techniques on adjacent frames can also help determine what direction the car is moving, since we can then use tracking techniques to figure out the direction a car is moving.

Lane Change Detection

The algorithm based on the 2D location of the lane is naive and works better when the lane is straight. when there is a curve lane and the care is turning, the algorithm may generates

some false positives. An improvement to this is to fit a lane plane and generates the 3D and check if the camera origin is still position to the center.

Reference

http://ivrl.epfl.ch/research/superpixels

Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk, SLIC Superpixels, EPFL Technical Report no. 149300, June 2010

Jason Brownlee,'Support Vector Machines for Machine Learning' on April 20, 2016 in Machine Learning Algorithms

Features related
http://www.teach.cs.toronto.edu/~csc420h/fall/slides/lecture20.pdf

Appendix A: Assignment of Tasks

Road Detection: Yihan Wang, Yining Lin
Object Detection: Kyle Zhou