**Autonomous Driving Report**

Hongman Cho

Murali Komaravolu, 1000564490

Vibhavi Peiris, 1000597687

**Introduction**

For this project, we were tasked with taking a set of images and correctly identifying the road and the cars on the road. The advancement of machine learning and vision techniques has enabled growth of the autonomous driving industry and research. This was why we decided to tackle this particle project. Not only did it allow us to apply a lot of the algorithms and techniques we have learning through the semester but also apply it in a real world practical setting. If the is done correctly, it can allow cars to detect other obstacles while navigating on the road. It is very important that few mistakes are done else it can lead to accidents.

Before starting, we first researched concepts and looked back at what we learning in the course that could be helpful to our project. Some of the work done from previous assignments were also useful. We thought A3 could be important in finding the depth and 3D location of each pixel. For disparity, we found that there was a built in disparity method that could give us a disparity map. This disparity was required for calculating the depth and 3D location of each pixel. During the last couple of weeks, we learned about HOG descriptors and superpixels, which we thought could work well with training our classifier and predicting the road. For detecting the cars and viewpoints, we thought a modified DPM from A4 could be applied. Once the detections are complete, we just need to visualize them.

We have three members in our team. After researching what needed to be done, we decided that the classification sections for both parts should be worked on together as they were dependent upon each other. The rest of the tasks, such as drawing the bounding boxes and fitting planes, were done separately and we have indicated as such. The process was very free flowing in terms how we were grouped with our tasks.

* (d)  Train a road classifier on a set of annotated images, and compute road pixels in your image. Which features would you use? Try to use both 2D and 3D features.
* (e)  Train your algorithm on the training data, and show example outputs on the testing images.
* (f)  Fit a plane in 3D to the road pixels by using the depth of the pixels. Make sure your algorithm is robust to outliers.
* (g)  Plot each pixel in 3D (we call this a 3D point cloud). On the same plot, show also the estimated ground plane.

**Methods**

***Part 1:***

* First we downloaded the required training and testing images from KITTI. We thought that we could also download DPM from assignment 4 and modify the getData.m file to retrieve the road images like we did with the car images. This made it easier for us to set and get the disparity and calibration values.
* To find the disparity map, we took the left and right images and used the build-in disparity method to return the disparity vales and save them (findDisparity.m). This disparity method takes in two stereo images, one for left and right, and returns a disparity map. It also takes in a Disparity Range which is a two-element vector with a MinDisparity and MaxDisparity. It returned something that looked like this:



* Now that we had the disparity map and the calibration values, we could find the depth of the image using:

***depth = calib.f \* calib.baseline / disparity;***

This equation is used in the findDepth.m method, which is called in the findCloud.m. A pointCloud is created in findCloud.m to store using the calculated depth values.

* Each of us had several different ideas on how to tackle training the road classifier. Murali and Vibhavi thought of originally using the HOG descript to extract features from the images to create a feature vector that consists of the RBG values, the 3D location of the each pixel and the HOG features. Hongman suggested that we should use the superpixels of each image as the features and use them to compare with the gt\_road images. We took this approach instead.

We extracted the superpixels from each of the training images and compared them with the gt\_road. We labeled each superpixel as road or not-road based on its location of the corresponding gt\_road image.

…

* Now we use the features from above to train using svmtrain. The more we train, the more accurate the predictions were.

...

***Part 2***

* Same as with road, we started from downloading the car training and test images. We also downloaded DPM and modified the getData.m file to retrieve the images.
* We used the pre-trained models that we downloaded and DPM to detect cars in the images. We drew 2D bounding boxes and texts around the cars in the image. We did not detect pedestrians and cyclists.

**Main Challenges**

By far the biggest challenge we faced was figuring out what kind of features we would use for both classifiers. Our initial approach of using a feature vector given from the HOG descriptor turned out not to fare well. The speed of the SVM training took a look time with our first approach because we were calculating for each pixel. The use of superpixels fixed this problem because it reduced the number of pixels from thousands to around 500. The speed of the algorithms increased greatly while sacrificing little accuracy. It reduced the time for each image by … %.

**Results**

Dope dope dope …

**Conclusion**

Overall, we are satisfied with the level of accuracy and speed of our solution. We used concept and techniques that we learned through the year starting from RANSAC all the way up to SVM training with superpixels. We believe that this project showcased what we learned and how we would apply them. If we had to move forward with our, we would also train for pedestrians and cyclists to that the car could be even better at navigating safely.