CSC420 Final Project Report

Autonomous Driving:
Road & Vehicle Detection

11.25.2017

Yuheng Huang

999557876

tonyuheng.huang@mail.utoronto.ca

XiaoMing Zhu

1000459782

britha.zhu@mail.utoronto.ca

Introduction

Road & Vehicle Detection is a key requirement for unmanned guided vehicles. However, current vision-based unstructured driving environment detection algorithms are usually affected by continuously changing backgrounds, different road types (shape, colour), variable lighting conditions, other vehicles, obstacles and weather conditions. In this project, we choose to tackle these problems by segmenting the image captured from the perspective of a vehicle then extract important features from the images. Different machine learning models will be used to map out the relationship between these features. There are two parts to this study: road detection and car detection. We compare and contrast different machine learning models such as Binary SVM and Multi class SVM for these problems.

PART A: ROAD DETECTION

— Xiaoming Zhu

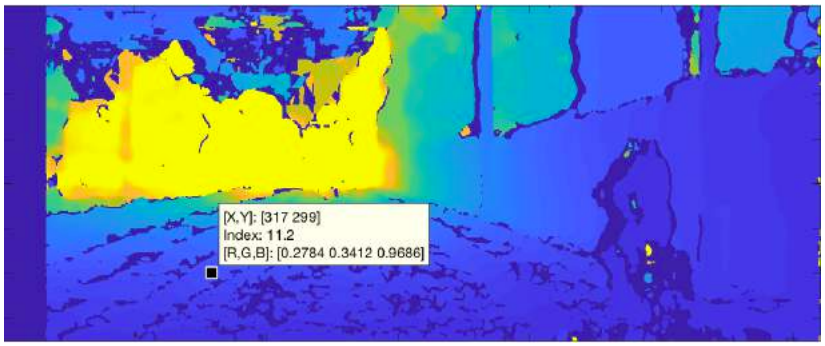
1.METHODS

1.1 Data

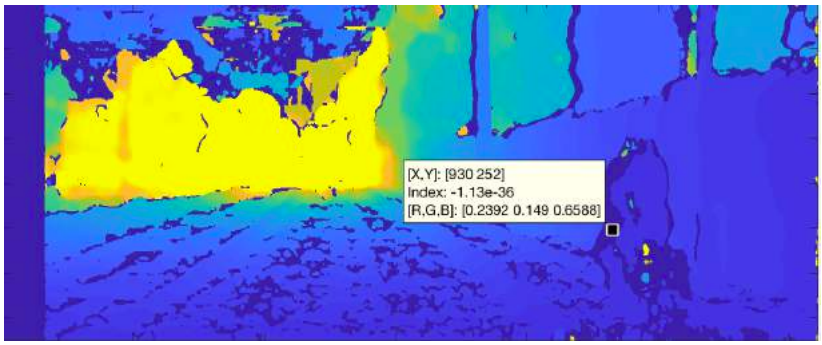
The testing and training data I used is from The KITTI Vision Benchmark Suite. 290 pairs of left and right images were used to calculate disparity and depth for each pairs of images. However only left viewpoint images are used as candidates for training. 5000 road superpixels and 5000 non-road superpixels are randomly selected from all superpixels amount all 290 left viewpoint images. A different 290 images were used as testing set.

1.2 Disparity and Depth

I used the build in disparity (I1, I2) function in MATLAB to get the disparity map of the two stereo images. The two stereo images must be in the same size and must have the corresponding points are located on the same rows. The depth (z) of image can be computed using the formula: $Z = \frac{f \cdot T}{x_l - x_r}$ where *xl - xr* is the disparity of the image, f is the focal length and T is the baseline. The below figures are drawn using the depth of an image. As the scene gets closer to the front, depth which is the index shown on the figure, gets smaller. However,

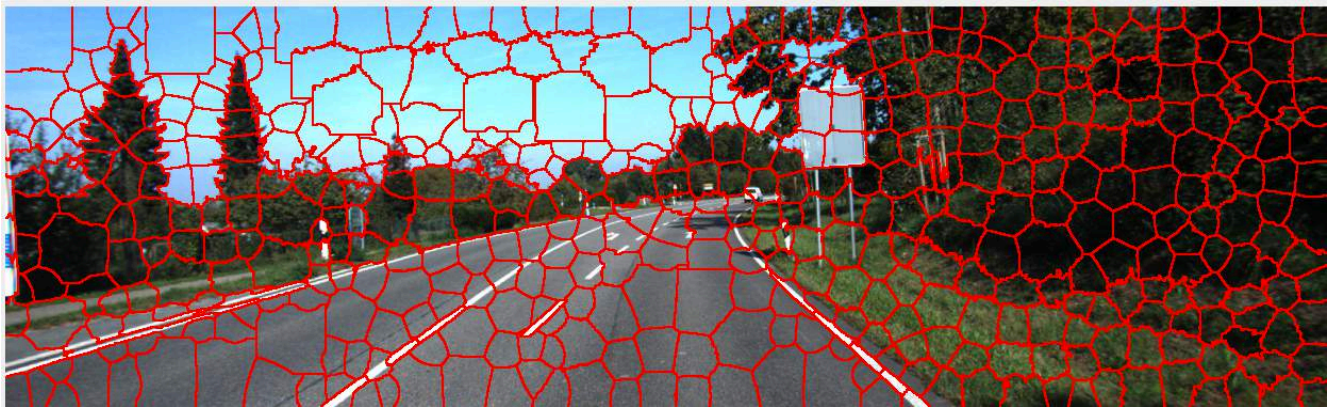


as the figure shows, there are some index with numbers like -1.13e-36. These values do not truly reflects the depth of the current pixel. Therefore these are outliers that need to be eliminated from data that is used to train classifiers and the prediction will be a lot more accurate.



1.3 Approach - Superpixels

While half way solving the problem, I realized there are more than 40,000 pixels in each picture and I need to collect all pixels from 290 pictures. It took me 18 hours to train my SVM and it is nearly impossible to go through each pixel while testing. To solve this problem, I decided to group pixels into superpixels. Superpixels can group pixels with similar intensity together; it is better aligned with intensity edges than a rectangular patch because of its irregular shape. I could now perform extractions and calculations on a super pixel level. The total number of pixels reduced from 40,000 to 500 which greatly improves the runtime. There are two important parameters: the number of superpixels and the compactness. The compactness ranges from 1-20 the smaller the value, the more irregular each patch is. I tested and compared different result from different compactness and number of pixels and settled



with using 500 superpixels and compactness 5.

1.3 Feature Selection

Important features that define roads to non-roads are extracted from images. To ensure the best performance, I tested different combinations of these features on how much they affect the accuracy of prediction. The features that I considered are:

I. Colour

Colour is a very important factor that could differentiate road from non-road pixels. The colour of roads is mostly grey amount all pictures and it is quite distinctive from the surrounding objects. Since I grouped pixels into superpixels, I use the mean of the RGB value from all the pixels in a superpixels. Shadow and lightning change might affect the recognition a little, therefore I decided on using multiple other features like location and texture.

II. Location

Since the position of the road is mostly entered in the middle, I decided that the coordinates of pixels are another important feature. I tested my data on using both camera coordinates (Xc,Yc,Zc) for each pixel and pixel coordinate (Xp,Yp) and finally found out that the combination of pixel Xp,Yp coordinate with camera depth Z gives out the most accurate predictions. See 1.4 for details.

III. Texture

Texture is another very important feature that distinguish the road fond its surroundings. The texture of the road is smoother compare to the texture of grass and buildings but a little rougher than the texture of car. Therefore, my classifier could distinguish road from the car on top and also worked very well when the road is between grasses.

1.4 Feature Combination & Input

- RGB+Texture
- RGB+Xc,Yc,Zc+Texture
- RGB+Xp,Yp+Texture
- RGB+Xp,Yp,Zc+Texture

The input of the sim is in the format [meanX meanY meanZ meanR meanB meanG texture road], where meanX and meanY are the average of the pixel coordinate of all pixels from each superpixel, meanZ is the average depth in a superpixel, meanRGB are the average colour intensity in a super pixel, texture is the average texture and road is either 0, which represent non-road, and 1 which represent road. Binary SVM takes the first 7 features as training features and the last feature road as response

2.MAIN CHALENGES

The main challenge of road detection is Feature Selection. It is hard to decide what features should use to train the classifier. At the very beginning, I just used RGB + XcYcZc features, but the accuracy is very low and that makes me to think whether I should add more features or redefine my feature structure. After few research, I realized that texture feature actually is a very important feature for road detection. Therefore, my final feature combination is RGB+XpYpZc+Texture.

Second, I didn't expect to use superpixel technique at the beginning, which makes my training time very long and not effectivity. Superpixel technique is useful skill for SVM training and road detection, it makes my algorithm much effectivity.

3. RESULTS AND DISCUSSIONS

3.1 Models & Confusion Matrices

I chose to use Support Vector Machine (SVM) because it generates classifier that maximizes the distance (margin) between it and the nearest data point of each class. It only cares about the sample near the margin therefore it could avoid overfitting most of the times. The follow confusion matrixes show the result for the two most accurate SVM algorithms: Cubic SVM and Fine Gaussian SVM

N=1000 Cubic SVM	Predicted Non-Road	Predicted Road
Actual Non-Road Y=0	4790	210
Actual Road Y=1	0	5000

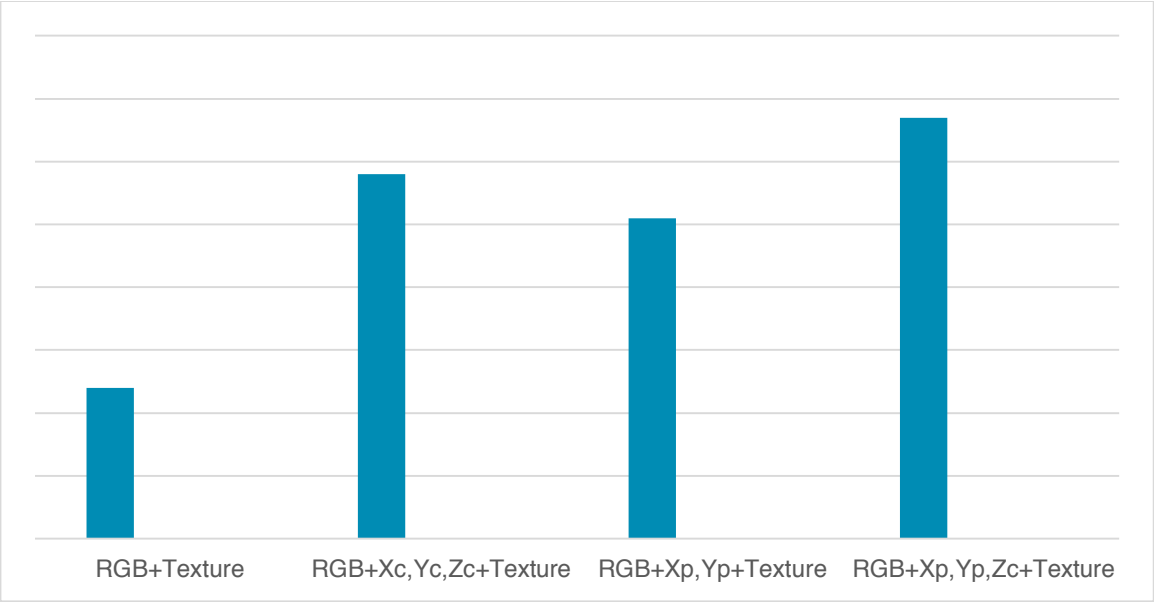
Accuracy = (4790+5000)/(4790+5000+210) = 0.979= 97.9%

N=1000 Fine Gaussian SVM	Predicted Non-Road	Predicted Road
Actual Non-Road Y=0	4864	136
Actual Road Y=1	0	5000

Accuracy = (4864+5000)/(4864+5000+136) = 0.9864 = 98.6%

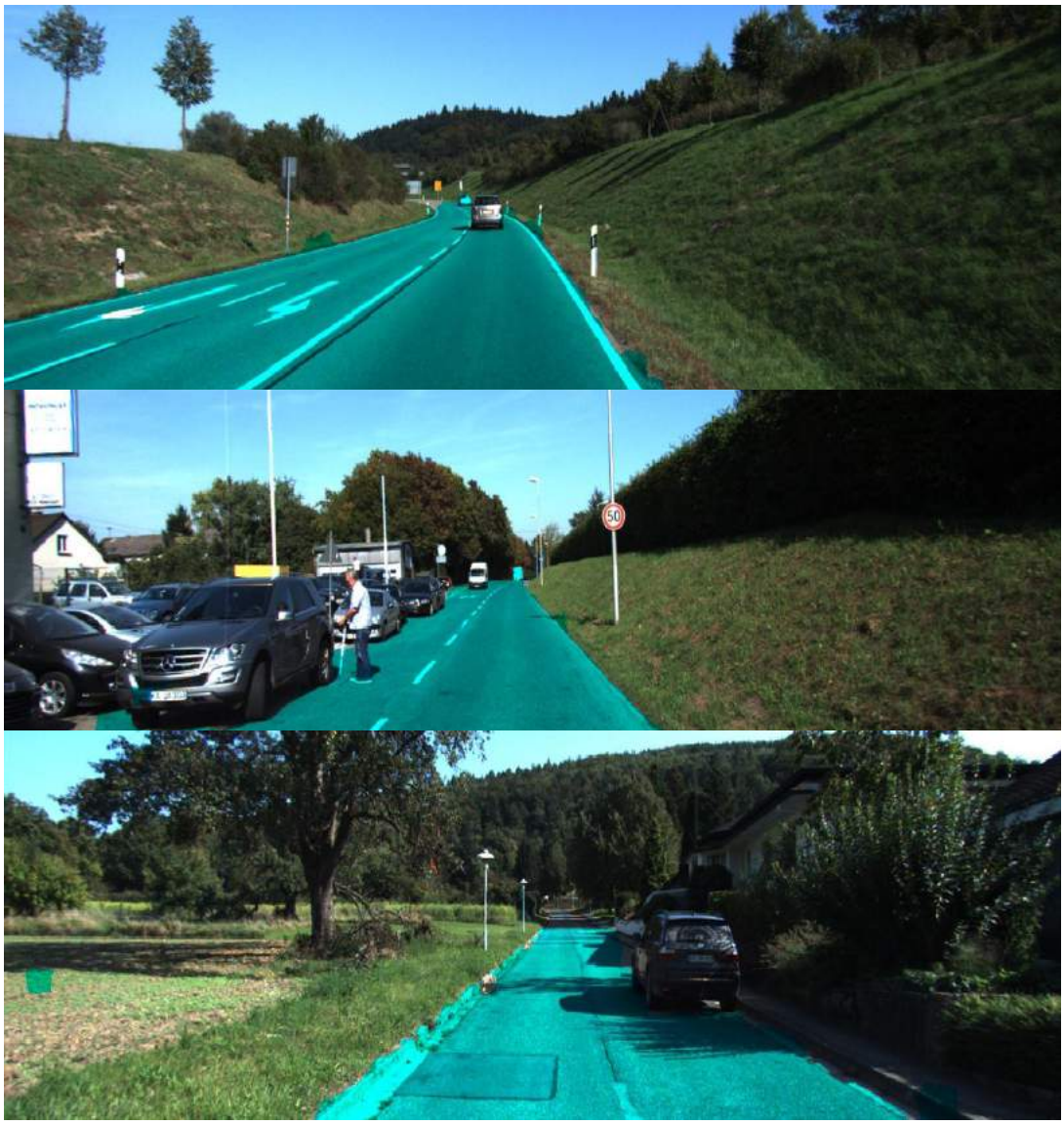
Fine Gaussian did a slightly better job is possibly because it makes more finely detailed distinctions between classes than other algorithms.

3.2 Feature Combination Result



3.3 Road Detection

Good Prediction examples

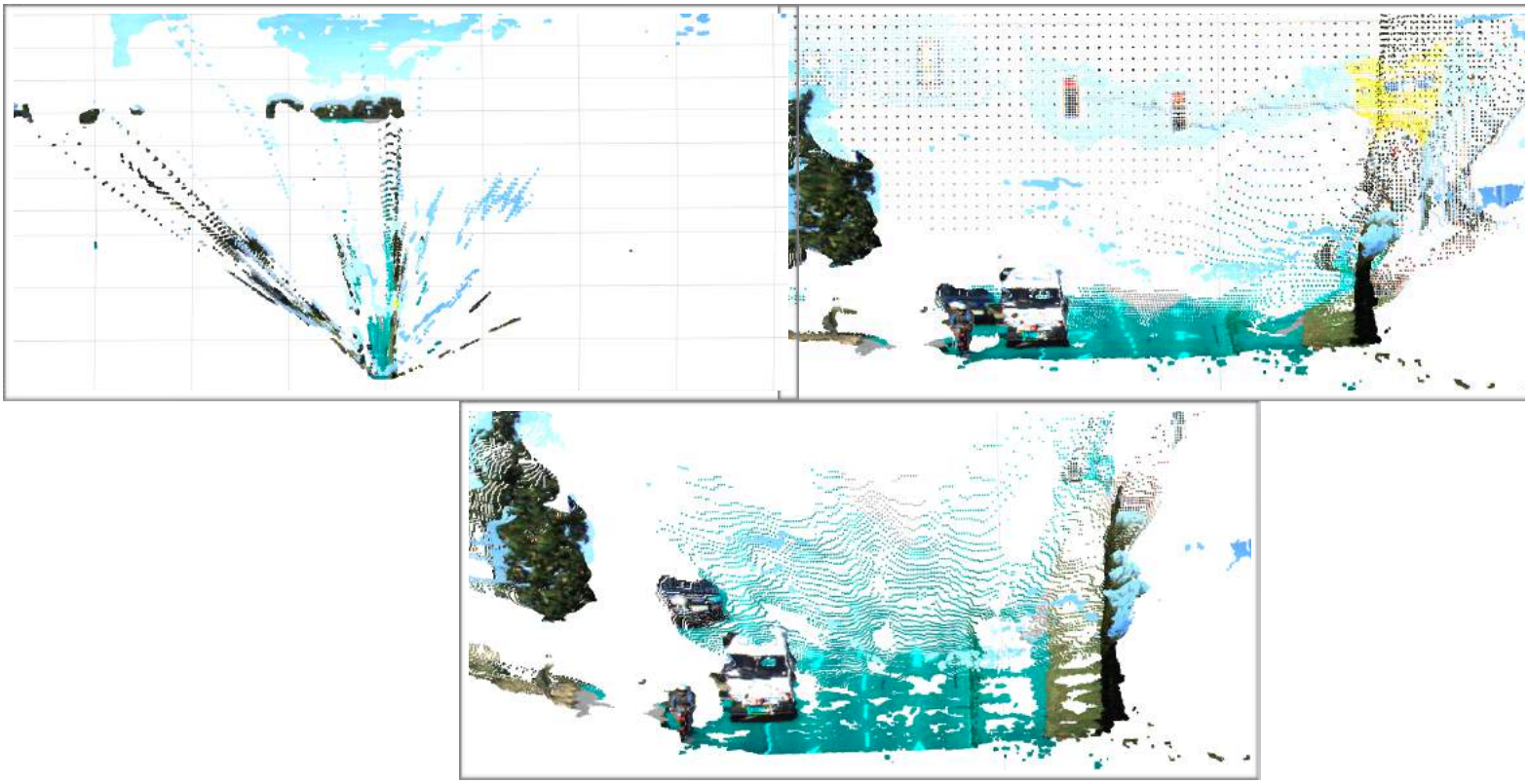


Bad prediction examples

The above two sets of images are the examples of my good predictions and bad predictions. It is obvious that when the road is beside two grass lane the detection works so much better, this is I think is because I used the combination of colour and texture features for training my classifier. However, when there is different road type: train rail, side walk, the classifier doesn't work so well in terms of detecting the actual road since the colour and texture of these different types of roads are very similar. The distance detection works very well in general as I added the depth of the image as a feature.

Next time to improve the performance of the algorithm, I am planning to add HOG as another feature. This way I could detect the bounding line of the lane since Hog is great for detecting gradient and boundaries. By doing that I could eliminate any pixels outside of the boundary line of the lane which will make the prediction a lot more accurate.

3.4 3D pointClout model of the Road



To generate the 3d pointCloud image, I first build a pointCloud object using which first three parameters being images' Xc, Yc, Zc, and the fourth parameter as colour = [R G B]. Then use pcshow(ptcloud) to map the 3d point cloud graph.

The result is not optimum because the depth of the image is calculated only from two stereo images which is not very accurate, it approximates the depth of each pixel but since there are a lot of outliers as I showed from 1.2, the resulting 3d points are not continuous. Even though I eliminate these outliers in my code, it will result in a lot of gaps and discontinuity in the 3d model. Also because that the two stereo images (left,right) provided are very similar in terms of the angle chosen to take the picture, we couldn't get an accurate estimation of what the picture look like from other angle. This causes that the final 3D model is in a cone shape where the centre is the start of the road.

PART B: VEHICLE DETECTION

—Yuheng Huang

1. Methods:

a. Download data:

I download The KITTI Vision Benchmark Suite – Object detection (*includes pre-train model, left and right image, labels file*) as my data. And I organize the data as the following way.

```
Data----------------
  |--------detectors (includes pre-train model)
  |--------train
  |            |--------calib
  |            |--------left (left train image)
  |            |--------right (right train image)
  |            |--------label (training label file)
  |--------test
               |--------calib
               |--------left (left test image)
               |--------right (right test image)
```

* The KITTI Suite did not provide label file for test data.

b. Detect cars in the image:

We used deformable parts model (*dpm.m*) to detected cars with pre-train model. (*model.threshold = -0.6*). And using 2D bounding boxes to enclose each car that we detected. We also imply Non-Maximum Suppression to correctly able to reduce the number of bounding boxes to one. All of the locations of 2D bounding boxes are been stored for further usage.

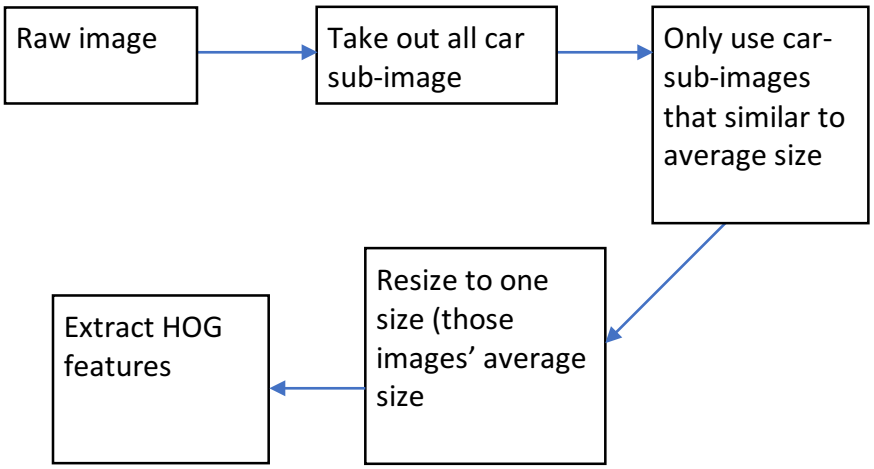
c. Train a classifier that predicts viewpoint for each car:

1. We decided to use HOG feature of each car + SVM classifier to predict viewpoint of each car.
2. Because the image that KITTI proved is raw picture, which means the image not only contains cars but also roads, houses, traffic lights etc. We need to find out all the cars in those images and then convert those cars into HOG features in order to train our classifier in the future.

label file guide:

```
#Values    Name      Description
----------------------------------------------------------------------------
   1    type         Describes the type of object: 'Car', 'Van', 'Truck',
                     'Pedestrian', 'Person_sitting', 'Cyclist', 'Tram',
                     'Misc' or 'DontCare'
   1    truncated    Float from 0 (non-truncated) to 1 (truncated), where
                     truncated refers to the object leaving image boundaries
   1    occluded     Integer (0,1,2,3) indicating occlusion state:
                     0 = fully visible, 1 = partly occluded
                     2 = largely occluded, 3 = unknown
   1    alpha        Observation angle of object, ranging [-pi..pi]
   4    bbox         2D bounding box of object in the image (0-based index):
                     contains left, top, right, bottom pixel coordinates
   3    dimensions   3D object dimensions: height, width, length (in meters)
   3    location     3D object location x,y,z in camera coordinates (in meters)
   1    rotation_y   Rotation ry around Y-axis in camera coordinates [-pi..pi]
   1    score        Only for results: Float, indicating confidence in
                     detection, needed for p/r curves, higher is better.
```

Since different car sub-image has different size, they have different HOG features, we need convert those car sub-images size into one size first before we extract HOG features. However, it may loss details and accuracy if some very small or very large image resize to one size. Therefore, we only select images sizes between *0.7 * average_of_all_car_sub_image and 1.3 * average_of_all_car_sub_image*. This can make sure that sub-images that we select out is not going to lose too much details after they resize because they are actual size are very similar.
After we select those sub-images out, and we calculated the average size of those images and the convert those images into their average size. Then extract their HOG features. ***For example:***



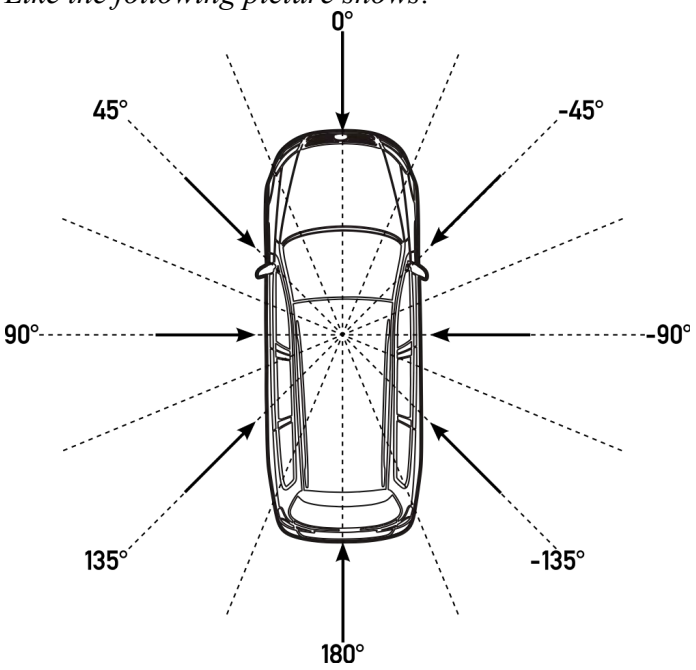
After we finished all that, we get 6838 observations and 2592 HOG features for each observation. And we split the data into 5470 observations for training set and 1368 observations for test set.

3. After we can have HOG features, we can start to train our classifier.

    a. <u>12 classes binary SVM classifier:</u>

    Because the viewpoint labels are in 30◦ increments, we have to create 12 binary SVM classifier. For example, the first classifier classifies whether the viewpoint between 0~30 or other degree.

    Like the following picture shows:



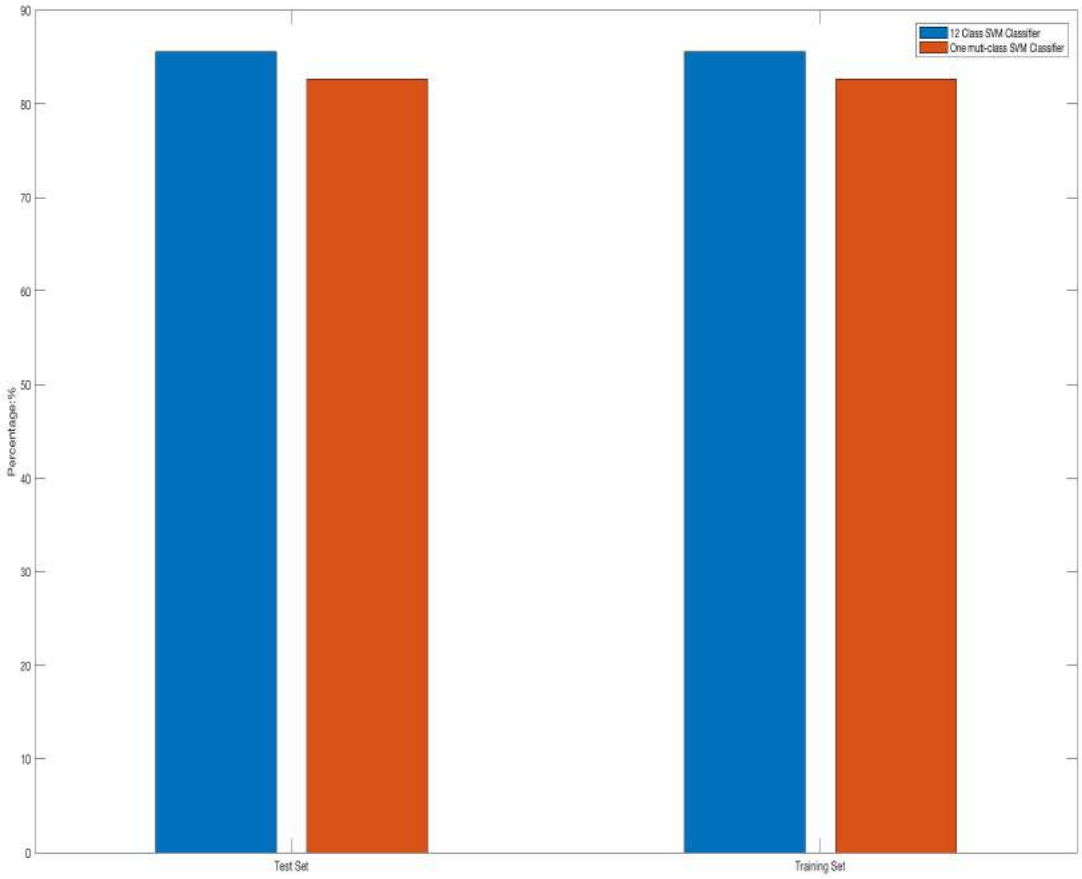
    Therefore, we use our training set to train 12 different SVM classifier *(Kfold = 5)*, each classifier will tell me whether the viewpoint belong to this range and also responds me a score.

    When our DPM for part b. detect a new car, we will crop the car sub-image and convert its size to average size and then extract its HOG features, using its features to run 12 different SVM classifier. After we collect 12 different scores, we will use the highest score as its viewpoint label.

    b. <u>Muti-classes SVM Classifier:</u>

    Instead of training 12 different binary SVM classifiers, we can just train one muti-classes SVM classifier.

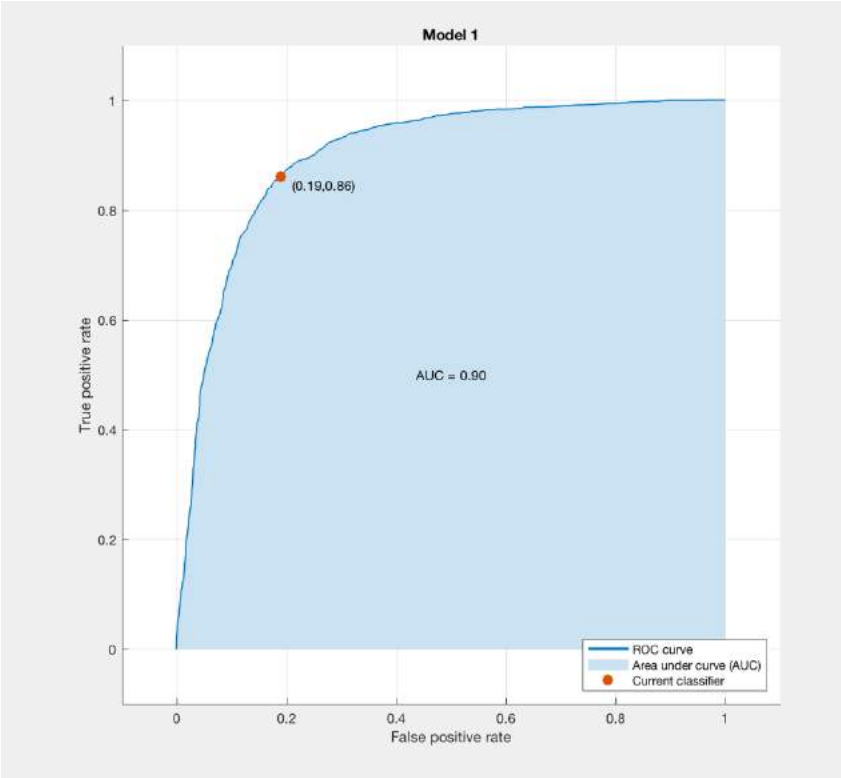
    There is the accuracy comparison between these two classifier:



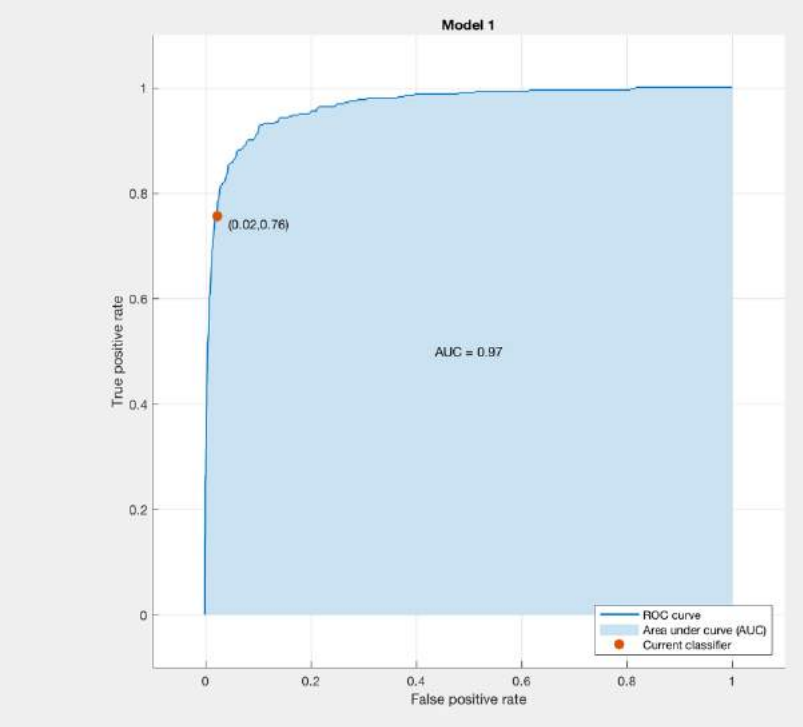
As you can see, the 12 Classes SVM classifiers has a little bit higher accuracy than muti-class SVM classifier. Therefore, we decide to use 12 Classes SVM classifiers to predict viewpoints.

confusion matrix for 12 class-SVM Classifier

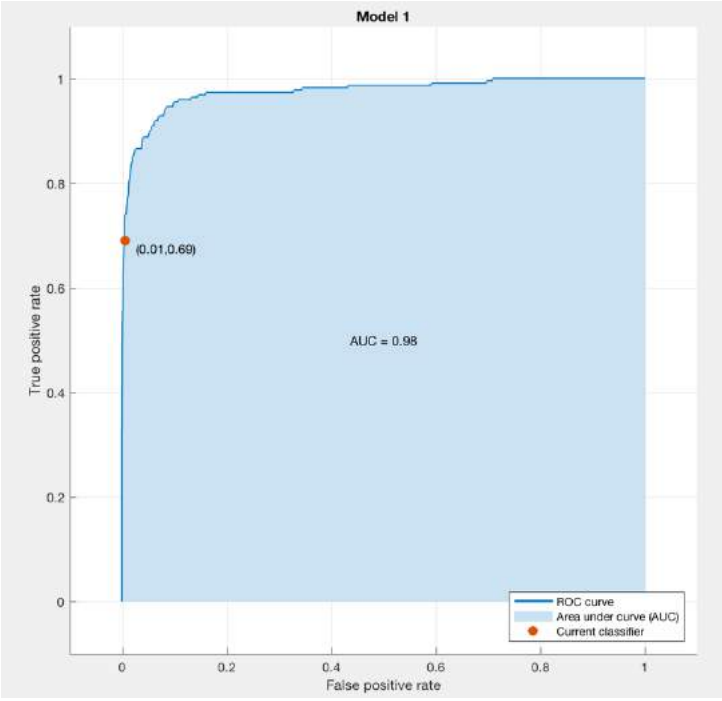
n = 1335	Predicted: 0	Predicted: 30	Predicted: 60	Predicted: 90	Predicted: 120	Predicted: 150	Predicted: 180	Predicted: -30	Predicted: -60	Predicted: -90	Predicted: -120	Predicted: -150
Actual: 0	442	1	5	1	1	0	26	25	3	3	0	0
Actual: 30	0	0	0	0	0	0	0	0	0	0	0	0
Actual: 60	6	0	64	2	0	0	0	1	0	0	0	0
Actual: 90	5	0	5	92	6	0	1	0	1	2	0	0
Actual: 120	2	0	1	3	39	1	3	0	1	0	0	0
Actual: 150	1	0	0	0	0	15	2	0	2	0	0	0
Actual: 180	27	1	2	1	0	2	215	2	3	0	0	0
Actual: -30	18	0	0	0	2	0	2	197	5	0	0	0
Actual: -60	3	0	0	1	0	0	1	4	28	0	0	0
Actual: -90	4	0	1	2	1	0	1	0	1	39	0	0
Actual: -120	0	0	0	0	0	0	0	0	0	0	0	0
Actual: -150	0	0	0	0	0	0	0	0	0	0	0	0



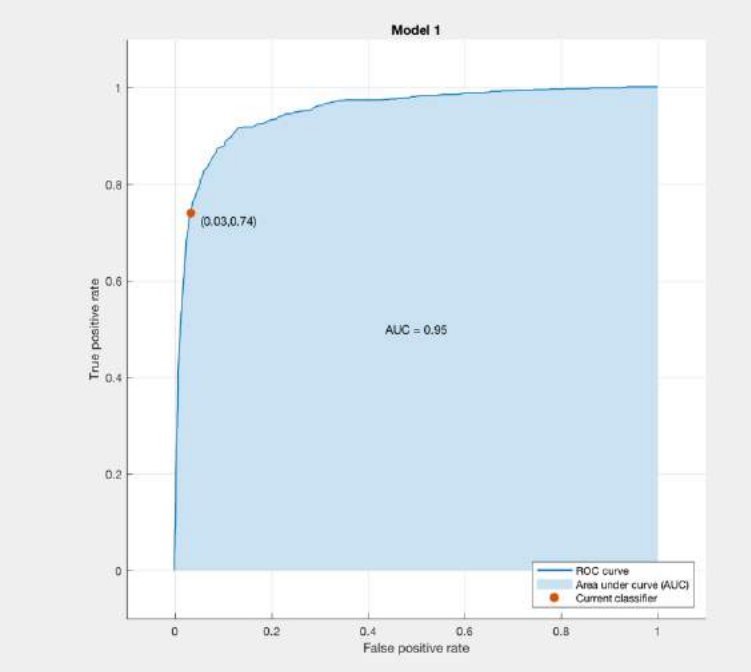
ROC Curve for positive class = 0



ROC Curve for positive class = 90



ROC Curve for positive class = -90

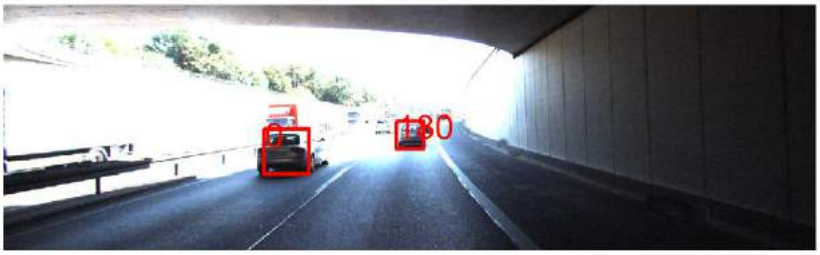


ROC Curve for positive class = 180

d. Show a test image with 2D box and viewpoint

good example

Bad example



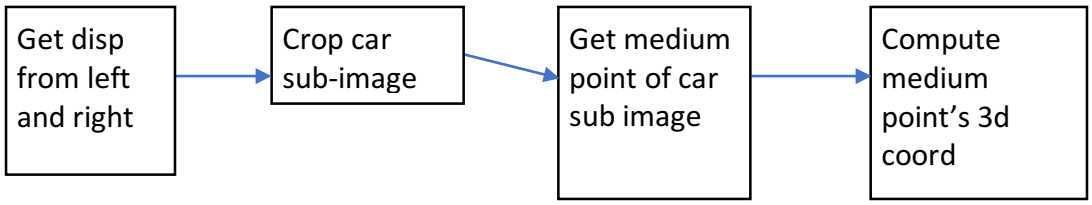
It is very easy to get mix up to detect viewpoint for 0 or 180. Because the HOG feature of 0 degree and 180 degree are very similar, especially for sedan car.

For more test examples, please see our video.

e. compute a 3D bounding box

First, because every car has similar dimensions, like the label file guild above shows, the label file actually provides each car's dimension. So, we calculated the average dimension of all the car's dimension in our data as a fix dimension for car.

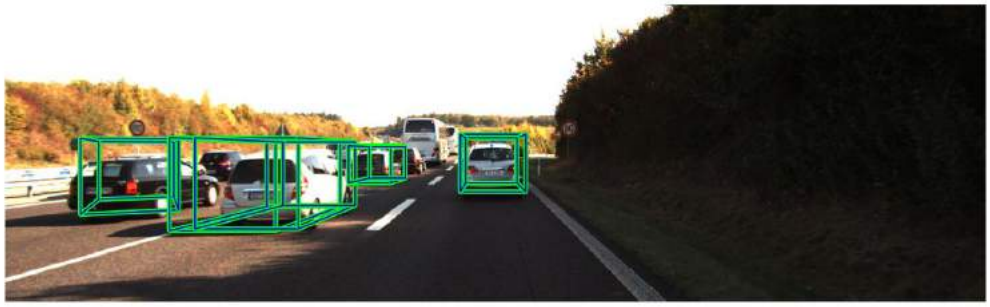
Second, we need to compute the centre of mass, in this case, I use the medium point of car image as the centre of mass. We need to compute its 3D camera coordinate. Before we compute its 3D camera coordinate, we need to compute the disparity first.



Third, because car is usually in the ground, we can assume their y is ground plane, and we have the fixed dimension and centre of mass, base on the viewpoint, we can construe the bounding box in 3D.

3D bounding boxes to your plot from 1.g

After we have 3D coordinate, we can use projection P to project to image pixel. **Examples:**



3. Result and discussions:

For part 2,

- we use DPM to detect cars and use SVM to detect viewpoints of each car.
- The SVM accuracy is 85%.
- We also use fixed dimension method to construe 3d boundary box and project it to 2d by using matrix P.

At the very beginning, we actually use all car image to train SVM, but it is not accuracy, because those images' size is so different, and we force them to resize to one size which make them lose a lot of details. Therefore, we only select images that has similar size to the average size so that it would not lose that much detail when it resizes.

4. Main challenges:

The main challenge for part 2 is we don't know choose what feature of car to train the SVM for predict viewpoint, because if we choose useless features, that means our classifier is useless, we have to choose some features that can split the class label. Finally, we found out HOG is a good feature for predict viewpoint.

Conclusion

Xiaming did part 1's work. In the future, we can try to not only just detect the road, but also detect different footpath, path, alley, sideway and lane, and possibly detect the actually line on the lane.

Yuheng did part 2's work. In the future, we can try to do it in real time for car detection so that it can been used in real industry area.