Autonomous Driving

BY: HONGMAN CHO, MURALI KOMARAVOLU & VIBHAVI PEIRIS

Intro

- Part 1: Finding the road
- Part 2: Finding cars and their viewpoints

Intro

- Goal
  - Identify the road and cars in an image
- Motivation
  - Save lives, make life easier
  - We get to use what we learned in a real life application
  - More and more cars have this technology

Intro

- We all contributed equally to all parts of this challenging project, and there isn't an easy to say who did what part. However below are the rough divisions as requested by the TA.

- Murali: Found the disparity, depth, fit the plane for the road & detected cars. Part 1b,c,f & 2b
- Hongman: trained viewpoint classifiers, predicted viewpoint & the 3d bounding box. Part 2 c,d,e
- Vibhavi: trained road classifiers, predicted road & plotted 3d point cloud. Part 1d,e,g

Part 1 – The Road

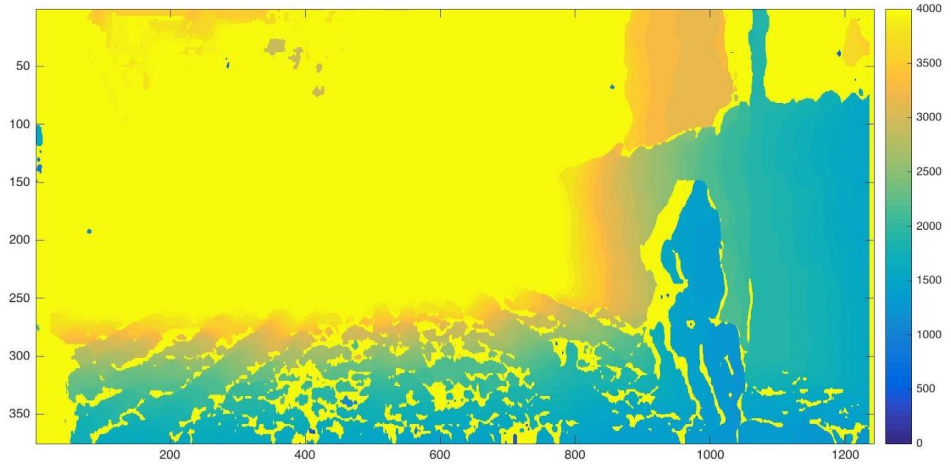
Disparity

- Part 1b
- To find the disparity map, we took the left and right images and used the build-in disparity method to return the disparity values and save them.
- With the Disparity Range of [0 16*15] and a patch size of 15

Depth

- Part 1c
- 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;***



Training

- Part 1d
- First Attempt:
  - We computed features for every pixel
  - We also used HOG as a feature
  - This resulted in really bad and random predictions

Training

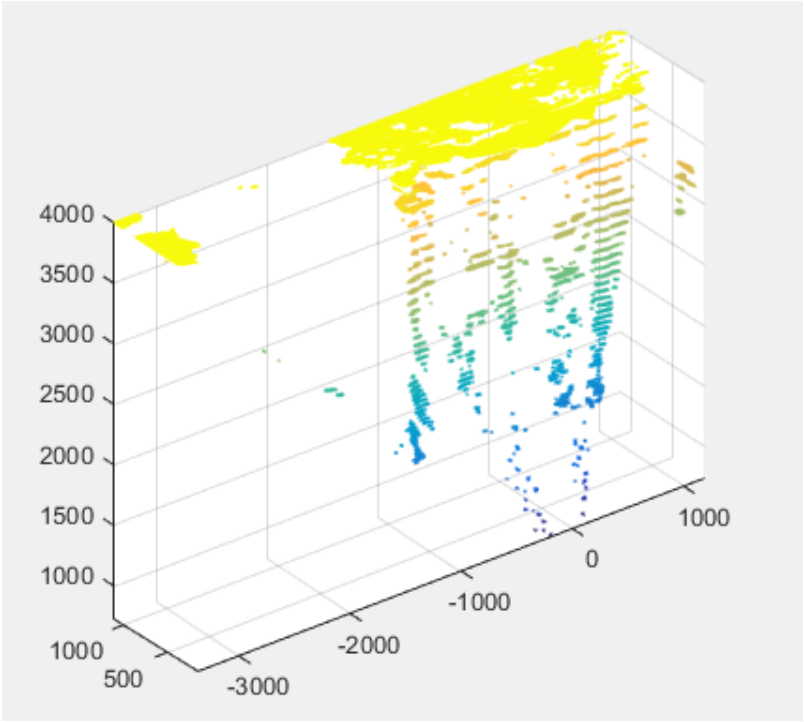
- Part 1d
- Second Attempt:
  - We decided to get features for every super pixel.
    - Around 500 superpixels in an image
  - We created labels for each super pixel using ground truth image
  - Then we got the following features for each image.
    - **- Color (R, G, B)**
    - **- Histogram of Hues**
    - **-3d Y coordinate**
    - **-Gradients in X and Y direction**
  - We choose a 250 random superpixels from each image.
  - Finally we ran svmtrain on all the choosen superpixels

Predicting

- Part 1e
- We computed the feature vector for each test image
- Ran the built-in predict function on those features with our model
- Then created a ground truth and color image of the predictions

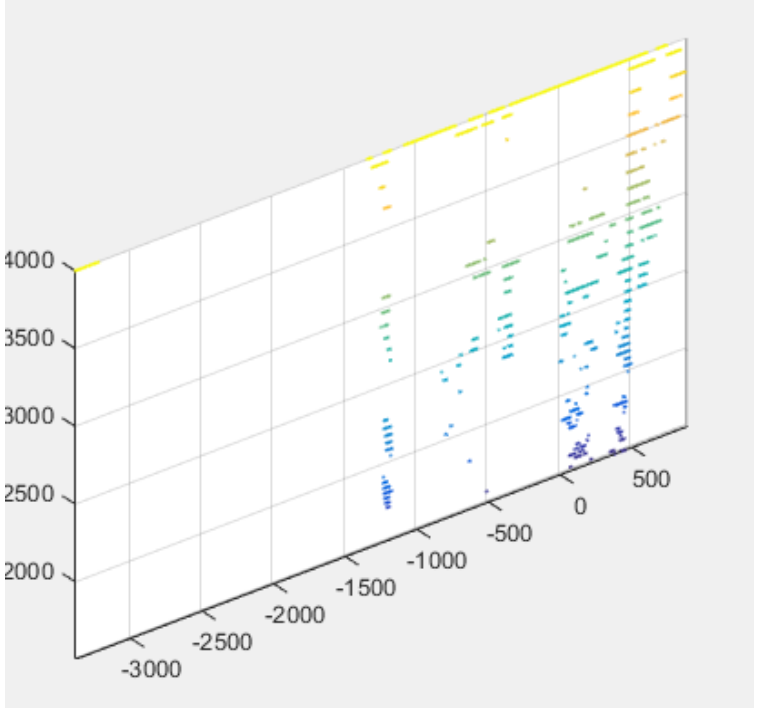
Fit Plane

- Part 1f
- We used select() to return a cropped point cloud given the predicted image and the original point cloud.



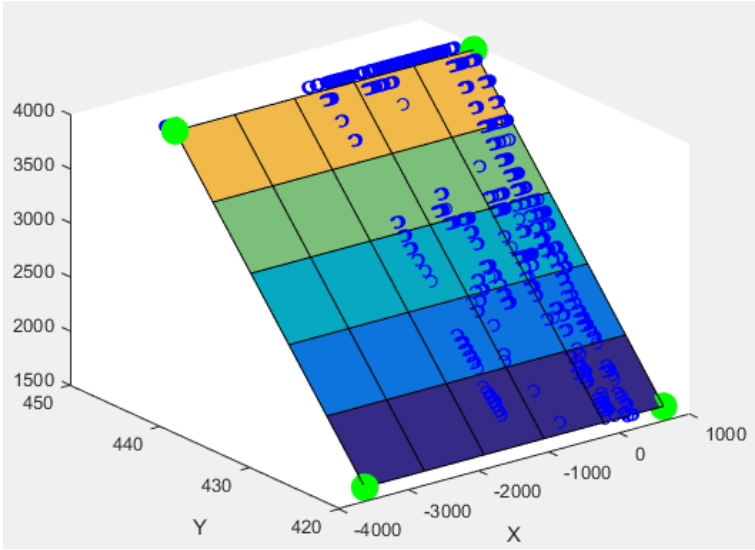
Fit Plane

- Part 1f
- Then, we used pcfitplane to return a plane model that goes through the point cloud and is robust to outliers.
- It returns a set of inliers outliers, and the plane road.



Road point on the ground plane

- Part 1e
- After getting a set of inliers, we fit a mesh grid through the points



Main Challenges

Main Challenges

- Deciding what features to use for road detection
  - Initially used HOG, and got features for every pixel

  - In the end we use color, hue 3dY and gradient
  - Also we got features for every super pixel

Results

Results: Predicted GT Road

- Trained 94 images
- Each took 15 to 30 seconds to compute features
- Total of 48 minutes to train all images

Good Predictions

- Very few non-road object are detected as road
- Detects most of the road as 'road'

Bad Predictions

There are 2 possible solutions to these issues:

- One is to further tune the parameters like what features we use and how many random superpixels we choose. For example, we noticed that using histogram of hue values (as suggested by our professor), as one of our features greatly improved the results.
- The other option is to use ransac to only take predicted superpixels that lie on the ground plane.

Part 2 – The Cars

Procedure for gathering train samples

- 7000 train images are associated with files containing pre-detected object info.
- Used alpha (object orientation) & Bbox (corner coordinates of 2D detection boundary) to crop out train samples from train images.

type	truncated	occluded	**alpha**	**Bbox (x1, y1, x2, y2)**	Dimensions (h, w, l)	location	rotation_y
Pedestrian	0	0	**-0.2**	**712, 143, 810, 308**	1.89,0.48,1.2	1.84, 47, 8.41.1	-1.56

Table 1. Pre-detected object info table (example)

- 17,235 train samples were then saved into 12 viewpoint categories.
  - [1: -120, 2: -150, 3: 180, 4: 150, 5: 120, 6: 90, 7: 60, 8: 30, 9: 0, 10: -30, 11: -60, 12: -90]
- Good samples (high score samples judged by eyes) were then selected for training classifiers.
  - Bad samples such as occluded, shadowed, partial, and inaccurate (labeled as -60 when closer to -30) were filtered out.

Figure 1. Examples of bad samples

- Finally, samples sizes were adjusted to meet the balance between classes. Following table summarize the logistics.

viewpoints	-150	-120	-90	-60	-30	0	30	60	90	120	150	180	Total
# training	877	1110	815	147	79	5448	58	24	1261	52	154	7210	17235
# selected	11	13	20	11	14	73	11	6	37	10	9	100	315
# final	10	10	10	10	10	10	10	6	10	10	9	10	115

Table 2. Summary of training sample filtering process

Procedure for gathering test samples

- 531 test images
- 311 test images were detected using pre-trained car model.
- The detected corner coordinates were used to crop-out the test samples.
- Same rules for grouping into viewpoint categories is applied.

Expected viewpoints	-150	-120	-90	-60	-30	0	30	60	90	120	150	180	Total
# selected	25	5	20	66	81	31	8	4	31	5	8	27	311

Table 3. Summary of test sample filtering process

Class training rules

- Selected equal class sample sizes
  - For every class training, we have sampled approximately the same sample size for each chosen class.
- Train sample size = Test sample size
  - Also we tried to match the test sample size to the train sample size.
- Applying these two rules resulted in high confidence classification results.

Classifier training procedure

- HOG for extracting image features.
  - With fixed feature size: 3 by 3 (=3*3*9) → all samples have the same number of features regardless of scale.
- Used ‘fitcsvm()’
- Total of 12 classifiers were generated.
- Each classifier is trained with two labels:
  - Positive label: a viewpoint class
  - Negative label: every other class

Classification procedure

- Similarly, used HOG to generate test sample features.
- Used ‘predict()’
- Predicted labels were generated by classifying test features against every trained viewpoint model.
- Each predicted label was aligned with expected-viewpoint labels to evaluate the accuracy of classifiers.

Classification test result

Table 3: Classification Test #1 (front-view vs left-view)

viewpoints	-150	-120	-90	-60	-30	0	30	60	90	120	150	180	Total
# training	-	-	20	-	-	20	-	-	-	-	-	-	40
# testing	-	-	20	-	-	20	-	-	-	-	-	-	40
# positively classified	-	-	16	-	-	17	-	-	-	-	-	-	33
Confidence	-	-	0.8	-	-	0.85	-	-	-	-	-	-	**0.825**

Table 4: Classification Test #2 (4 viewpoints)

viewpoints	-150	-120	-90	-60	-30	0	30	60	90	120	150	180	Total
# training	-	-	18	-	-	20	-	-	20	-	-	20	78
# testing	-	-	20	-	-	20	-	-	20	-	-	20	80
# positively classified	-	-	6	-	-	11	-	-	9	-	-	6	33
Confidence	-	-	0.3	-	-	0.55	-	-	0.45	-	-	0.3	**0.4**

Table 5: Classification Test #3 (12 viewpoints; <= 10 samples each)

viewpoints	-150	-120	-90	-60	-30	0	30	60	90	120	150	180	Total
# training	40	40	18	35	35	35	33		17	27	36	36	352
# testing	10	5	10	10	10	10	8	4	10	5	8	10	100
# positively classified	2	0	0	2	0	0	2	0	0	0	2	0	8
Confidence	0.2	0	0	0.2	0	0	0.25	0	0	0	0.25	0	**0.08**

Classification test analysis analysis

Factors contributing to classification accuracy

- Feature similarity between classes
  - More viewpoint classes → less distinguishable features between positive and negative → accuracy ↓
- Sample size between classes
  - Evenly balanced sample size between classes → accuracy ↑
- train sample size vs test sample size
  - Similar sample size between train and test → accuracy ↑

- Uncertain factor:
  - Ratio of expected-classes in test sample to the ratio of classes in train samples.
    - Similarity in ratio → accuracy ↑

Computing 3D bounding box

Figure 2-1. Computing segmented points of the detected object.

Figures 2-2. Computing 3D-bounding box around detected car.

Once computing segmented points of a detected object (from A4), depth of center point, averaged depths of the closest and furthest points in segmented pool were used to compute the ratio for adjusting boundary size.

Then the boundaries were placed at the averaged closest and furthest points.

Main Challenges

- The main challenge of classification problem was to be able to find the factors contributing to the classification results.
- Because the classification of test data depended on the controlled train data with respect to the expected classes and number of test data, it presented a problem when test objects needed to be classified separately.
- Having too many viewpoint divisions resulted in similar features between classes which contributed to low confidence score of classification results.
- There were also large data number gap between prevalent and rare viewpoints. This led to disproportionate number of samples between prevalent and rare viewpoint classes which in turn added to the low accuracy classification results.