Project Report

Autonomous Driving

Tianshu Zhu, Haoran Fang

Introduction:

Our problems is to use neural network to detect road and vehicles in the real life situation. As the development of machine learning through these years, autonomous driving is no longer a unpractical idea. Some big companies like Uber, Google, Apple are working on developing their own auto-driving system. The core tech of autonomous driving is road detection and car detection. Through CSC420, we learned how to detect features and find depth. In order to use Stereo to calculate the depth, we read the paper “ Efficient joint segmentation, occlusion labeling, stereo and flow estimation” . Also we reference another article “ Feature extraction for Vehicle Detection using HOG+” to find a efficient way to extract feature for training.

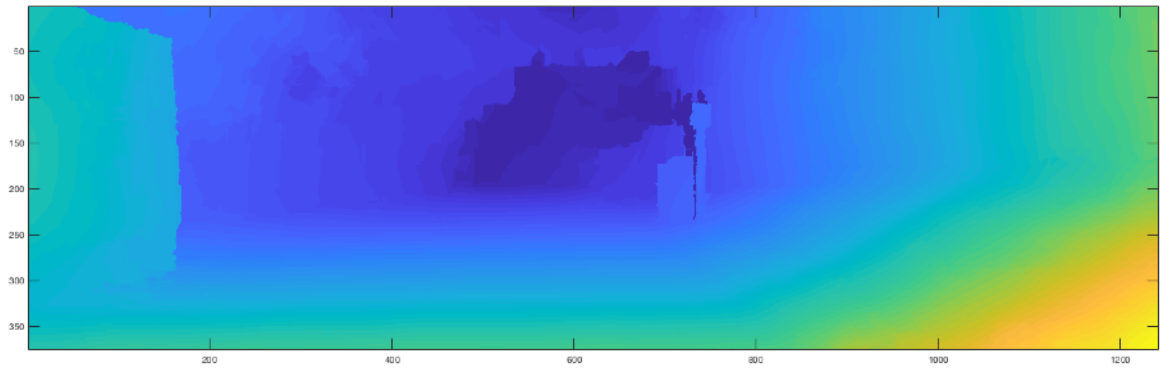
Methods:

Q1

b)

calculated the disparity of stereo images. In this part, we the reuse the code “spsstereo” from TTIC Chicago to compute the disparity between left and right image.

Result:



c)

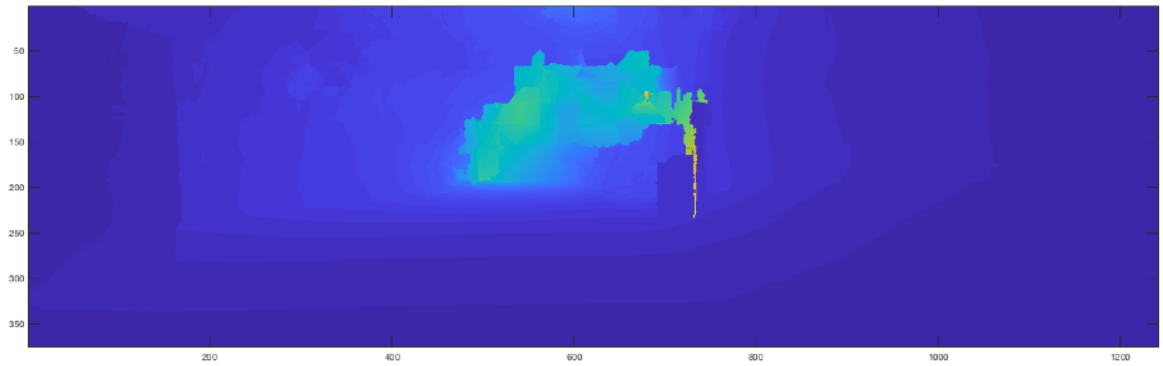
Computed depth using disparity.

```
% compute depth
camera_parameters = getData(imname, imset, 'calib');
f = camera_parameters.f;
T = camera_parameters.baseline;
disparity_struct = getData(imname, imset, 'disp');
disparity = disparity_struct.disparity;
depth = f*T./(disparity);
```

Computed 3D location using depth.

```
% compute and save 3d location for each pixel in image with name imnam
location = zeros(num_rows, num_cols, 3);
for row = 1:num_rows
    y = num_rows+1-row;
    for col = 1:num_cols
        x = col;
        Z = depth(row, col);
        result = K\[x; y; 1];
        w = Z/result(3);
        X = result(1)*w;
        Y = result(2)*w;
        location(row, col, :) = [X Y Z];
    end
end
```

Result:

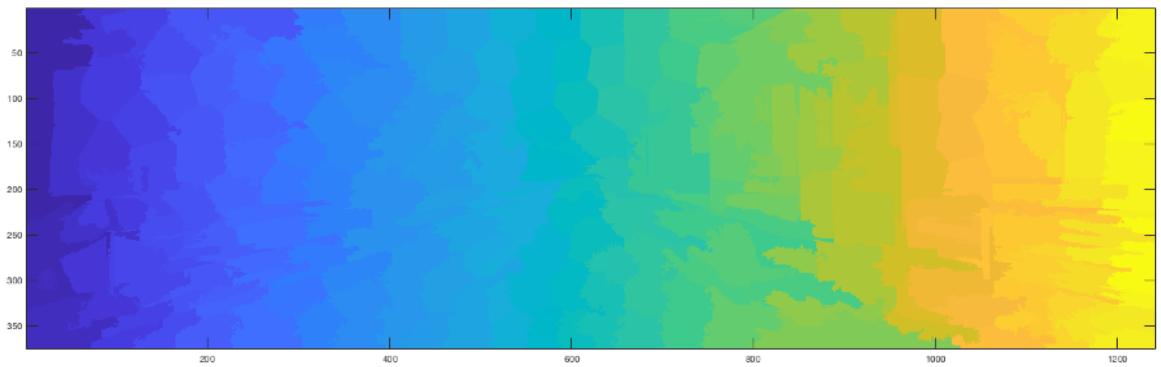


d):

Used “matlab’s superpixel()” function to do segmentation. Then for each super pixel, used histogram of hue, std of hue, mean 2D x, mean 2D y, mean 3D y, min 3D y, max 3D y, std 3D y as training features

```
% extract feature of each superpixel
% extract histogram of hue
num_bins = 16;
im_hsv = rgb2hsv(im);
im_hue = im_hsv(:,:,1);
sp_hue = im_hue(sp);
[sp_hue_hist_values,edges] = histcounts(sp_hue, num_bins, 'Normalization','Probability');
sp_hue_hist_values = sp_hue_hist_values*10; % scale a little
std_sp_hue = std(sp_hue);
% extract 2D location
[sp_row, sp_col] = find(sp);
avg_sp_y2 = mean(sp_row)/10; % scale a little
avg_sp_x2 = mean(sp_col)/10;
% extract 3D location
location_y = location(:,:,2); % may not work since location is nxmx3
sp_y = location_y(sp);
sp_y = reshape(sp_y,[1,size(sp_y)]); % check dimension, may need to use (:,2)
avg_sp_y3 = mean(sp_y);
std_sp_y3 = std(sp_y);
max_sp_y3 = max(sp_y);
min_sp_y3 = min(sp_y);
feature = [sp_hue_hist_values std_sp_hue avg_sp_x2 avg_sp_y2 ...
    avg_sp_y3 std_sp_y3 max_sp_y3 min_sp_y3];
if isnan(std_sp_y3)
   continue;
end
```

Result

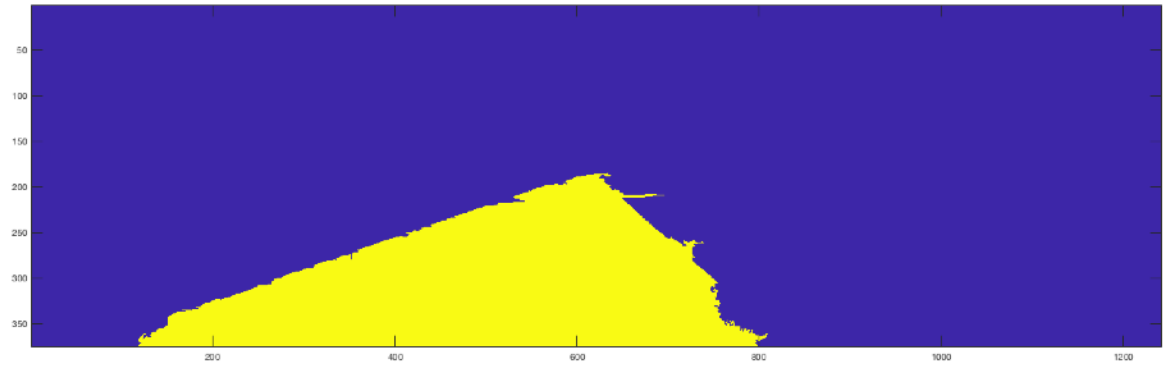


e):

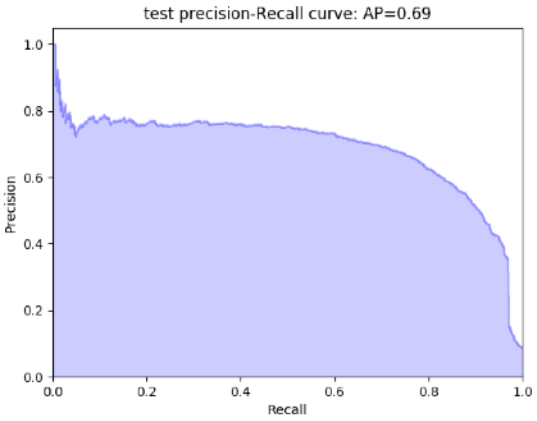
Trained binary classifier using “svm” from “python’s sklearn” package. Used “Python’s pickle” package to serialize the trained model.

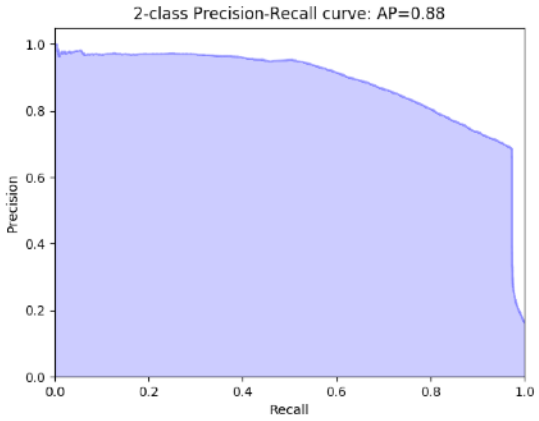
```
# read in features and labels
features = []
features_filename = "../data/car/train/results/train_data/train_feature.mat"
with open(features_filename,'r') as ff:
    for feature_string in ff:
        feature_string = feature_string[0:-2]
        feature = [float(x) for x in feature_string.split(" ")]
        features.append(feature)

labels = []
labels_filename = "../data/car/train/results/train_data/train_label.txt"
with open(labels_filename) as lf:
    for label_string in lf:
        label_string = label_string[0:-1]
        label = float(label_string)
        labels.append(label)

# fit models
print("started training\n")
print(datetime.datetime.now())
model = svm.SVC()
model.fit(features, labels)
print("finished training\n")
print(datetime.datetime.now())
```

Result



Train accuracy: 0.93611
Test accuracy: 0.92537
Note: the left plot is for test, the right plot is for train





f)

by using the detected road picture, the road can be detected by using filter. A threshold is set for detect road. On the detected image, we marked the road as 255 and the rest as 0. So, we can extract the road part from the image. pointCloud() is used to plot the road in 3d space. Since we already have the each pixels' 3d location, its easy to create a 3d plan of the detected road.

```
imset = 'test';
imnames_fid = fopen(fullfile('../data',imset,'/results/train_road_detect.txt'));
imname = fgetl(imnames_fid);
while ischar(imname)

    marked_road= load(fullfile('../data',imset,'/results/roads_300',imname));
    im_split = strsplit(imname,'_');
    real_name = strcat(im_split{1},'_',im_split{2});
    real_road = imread(fullfile('../data',imset,'/left',strcat(real_name,'.png')));


    marked_road = marked_road.im_road;

    % perform thresholding by logical indexing

    image_thresholded = real_road;

    image_thresholded(marked_road>0.5) = 255;
    % image_thresholded(marked_road<0.5) = 0;

%     display result% figure()
%     subplot(1,2,1)
%     imshow(marked_road,[])
%     title('original image')
%     subplot(1,2,2)
%     imshow(image_thresholded,[])
%     title('thresholded image')

    location_filename = fullfile('../data',imset,'results/road_detect_results_300',strcat(imname, '_detect.png
    imwrite(image_thresholded,location_filename);
```

g)

Here we use the threshold to find the road on the original pic. When found a pixel has non-zero value on the detected image, we mark the pixel on the corresponding location of original image. So at the original image we will have a marked road appear on the graph. Use pointCloud(), we can plot the original image with road marks to the 3d place

```
roadImage = imread('../data/test/results/road_detect_results_500/um_000012_road_detect.png');

xyzMatrix = zeros(375,1242,3);
S = load('../data/test/results/location_results/um_000012_location');
xyzMatrix(:,:,:) = S.location;

ptCloud = pointCloud(xyzMatrix,'Color',roadImage);

P = boundary(ptCloud);
plot3(P(:,10),P(:,20),P(:,30),'.','MarkerSize',10)



player3D = pcplayer([-50, 50], [-5, 30], [0,80], 'VerticalAxis', 'y', ...
    'VerticalAxisDir', 'down');


view(player3D, ptCloud);
% pcwrite(ptCloud,'ptCloud');
```

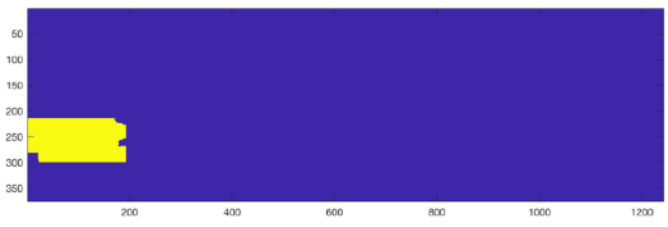
Q2

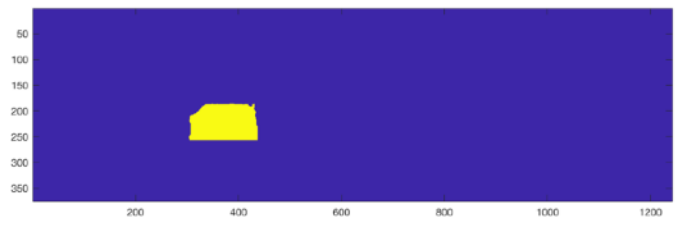
b):

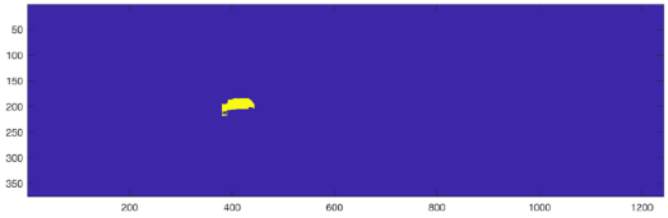
Used “DPM” detector provided in a4 to detect cars. Did basic segmentation for each car based on 3D location. Computed centre 3D location of each car.

```
% compute center location 1 as mean location of bounding box
detection_location = location(y1:y2, x1:x2, :);
detection_location(isinf(detection_location)) = 0;
detection_location(isnan(detection_location)) = 0;
center_location1 = reshape(mean(mean(detection_location, 1), 2), [1 3]);

% compute segmentation based on center location 1
for col = x1:x2
    for row = y1:y2
        pixel_location = reshape(location(row, col, :), [1 3]);
        if norm(pixel_location-center_location1) <= 5
            segmentation(row, col) = 1;
            segmented_location = [segmented_location; pixel_location];
        end
    end
end

% compute center location 2 as mean location of segmentation
if size(segmented_location,1) > 0
    center_location2 = reshape(mean(segmented_location, 1), [1 3]);
else
    center_location2 = center_location1;
end
```

results:







c):

Used “Matlab’s extractHOGFeatures()” to extract hog of detected patches of cars. Used Hog and downsampled HSV raw image as training features.

```
patch = imcrop(im,[x1 y1 width height]);

% hog features
hog_feature = extractHOGFeatures(imresize(patch, [64 64]));

% hsv features
patch_hsv = rgb2hsv(imresize(patch, [32 32]));
hsv_feature = reshape(patch_hsv, 1, []);

% feature
feature = [hog_feature hsv_feature];
```

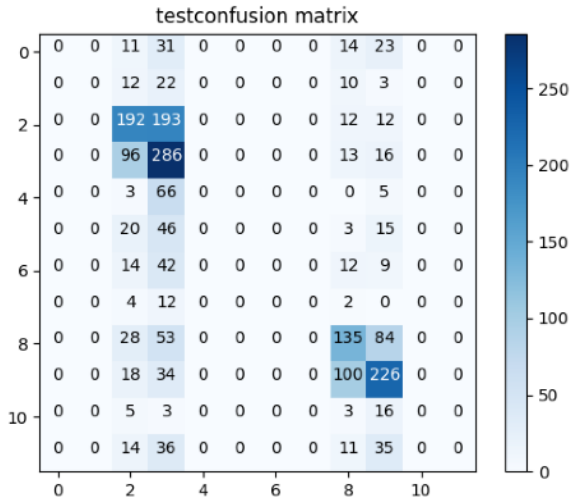
Used the same packages to train a 12 class SVM classifier.

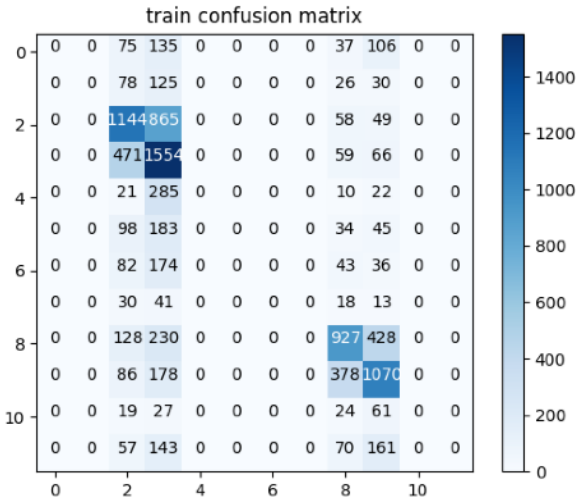
```
# read in features and labels
features_filename = "../data/car/train/results/train_data/train_feature.mat"
features_dict = scipy.io.loadmat(features_filename)
features = features_dict['train_feature']

labels_filename = "../data/car/train/results/train_data/train_label.mat"
labels_dict = scipy.io.loadmat(labels_filename)
labels = labels_dict['train_label']
labels = np.ravel(labels)

# fit models
print("started training\n")
print(datetime.datetime.now())
model = svm.SVC()
model.fit(features, labels)
print("finished training\n")
print(datetime.datetime.now())
```

Train accuracy: 0.7279

Test accuracy: 0.7040

assumption: if the predicted orientation is within 90 degree around true orientation, it is a correct prediction.





d):

In this part, we download the development kit form KITTI. By using the information of lables, we can get information about the 2d bounded box's location. Then we write line on the graph to form a rectangle box around each detected car. Since we store the rotation-y in each object, we can use the rotation-y to calculate the orientation of the car. Finally we project this orientation into the image plane.

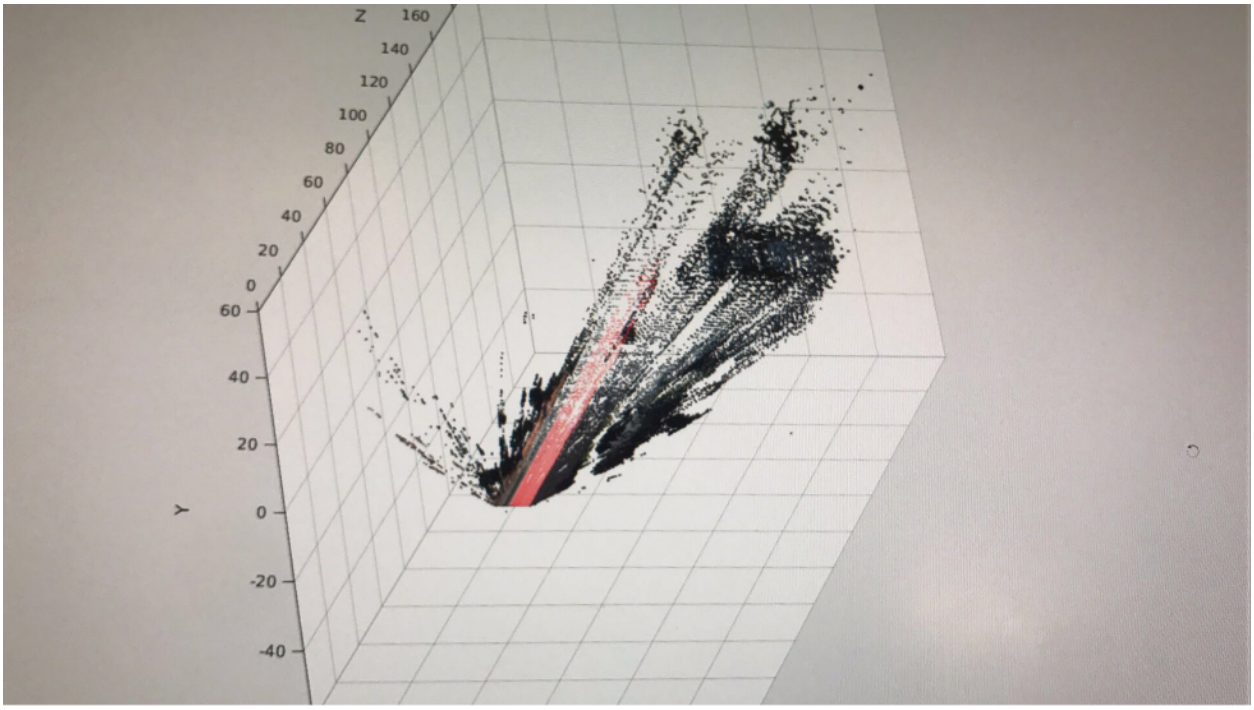
```
function drawBox2D(h,object,orientation)

% set styles for occlusion and truncation

% draw regular objects
if ~strcmp(object.type,'DontCare')

  % show rectangular bounding boxes
  pos = [object.x1,object.y1,object.x2-object.x1+1,object.y2-object.y1+1];
  rectangle('Position',pos,'EdgeColor','g','LineWidth',3,'parent',h(1).axes)
  rectangle('Position',pos,'EdgeColor','b', 'parent', h(1).axes)
end
%   % draw label
%   label_text = sprintf('%s\n%1.1f rad',object.type,object.alpha);
%   x = (object.x1+object.x2)/2;
%   y = object.y1;
%   text(x,max(y-5,40),label_text,'color',occ_col{object.occlusion+1},...
%        'BackgroundColor','k','HorizontalAlignment','center',...
%        'VerticalAlignment','bottom','FontWeight','bold',...
%        'FontSize',8,'parent',h(1).axes);
% end

 % draw orientation vector
if ~isempty(orientation)
line([orientation(1,:),orientation(1,:)]+1,...
     [orientation(2,:),orientation(2,:)]+1,...
     'parent',h(1).axes,'color','w','LineWidth',4);
line([orientation(1,:),orientation(1,:)]+1,...
     [orientation(2,:),orientation(2,:)]+1,...
     'parent',h(1).axes,'color','k','LineWidth',2);
end
```

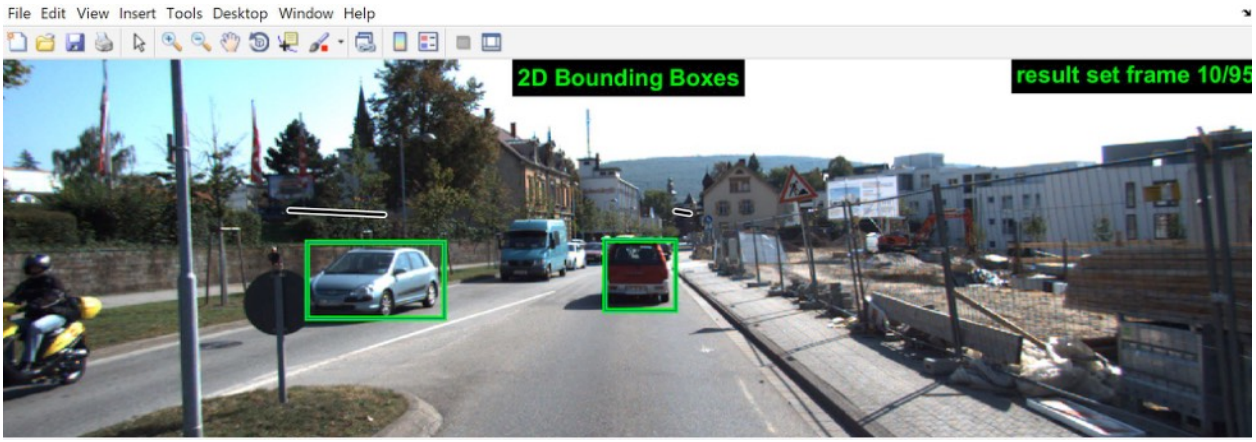
Q3

We treated each image as one frame of film and set the framerate as 4 to make a film of detected road

```
workingDir = '../data/test/results/road_detect_results_300/';
imageNames = dir(fullfile('../data/test/results/road_detect_results_300/','*.png'));
imageNames = {imageNames.name};
outputVideo = VideoWriter('road_detect_out.avi');
outputVideo.FrameRate = 4;
open(outputVideo)
for ii = 1:length(imageNames)
    imname = imageNames{ii};
    img = imread(fullfile(workingDir,imname));
    frame = im2frame(img);
    writeVideo(outputVideo,frame);
end

close(outputVideo)

shuttleAvi = VideoReader('road_detect_out.avi');
ii = 1;
while hasFrame(shuttleAvi)
    mov(ii) = im2frame(readFrame(shuttleAvi));
    ii = ii+1;
end

f = figure;
f.Position = [150 150 shuttleAvi.Width shuttleAvi.Height];

ax = gca;
ax.Units = 'pixels';
ax.Position = [0 0 shuttleAvi.Width shuttleAvi.Height];

image(mov(1).cdata,'Parent',ax)
axis off
movie(mov,1,shuttleAvi.FrameRate)
```

Results:


Z
160
140
120
100
80
60
40
20
0
60
40
20
0
-20
-40
Y


File Edit View Insert Tools Desktop Window Help
2D Bounding Boxes
result set frame 10/95

Challenges:

When tried to use text file to store training data computed from Matlab, a memory error occurred because the file is too large. Then used "scipy.io" to directly load and save mat file directly in Python.

For part 2, viewpoint classification. The accuracy was around 0.5 by using only HOG features and SVM. Increased accuracy to 0.7 by adding HSV features.

Reference:

Yamaguchi, Koichiro, David McAllester, and Raquel Urtasun. ”Efficient joint segmentation, occlusion labeling, stereo and flow estimation.” European Conference on Computer Vision. Springer International Publishing, 2014 Paper and code.

Mohan Karthik. “Feature extraction for Vehicle Detection using HOG+”.https://medium.com/@mohankarthik/feature-extraction-for-vehicle-detection-using-hog-d99354a84d10