Vehicle Detection Project

The goals / steps of this project are the following:

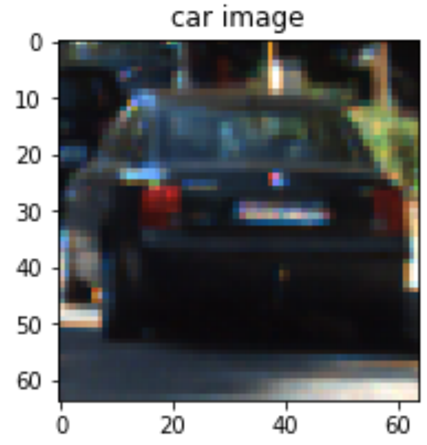
- Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images and train a classifier Linear SVM classifier
- Optionally, you can also apply a color transform and append binned color features, as well as histograms of color, to your HOG feature vector.
- Note: for those first two steps, don't forget to normalize your features and randomize a selection for training and testing.
- Implement a sliding-window technique and use your trained classifier to search for vehicles in images.
- Run your pipeline on a video stream (start with the test*video.mp4 and later implement on full project*video.mp4) and create a heat map of recurring detections frame by frame to reject outliers and follow detected vehicles.
- Estimate a bounding box for vehicles detected.

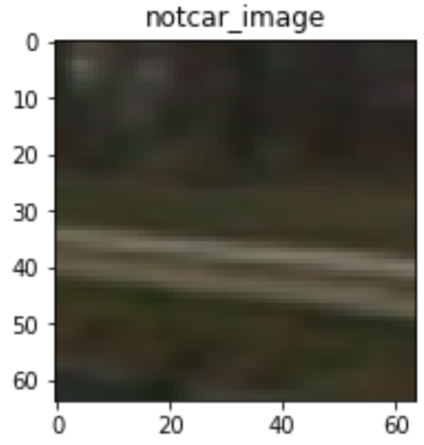
I have a notebook file with file name “visualization_test_codes.ipynb” as visualization and generation of images, “p5Utilities.py” file for all the utility functions use to train and create a pipeline, and “vehicle_detection.py” file as a main file.

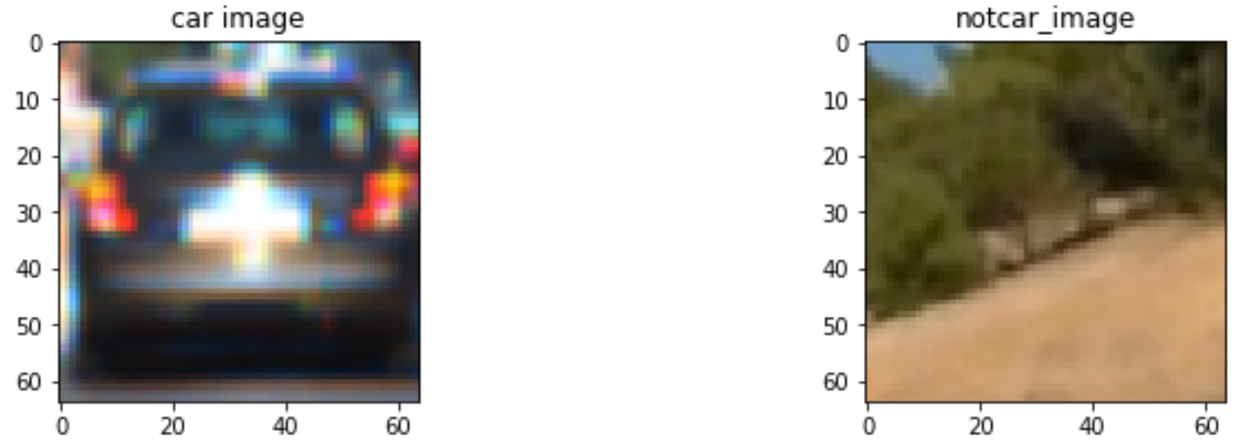
Histogram of Oriented Gradients (HOG)

The first cell of “visualization_test_codes.ipynb” file is to read in the training data from vehicles and non-vehicles data then listing the full path names of those data images in the cars.txt and notcars.txt file.

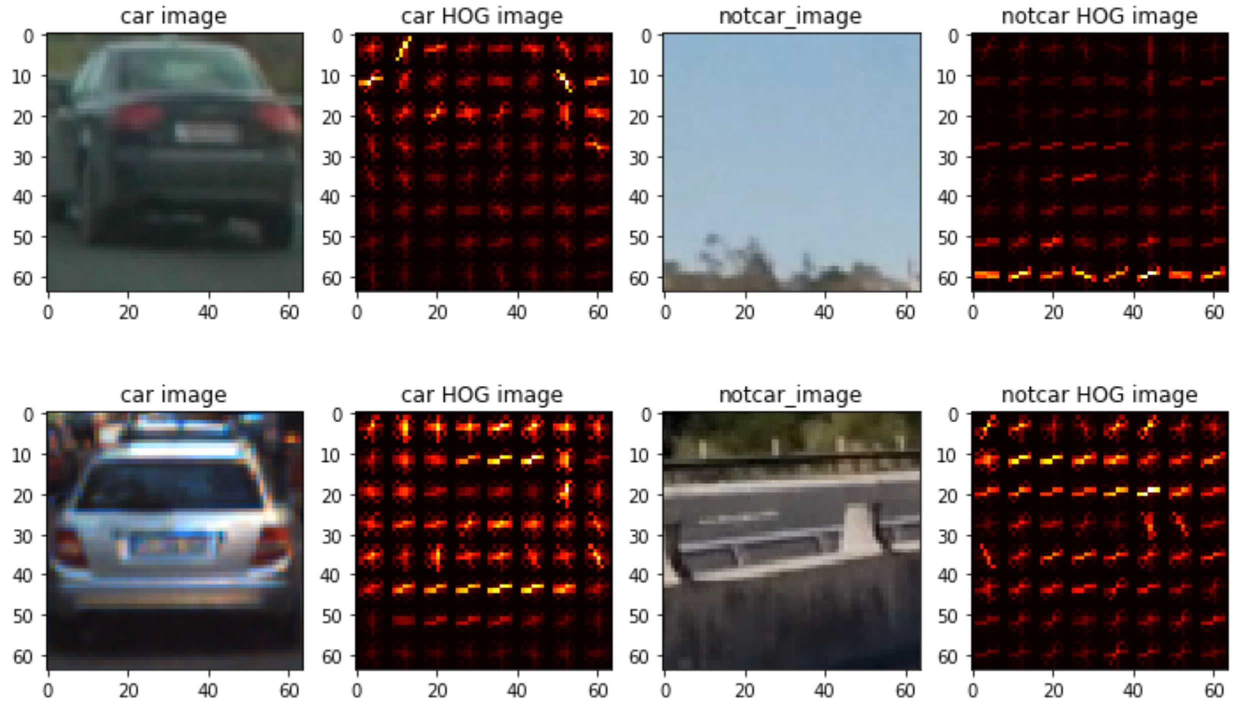
The third cell of the same file is trying to read in example cars and not cars images randomly to visualize those data. I ran the cell a couple times to get an idea on how those cars and not cars images look like. Below are the images generated and placed under “output_images” folder in this project repository.







I used the “get_hog_features()” function which wraps skimage.hog() and spatial binning feature and color feature functionality to visualize the HOG feature output images look like by running the cell fifth cell of the “visualization_test_codes.ipynb” file. Here are what output images. I settled with the following feature parameters for the project (YCrCb color space, orientations=9, pix_per_cell=(8,8), cell_per_block=(2,2)).



I have tried the following parameters for HOG features:
color space: RGB, HSV and LUV and settled with YCrCb. To me I can visualize the YCrCb HOG image better. I tried gradient orientation with 6, 8 and 9 which I settled and this orientation gave me better accuracy score when I trained the classifier later as well as pixels per cell of (8, 8) and cells per block of (2,2) seems to give me better accuracy score from the trained classifier.

Training and Using Other Features

I used the LinearSVC from Scikit Learn library to train my classifier. As from the documentation from Scikit learn, LinearSVC implements "one-vs-the-rest" multi-class strategy which seems to be appropriate for this project. (ref: http://scikit-learn.org/stable/modules/svm.html)

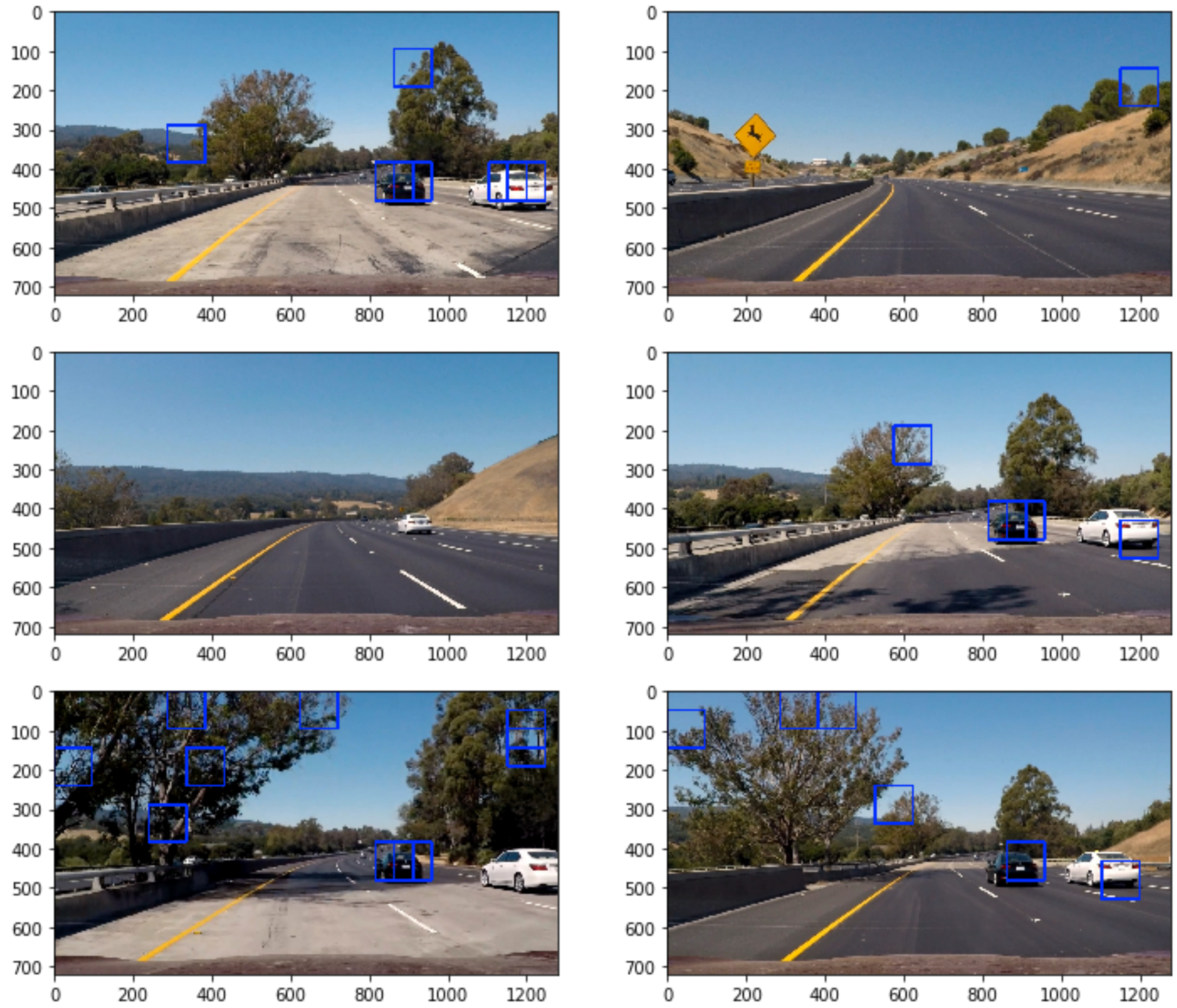
I also added the other features like spatial and color to combine with hog features to train my classifier. I have tested with different combinations of those three features with spatial size of (32, 32), histogram bins of 32 seems to give me the average accuracy score of 99%.

The sixth cell of the "visualization_test_codes.ipynb" file has all the functions that used as utility functions and the seventh and eighth cells have the LinearSVC training codes. I used the test_train_split function from Sklearn model selection package to split the training and test data with 10% test data. And I stored the classifier as classifier.pkl using Sklearn joblib.dump function.
Here is the output of training result with the time that spent on the training.

```
89.31323885917664 Seconds to compute features...
Using:  9 orientations 8 pixels per cell 2 cells per block 32 histogram bins
 (32, 32) spatial sampling
Feature vector length:  8460
15.62 Seconds to train SVC...
Test Accuracy of SVC =  0.9904
['classifer.pkl']
['scaler.pkl']
```

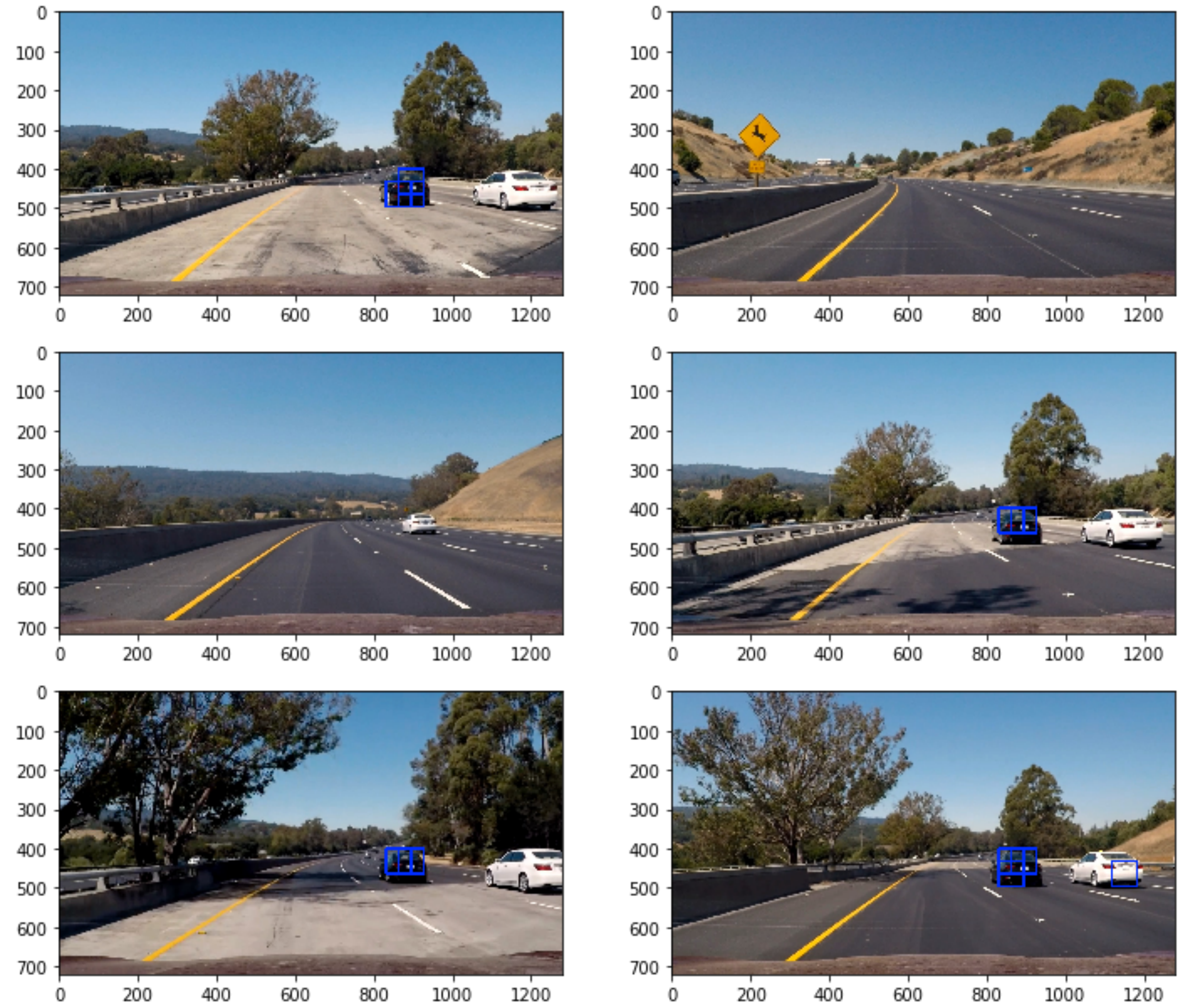
Sliding, Searching Window and Drawing Bounding Boxes

I defined slide_window(), search_window() and draw_boxes() functions in the cell ninth of the "visualization_test_codes.ipynb" file and run those functions on the test_images from the project repository to see whether look correct or not.
Later I adjusted these functionalities with heatmap method and labeling method discussed in Udacity lesson to get better quality of searching the cars on a given image.
I tried with different size of small window and different factor of overlapping window to see which one looks better and the first pass of this result with (96, 96) window and (0.5, 0.5) overlapping factor looks as following:

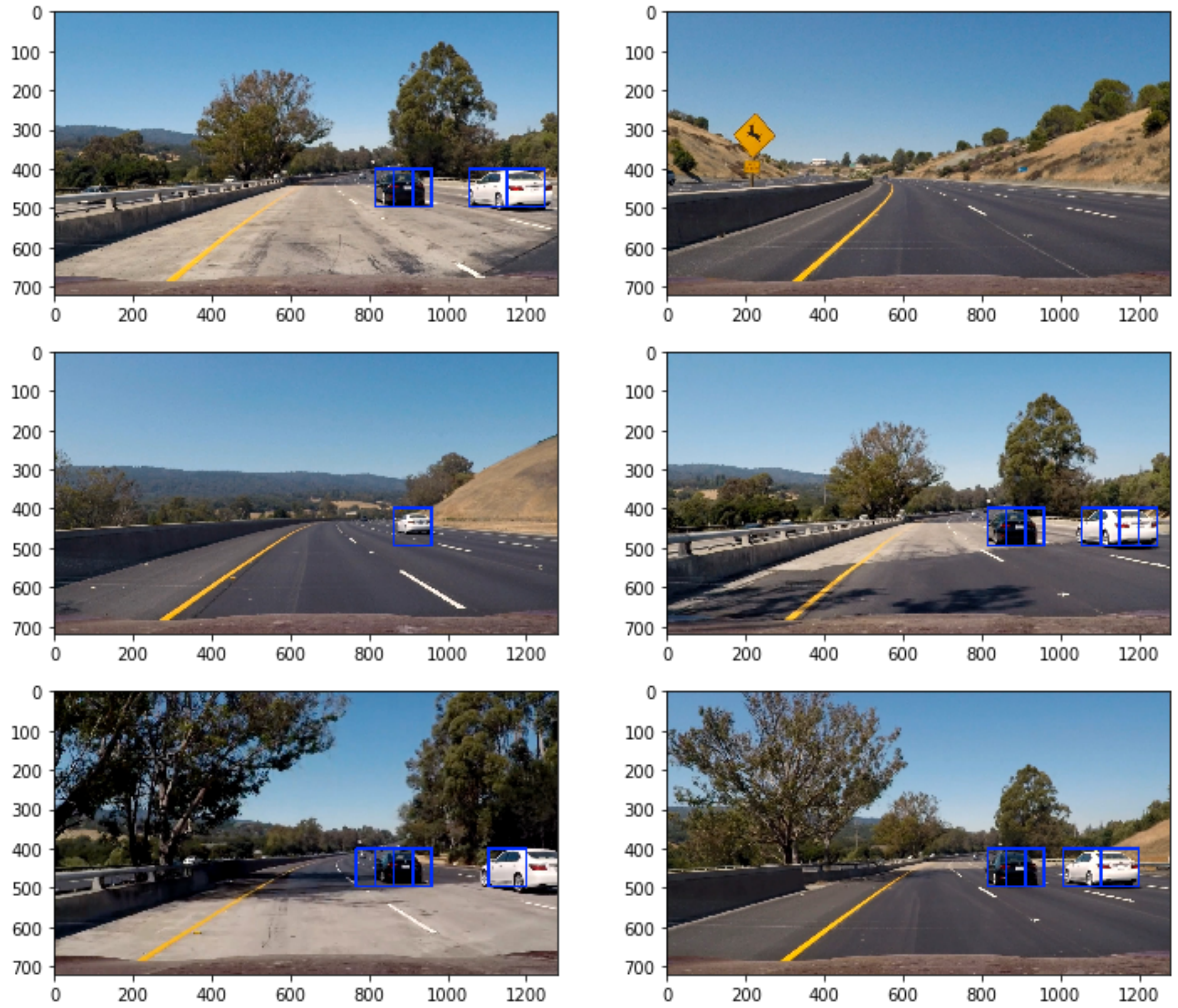


It has 100 small windows to go through per image and average of 1.5 seconds to process an image.

I tried different small window size and I chose to be 64 x 64 since all the cars and not cars images are in 64 x 64 so I think this is appropriate. And I chose the region of interest to be (0, 400) to (1280, 656). Which can be cut into multiple of 64 x 64 in y-direction. Here is what I got from using these two changes on the test images:



It turned out I missed finding some cars images as seen above. The process took around 1 second to process the image with 273 small windows per image. Then I changed my window size back to (96, 96) and region of interest to be (0, 400) to (1280, 656) and the third pass output looks as follow:



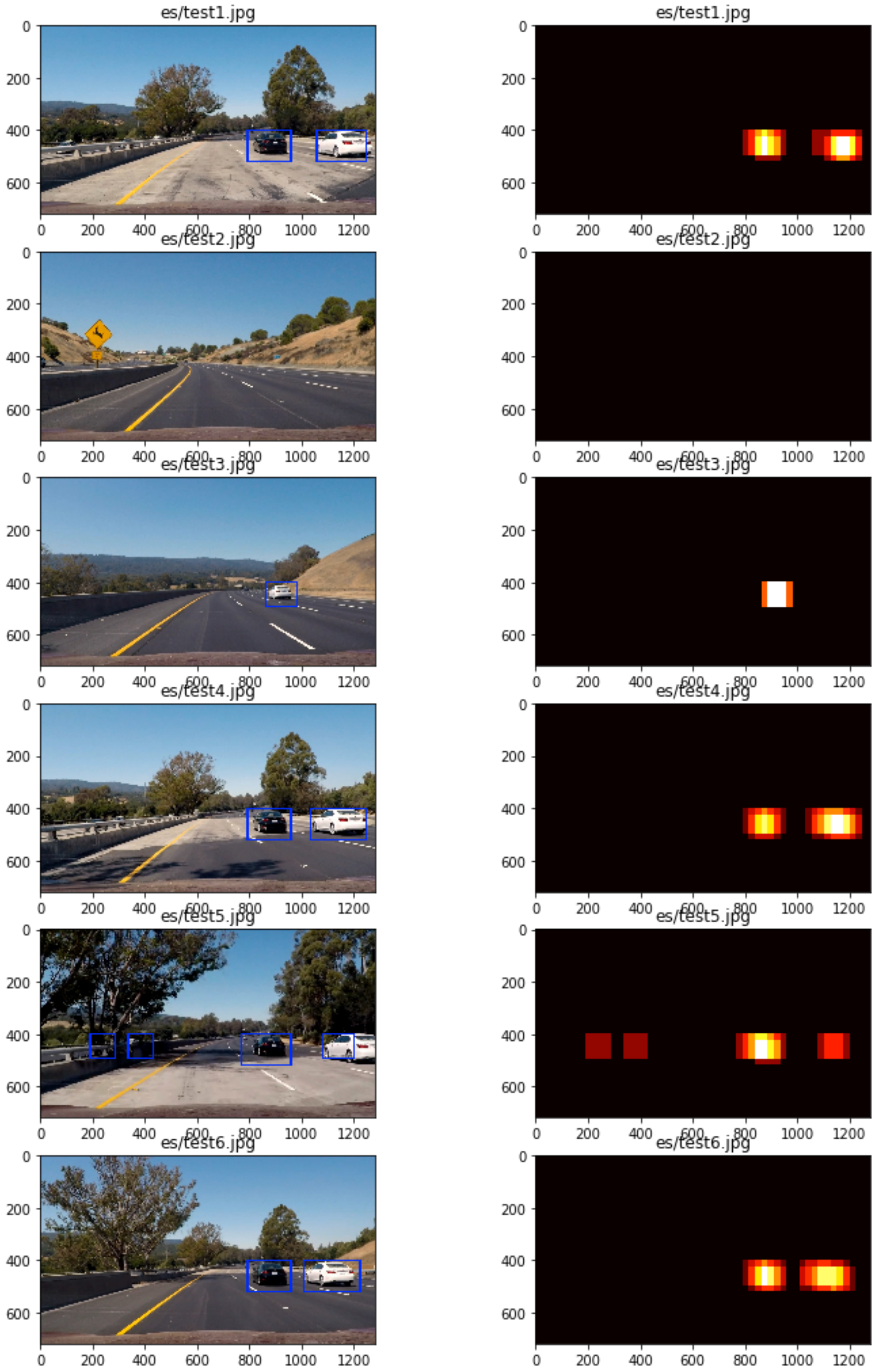
It looks good on these test images with several boxes on the same car and boxes are not drawn as bounding box but they are drawn on the cars.

All above three tests use the hog feature, spatial feature, color feature with color space in YCrCb. While I am testing the search window function I used the parameters to extract feature the same as while I was training it. I used all three features (i.e. hog feature, spatial feature and color feature).

I know I have to improve with scaling window, heatmap, thresholding and labeling technique that discussed in Udacity class as well as all the discussions around slack channel and forum.

I implemented these above methods in the cell twelfth of the file "visualization_test_codes.ipynb" with extracting the hog features not in each small window but in the whole entire image (region of interest) and only to convert the color for the region of interest. So basically, I tired to extract features in whole region of interest instead of small window by small window. Sliding through the small window size of 64 x 64 through like above test passes to find the cars in the image. I used the techniques that discussed on the YouTube – Self-Driving Car Project Q&A | Vehicle Tracking by Ryan Keenan.

The code that I implemented this is in the cell number eleven of the "visualization_test_codes.ipynb" file.


es/test1.jpg
es/test1.jpg
es/test2.jpg
es/test2.jpg
es/test3.jpg
es/test3.jpg
es/test4.jpg
es/test4.jpg
es/test5.jpg
es/test5.jpg
es/test6.jpg
es/test6.jpg
0
200
400
600
0
200
400
600
800
1000
1200

Video Implementation

After I generated the “test_video.mp4” file with above implementation with the pipeline implementation from the cell 14th to 18th of the “visualization_test_codes.ipynb” file.
You can see the video out file for “test_video.mp4” and “project_video.mp4” on the same notebook.

I see a lot of frames having flashes of false-positive bounding boxes. I have then added to ignore the flashed false positive boxes by applying the threshold. Now I got the better quality of detecting the cars. I also use the label() function from scipy.ndimage.measurements.label as above to label to draw the boxes.

I implemented this function in cell number fifteen and sixteen. You can see the final result of output in this following YouTube link.

https://youtu.be/AyR78PN1zAw

For python files, I have “p5Utilities.py” and “vehicle_detection.py” as main file. I probably should separate training code and pipeline code. Since I persist the classifier and scaler using joblib.dump() from Scikit, it will not be problem separating it.

Discussion

Another interesting project with SVM Machine Learning technique.
I had some problems using HOG features between png and jpeg. It cost me quite a time to go back and read and watch the lesson to figure out the problem. Another big one is that stacking the different features together and reshaping the features. When I trained the classifier, I used to stack those three in the different order than when I run it. So, I was getting pretty bad accuracy.

The third one was which I tried to implement the dynamic scaling windows to find the cars on image as my vehicle moves. I ended up not using it as my implementation has several bugs and I needed more time to review the implementation. What I was doing is to find the depth of the image with respect to the camera, and then define the four different depths. Depends on the depth my slide window boxes that basically has four different sizes. The smaller size the larger depth is. That way every frame I just need to search for total of 26 windows instead of hundred windows. It probably saves times to process the video frames. But I did not get to get it working with this implementation method. One of the items that I need to work on when I have some times.

I also wonder what kind of output will give by using ConvNet. I persist the classifier and scaler to use when trying to predict it at real time. But seems like I cannot achieve what I implemented at real time performance. I also would like to explore in tracking like pedestrians and other objects on the roads as well as experimenting with other machine learning techniques.

There are some of the considerations I think I need to look into such as:
Should I need to care about opposite direction vehicles if there is block divider between the two opposite ways. What about using detection to calculate relative speed between the cars to predict the chances of having accidents. The list goes on and on.

This journey of being on this class is a joy, challenging yet gaining new skills. From the instructors of the class to project reviewers, mentor, students and the platforms like slack, forums help a lot. I will be using all the knowledge and skills that I learn from this class at my work.