Vehicle Detection Project

The goals / steps of this project are the following:

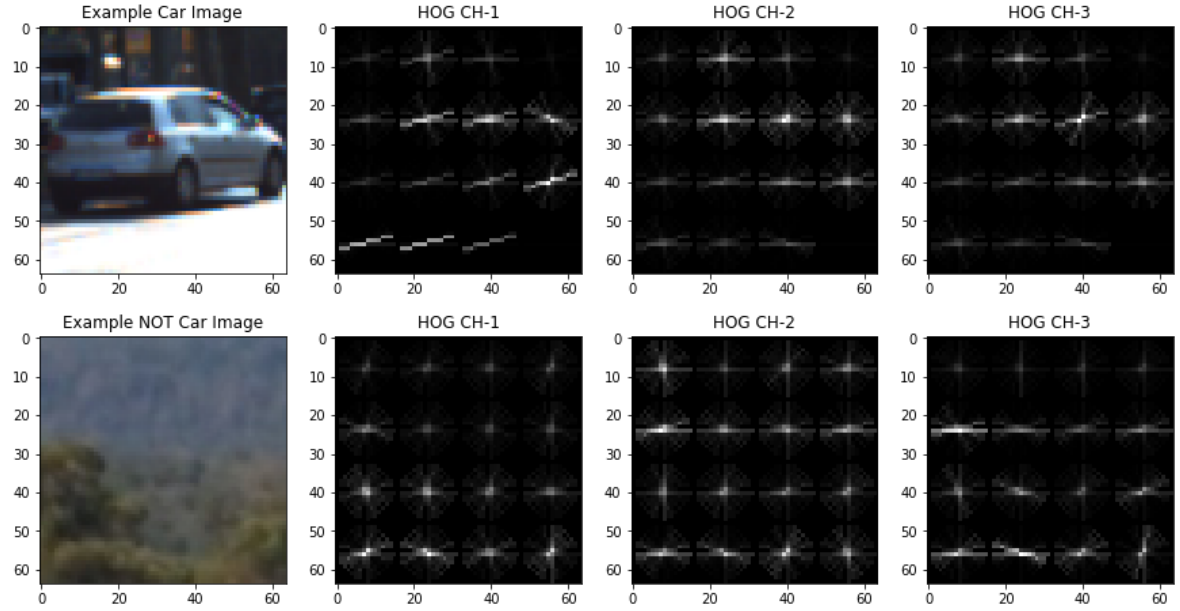
- Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images and train a classifier Linear SVM classifier
- Optionally, you can also apply a color transform and append binned color features, as well as histograms of color, to your HOG feature vector. Note: 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.

Histogram of Oriented Gradients

1. Explain how (and identify where in your code) you extracted HOG features from the training images.

I firstly took random images from both class (car, non-car). Below you can see the example images. (Cell 3)

Here is an example using the YCrCb color space and HOG parameters of orientations=16, pixels_per_cell= 16 and cells_per_block=2:



2. Explain how you settled on your final choose and your classifier

After doing some search on dataset started tweak with parameters before I dive into classifier. But It is hard to get an intuition by doing lots of changing until you see some accuracy result. That's why I would like to explain my classifier here together with features, which helps me to determine whether it is predicted as a vehicle or non-vehicle.

Although typical values are introduced for HoG orientations between 6 and 12 in Lesson, I found the best value, I mean accuracy on Test set, by choosing 16.

On the Other Hand YCbCr images are used instead of RGB Color images. So that my accuracy improved itself from 96% tom 98%. Parallel to this I tried to use different kernels such as Radial basis function (RBF). Thanks to RBF I increased my accuracy to about 99 %.

Below you can see my last parameter set until I realize in project Video that I am still finding false positives.

Parameter	**Value**
Orientation	16
HoG Channel	ALL → changed to 0 (thanks for the suggestion people from slack)
Pix_per_cell	(16,16)
Cell_per_cell	(2,2)
Spatial_size	(16,16)
Hist_bins	16

There are so many hyperparameters like the other Projects in term1. Some of them will be introduced in the next sections.

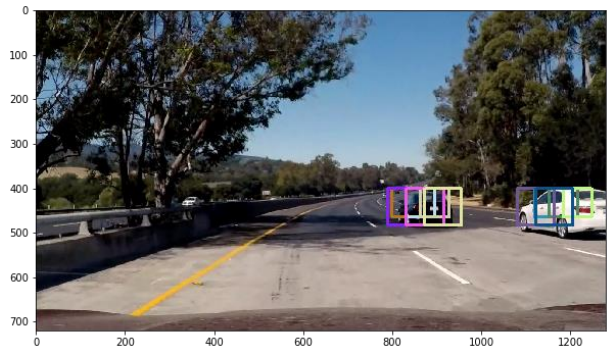
Sliding Window Search

1. Describe how (and identify where in your code) you implemented a sliding window search. How did you decide what scales to search and how much to overlap windows?

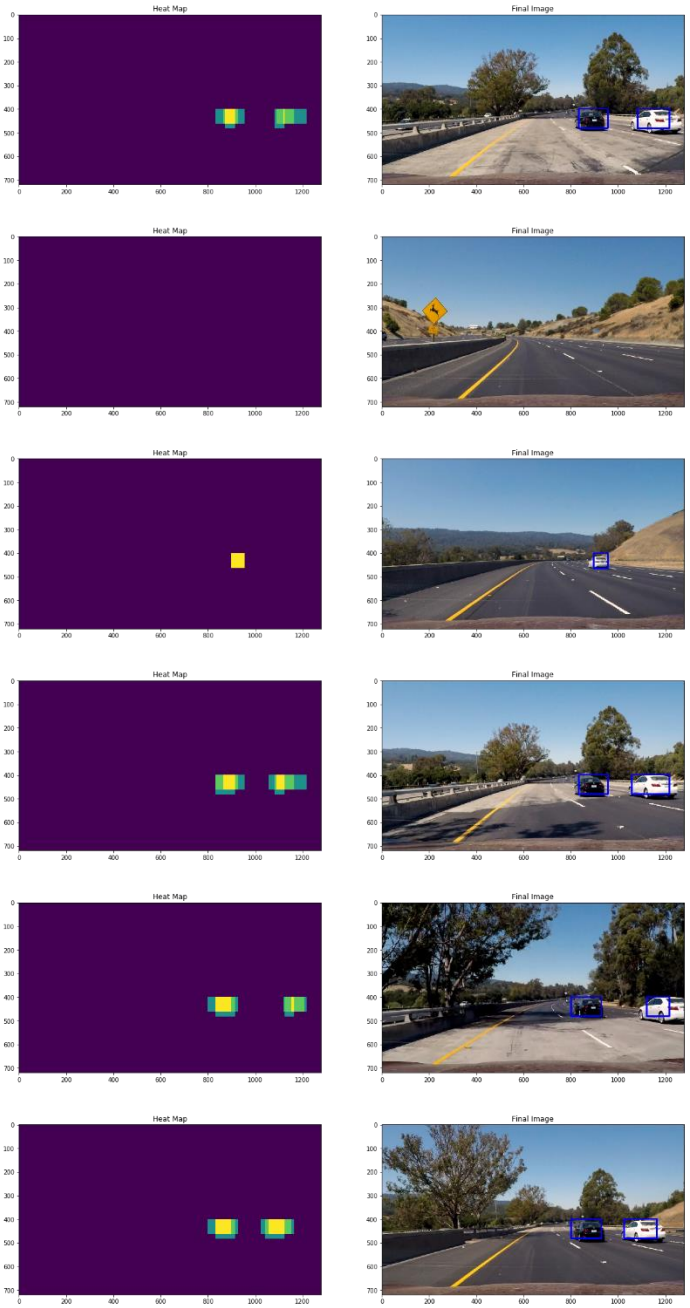
I decided to take various windows Scale between 1.0 and 3.0. Scale values and y_start-stop values were chosen by trial and error. The `find_cars` only has to extract hog features once, for each of a small set of predetermined scale argument), and then can be sub-sampled to get all of its overlaying windows. Each window is defined by a scaling factor that impacts the window size. The scale factor can be set on different regions of the image. See apply_sliding_windows in notebook

2. Show some examples of test images to demonstrate how your pipeline is working.

My final pipeline searched on different windows using the YCrCb 1-channel HOG features, plus spatially binned color and histograms of color in the feature vector, which provided a sufficient result. Here is an example image, which are prior to using heatmaps. After that I used heat map to reject false images.



In next figure you can see the result test images after using heat map to reject false images with threshold 1. See the function test_pipeline in notebook.



Video Implementation

You should see in Folder mp4 file called “project_video_out”. This is my final result video, after implementation. On the Other hand you can find merged version of Project 4 and this one “project_video_output_merged_P4P5”.

The code for processing frames of video is contained in the cell titled "Pipeline" From the previous 15 frames of video using the `prev_rects` parameter from a class called `VehicleDetect`. Rather than performing the heatmap/threshold/label steps for the current frame's detections, the detections for the past 15 frames are combined and added to the heatmap and the threshold for the heatmap is set to `9`. This was found by Trial and error. It would be better, if I didn’t choose such an empirical value.

Discussion

At first the threshold was chosen 9 to discard boundary boxes in oncoming traffic. Because they are on this type of road(highway) unnecessary. If I understood right, high threshold causes late detection and worse traction.

My Pipeline might not work under different lightning conditions or weather conditions. Thus, it cannot be used in real time. Another Problem is that It takes too much time i.e. about 15 min to produce the output video. Deep Learning approach could be a good start point.

Another classifier algorithms should be taken in consideration to compare with SVC. Because the first issue I ran into with this project was simply learning the best parameters to use, a common area of focus in machine learning.

In this project I am not satisfied with my results. I will definitely revisit to develop more robust and independent pipeline.