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.

General overview

The code for this project includes mainly the following files.

The heat.py is used to build the bounding boxes classified as cars, and heat-map is constructed.

The classifier.py is used to train a Linear SVM classifier on the labeled training set .

The feature_extraction.py combines features from histograms of color, spatial binning, and Histogram of Oriented Gradients (HOG), on the labeled training set of images to create a feature vector.

The util_functions.py includes some public functions which implemts color space convertion and feature computation.

The software pipeline is written in vehicledetection.py to detect vehicles in a video stream.

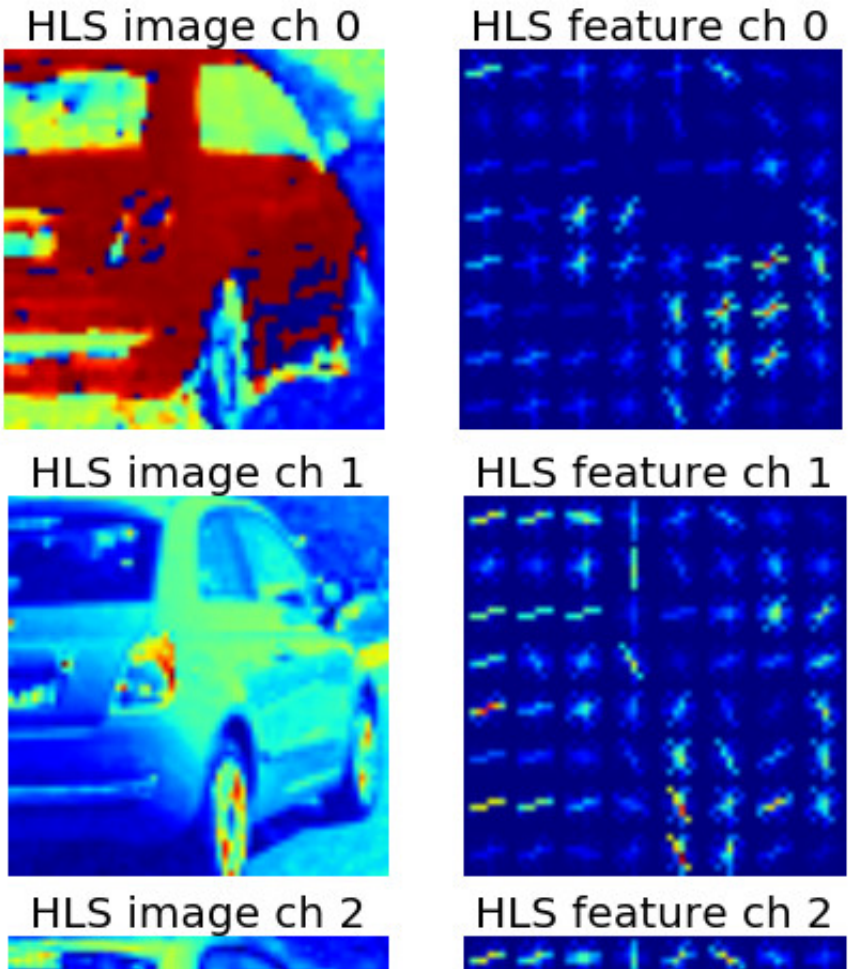
Histogram of Oriented Gradients (HOG)

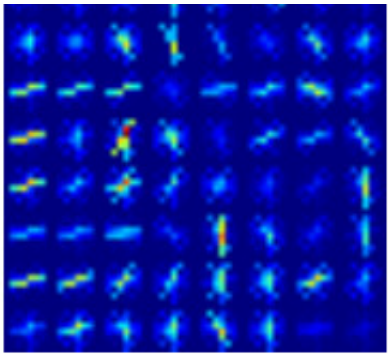
1. Explain how (and identify where in your code) you extracted HOG features from the training images.

The code for this step is contained in lines 27 through 49 of the file called `HOG.py` .

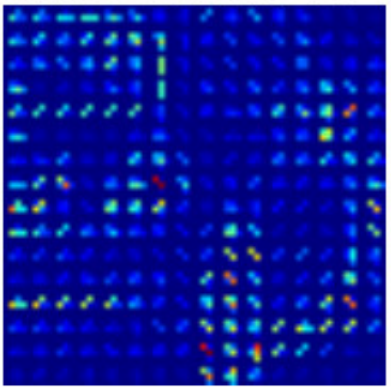
I started by reading in all the `vehicle` and `non-vehicle` images. Here is an example of one of each of the `vehicle` and `non-vehicle` classes:

I grabbed images from each of the two classes andI then explored different color spaces and different parameters (`orientations` , `pixels_per_cell` , and `cells_per_block`). Output looks like.

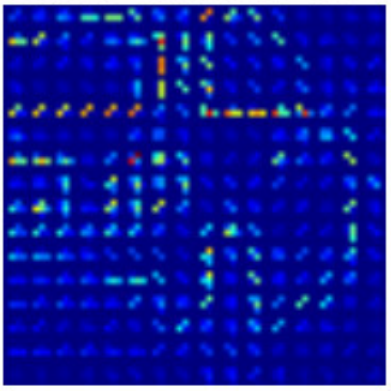


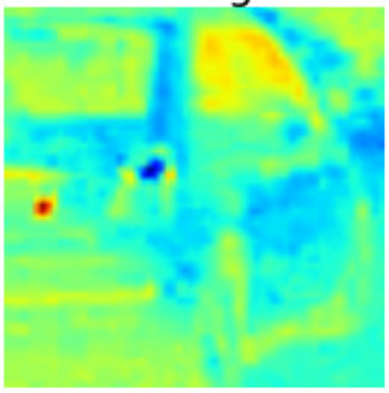
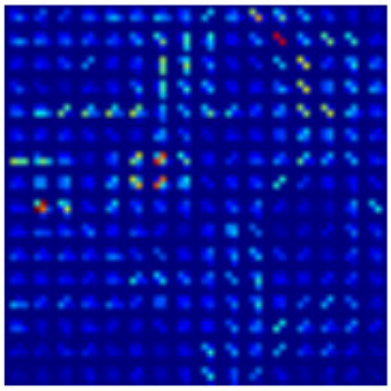
YCrCb image ch 0 YCrCb feature ch 0

YCrCb image ch 1 YCrCb feature ch 1

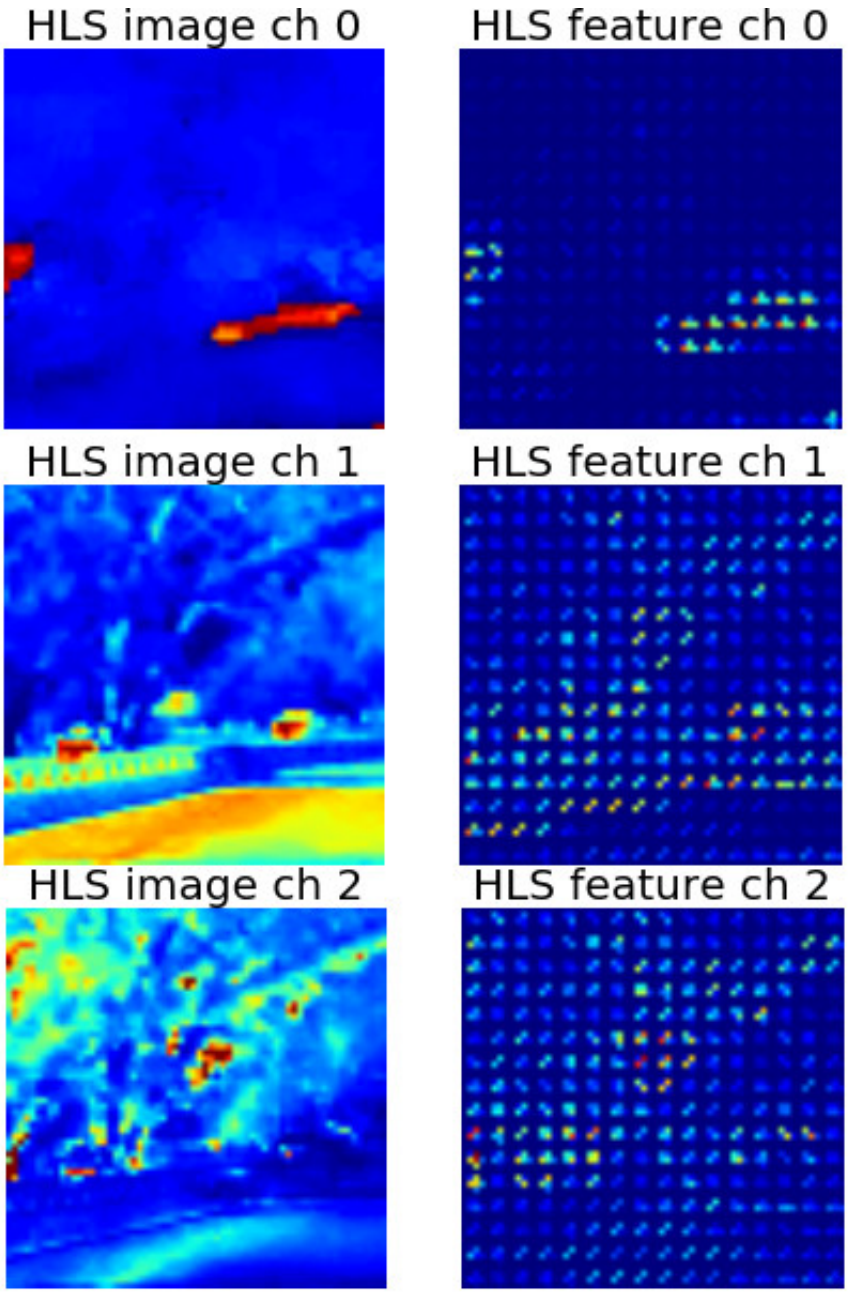



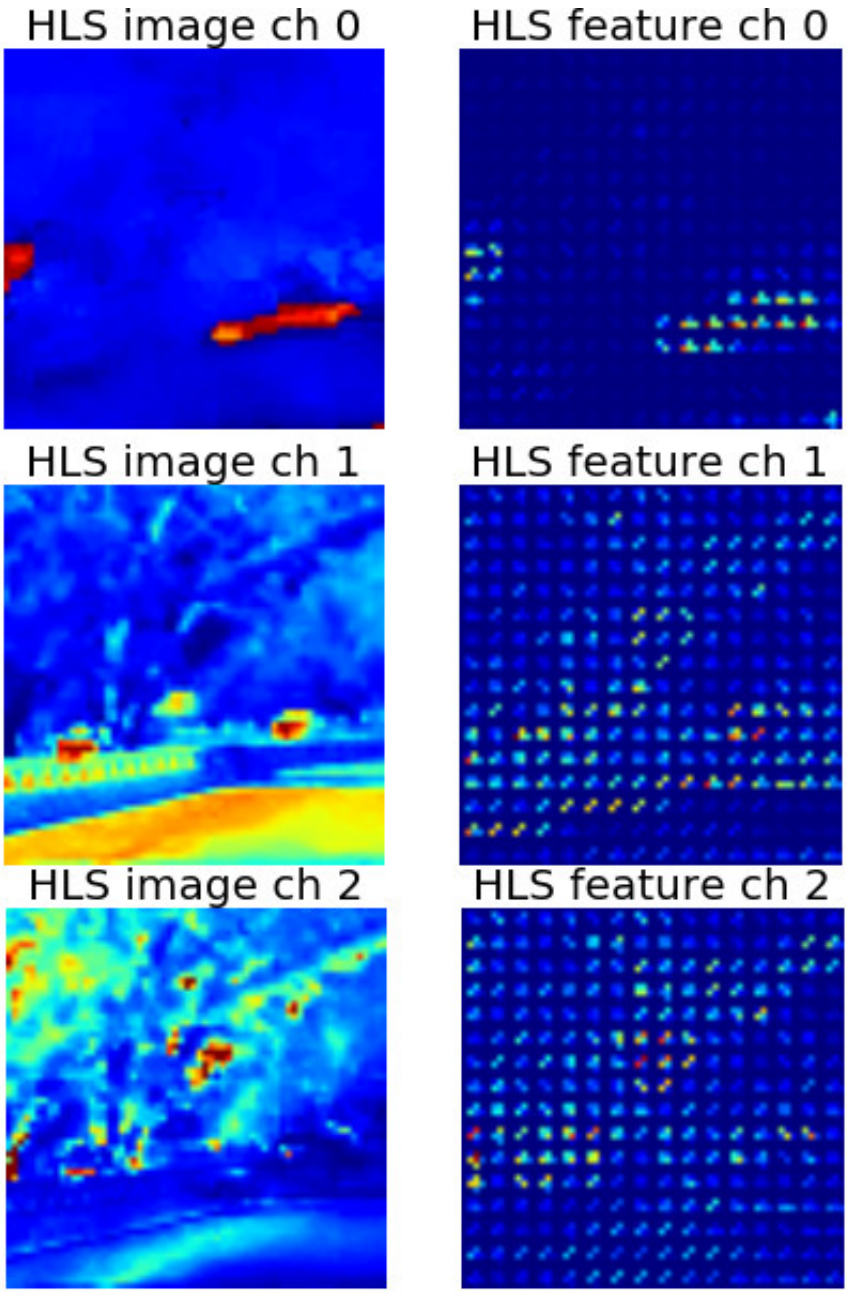
YCrCb image ch 2 YCrCb feature ch 2

car

(`HLS` color space and HOG parameters of `orientations=9` , `pixels_per_cell=(8, 8)` and `cells_per_block=(2, 2)` , `YCrCb` color space and HOG parameters of `orientations=12` , `pixels_per_cell=(4, 4)` and `cells_per_block=(2, 2)`)


HLS image ch 0
HLS feature ch 0
HLS image ch 1
HLS feature ch 1
HLS image ch 2
HLS feature ch 2



No car

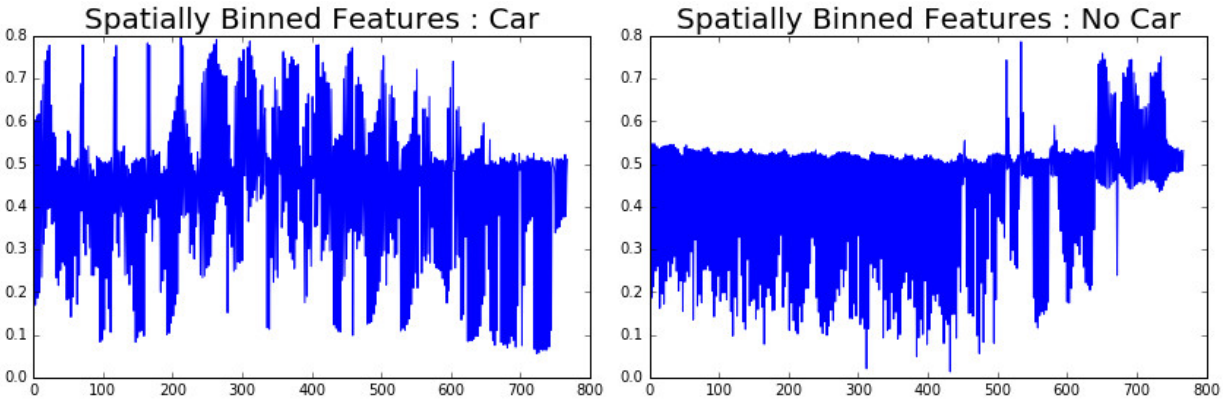
(`HLS` color space and HOG parameters of `orientations=9` , `pixels_per_cell=(8, 8)` and `cells_per_block=(2, 2)` , `YCrCb` color space and HOG parameters of `orientations=12` , `pixels_per_cell=(4, 4)` and `cells_per_block=(2, 2)`)

2. Explain how you settled on your final choice of HOG parameters.

I tried various combinations of parameters and I choosing orient = 12 pixels_per_cell=(2, 2) and cells_per_block=(8, 8) as HOG parameters. For the color space, to overcome the impact of shadow and accelerate processing of video, I chosed 3 channel of HLS.

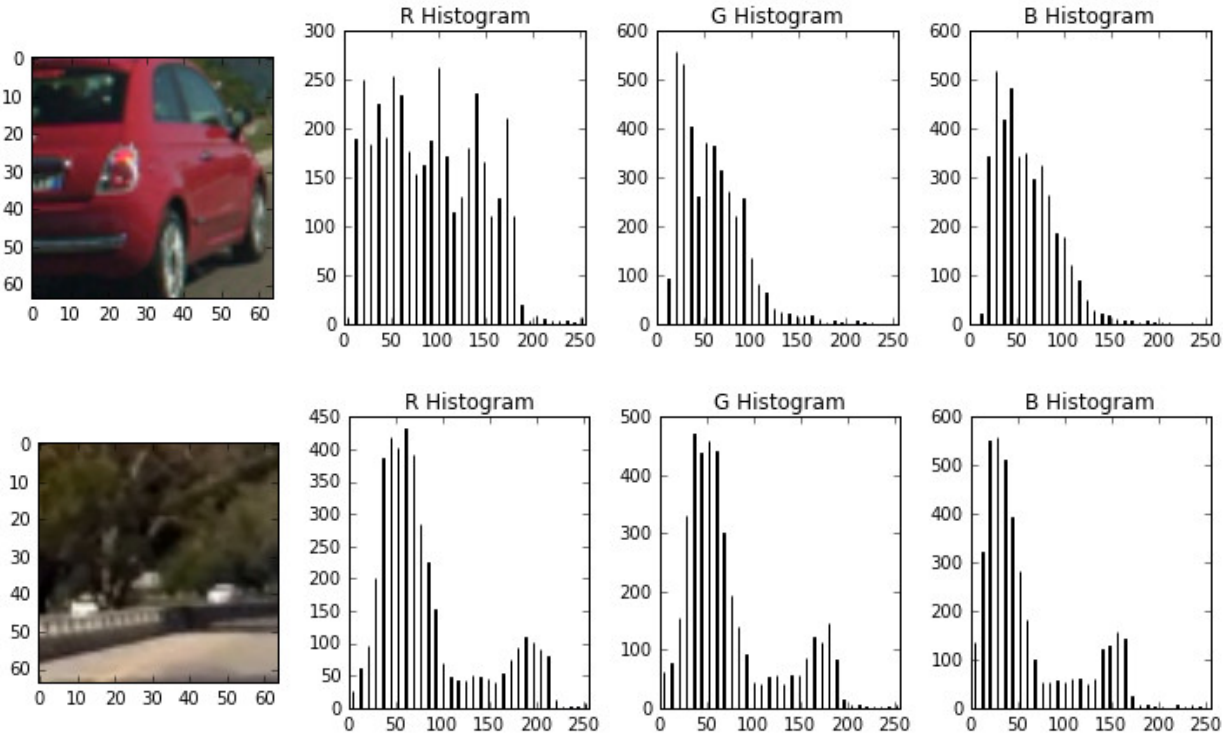
3. Other features: bin_spatial & color_hist

I also combined the spatial binning of raw pixel valuesbinary and histograms of color in features vector. I used a 16x16 resize for the binned spatial and 16 buckets for the color histogram procedure. Here an example:



Histograms of Color

Histograms of color are used to analyze the raw pixel intensities of each color channel as features in the image. Given an image in RGB color space, color histograms of features return a concatenated feature vector based on the number of intensity bins and pixel intensity ranges.



3. Describe how (and identify where in your code) you trained a classifier using your selected HOG features (and color features if you used them).

I trained a linear SVM using the fit method, and tested the accuracy with a test sample with 20% the size of the original dataset.
I use the dataset form the lesson(vehicle and non-vehicle images)), merge the 3 channels HOG with the bin_spatial and the color_hist features. The various features are normalized to have zero mean and deviation 1, by using StandardScaler preprocessor. To avoid overfitting, I shuffled the data before trainning SVC classifier. On a consistent basis, the accuracy for this classifier was above 95%..

Sliding Window Search

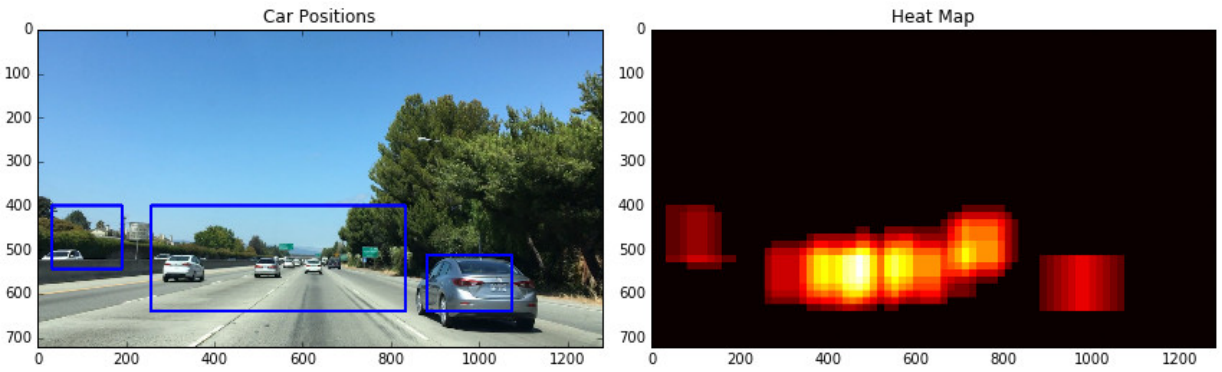
1. Describe how (and identify where in your code) you implemented a sliding window search. How did you decide what scales to search and how much to overlap windows?

I tried the scale varing from 1 to 2, I found that smaller scale can capture the vechile more accurately, but will cause more false positive, so its import to get a banlance between accuracy and remove flase_positive. I limited the search area to the bottom half of the image, minus the bottom 10%, for a total of 40% of the image to lower the total number of windows I needed to make. The parameters I chose for the sliding windows themselves are as follows:

Parameter	Value
y_start	350
y_stop	660
x_start	640
x_stop	1280
windows	(64,64)
cells_per_step	2

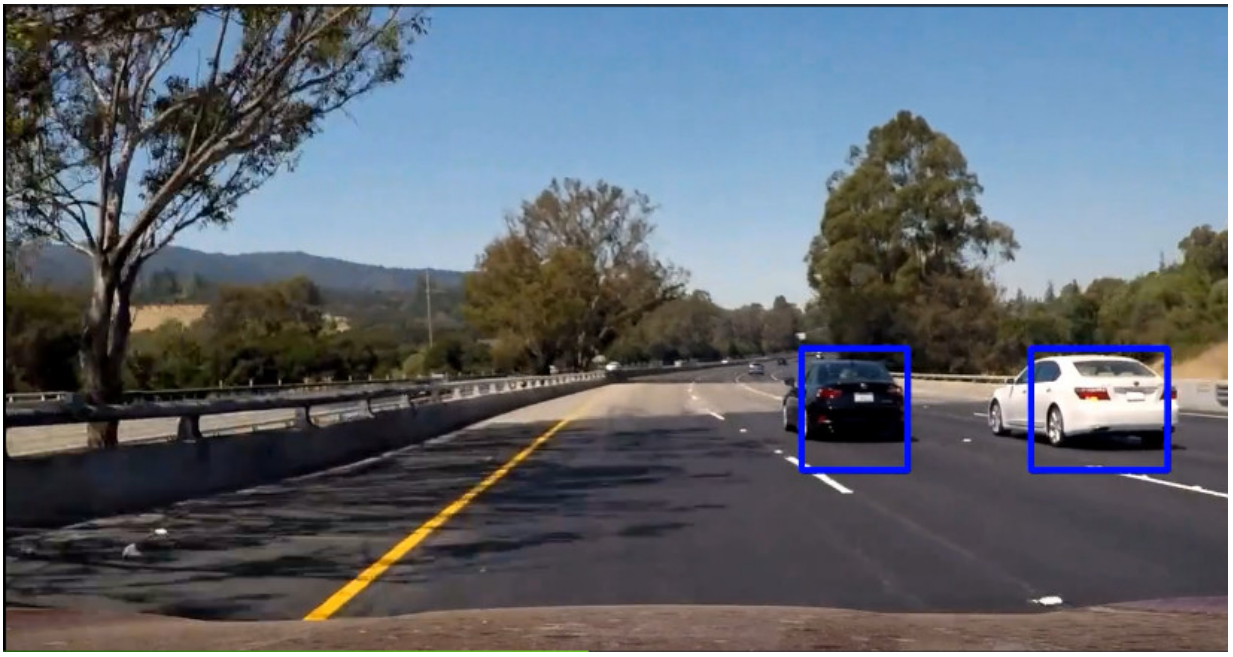
The sliding windows search is part of the vehicle_detection.py. It takes all the parameters given to the train method above, and proceeds doing a sliding window lookup, over the picture, computing the HOG function only once, in the whole image, and reusing its slices to get the window features.
The procedure returns a list of rectangle positions where it found positive car matches.

After recording the positions of positive detections in each frame of the video, I created a heatmap and thresholded it to identify vehicle positions.



2. Show some examples of test images to demonstrate how your pipeline is working. What did you do to optimize the performance of your classifier?

Ultimately I used HOG features in HLS plus spatially binned color and histograms of color in the feature vector, which provided a nice result. Here are some example images:



Video Implementation

1. A link to the final video output for this project is provided below. The code pipeline performs reasonably well on the entire video.

Here's a link to my video result : new_test_video.mp4

2. Describe how (and identify where in your code) you implemented some kind of filter for false positives and some method for combining overlapping bounding boxes.

In this vidoe, the car appear in the right of screen, so I implement search in this region. And because vechile differ with the background in relative speed, so I compute the average of hetmap of successive fram with a decay factor 0.8. I also chose a threshold base on the value of heatmap to remove some false_positive . I implement this procedure in `vehicledetection.py` (127 through 188)

Discussion

1. Briefly discuss any problems / issues you faced in your implementation of this project. Where will your pipeline likely fail? What could you do to make it more robust?

Here I'll talk about the approach I took, what techniques I used, what worked and why, where the pipeline might fail and how I might improve it if I were going to pursue this project further.

Firstly, The biggest challenge with this project was constructing a pipeline to work on a video stream. I find there are so many combination of parameters, so,it is difficult to troubleshoot the parameters as much as I would have liked.

Secondly, the false_positive is a troubles in this project. I tried to use information in successive frame and chose thresold carefuly to reduce the false_positive and smooth the vehicle detections. I think using some more complex method such as clustering, desicion tree etc. will be helpful to distinguish the car from background more than just `lable` function .

Lastly, problem with this code:
The parameters are fine-tuned for the project video but I see there could problems with other videos.
Monitored area is small than actual image.
The captured box is jumping and actual size of the car is not captured well.

Future Plans
I think the classifier is the key for this project, to improve its performance I will try also one of the big datasets available, the ones provided by Udacity seems big enough and have additional labeling. I also would like to test a CNN, I think the neural network should perform better in this case.