Writeup Template

You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.

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.

Rubric Points

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Writeup / README

1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. Here is a template writeup for this project you can use as a guide and a starting point.

You're reading it!

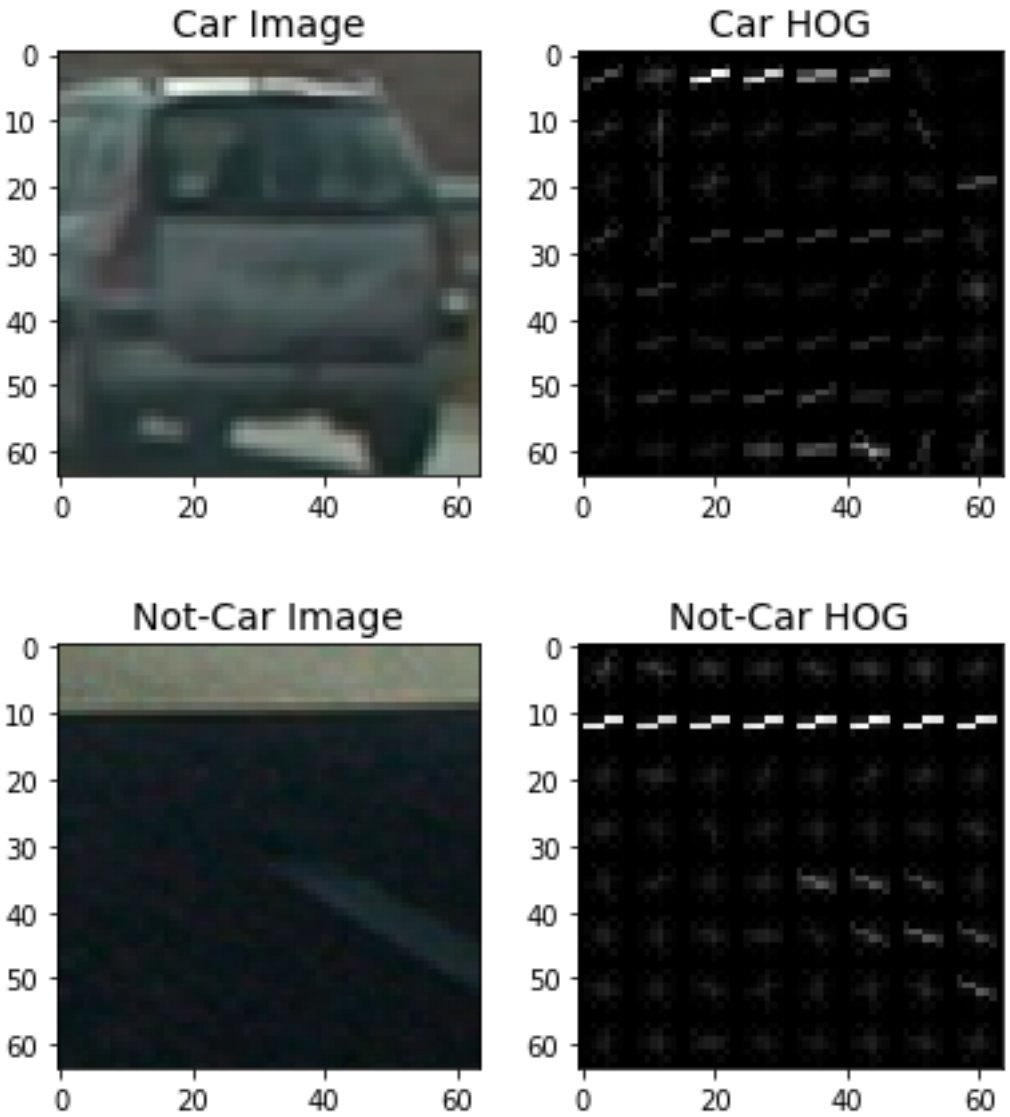
Histogram of Oriented Gradients (HOG)

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

I load vechicle and non-vehicle image from dataset. you can see random sample as below (code src/project.py line 17-38)

I then explored different color spaces and different `skimage.hog()` parameters (`orientations`, `pixels_per_cell`, and `cells_per_block`). I grabbed random images from each of the two classes and displayed them to get a feel for what the `skimage.hog()` output looks like.

Here is an example using the `YCrCb` color space and HOG parameters of `orientations=9`, `pixels_per_cell=(8, 8)` and `cells_per_block=(2, 2)`:(code src/project.py line 40-72)



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

I tried various combinations of parameters. Different colorspace and diff hog channel is my first try.(code src/project.py line 110-193)

Config 7 is good enough accuracy to next step. Extract time is not good enough. but for off-line video compute, I think no performance issue.

The table (Table 1)as below have 8 different parameter combinations that I explored.

Table 1

Config	Colorspace	Orientations	Pixels Per Cell	Cells Per Cell	HOG Channel	Extract Time (Second)	Train Time (Second)	Accuracy
0	RGB	9	8	2	ALL	83.1	19.41	0.9721
1	HSV	9	8	2	1	40.53	6.08	0.9212
2	HSV	9	8	2	ALL	83.54	8.7	0.9809
3	HLS	9	8	2	0	34.89	5.82	0.9347
4	HLS	9	8	2	1	34.48	4.33	0.9654
5	HSV	9	8	2	2	34.64	4.28	0.9589
6	YUV	9	8	2	0	34.24	4.4	0.9595
7	YUV	9	8	2	ALL	90.84	7.28	0.9856

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 with the default classifier parameters and using HOG features alone (I did not use spatial intensity or color features) and was able to achieve a test accuracy of 98.56%.

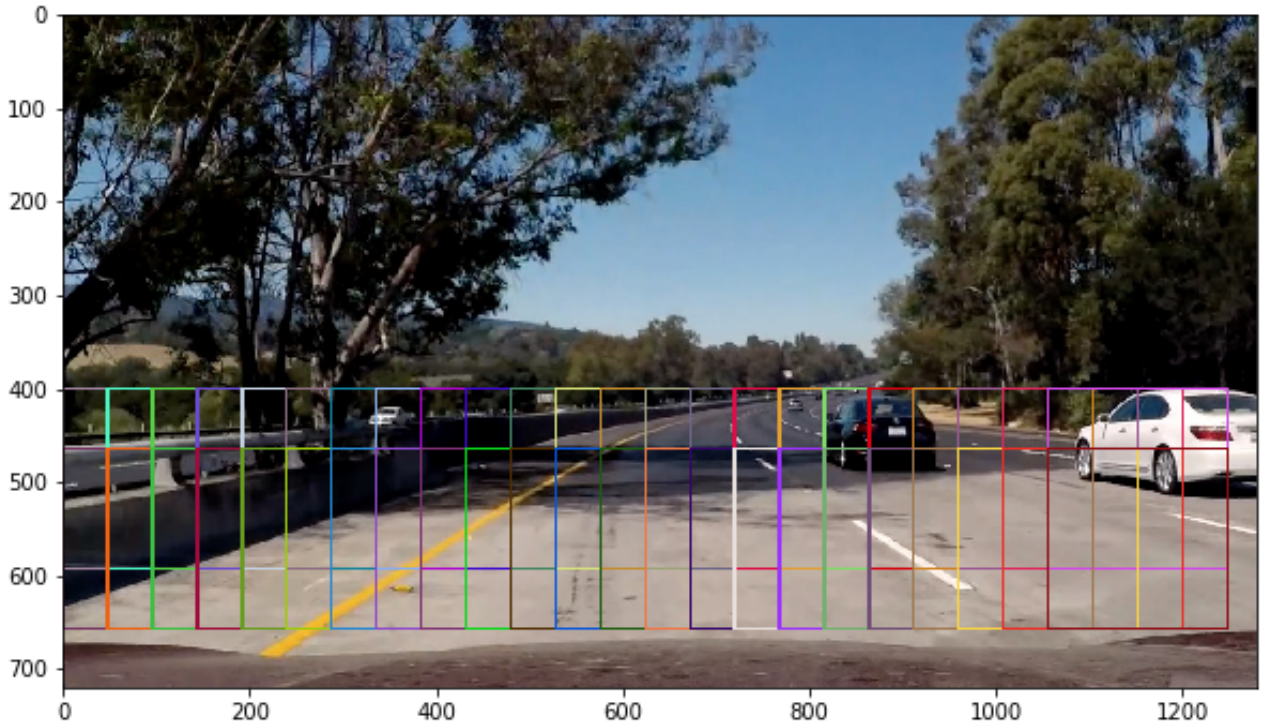
Sliding Window Search

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?

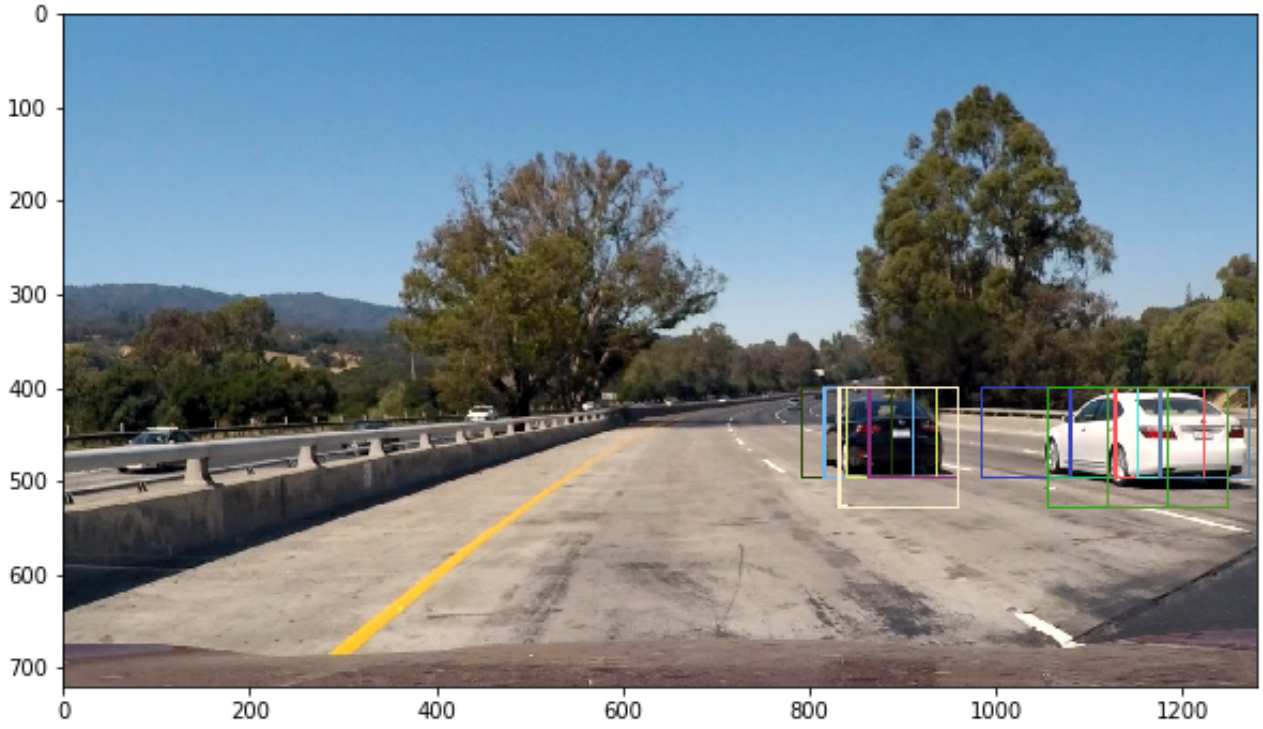
I decided to adapted the method find_cars from lesson materials. Combine HOG feature extraction with sliding window search. I try the sliding area ystart=400 and ystop=660. (code src/project.py line 197-310)

I explored various overlaps in X and Y directions. Below images is final implement for scale 1.0, 1.5, 2.0 and 3.x (code src/project.py line 314-355)

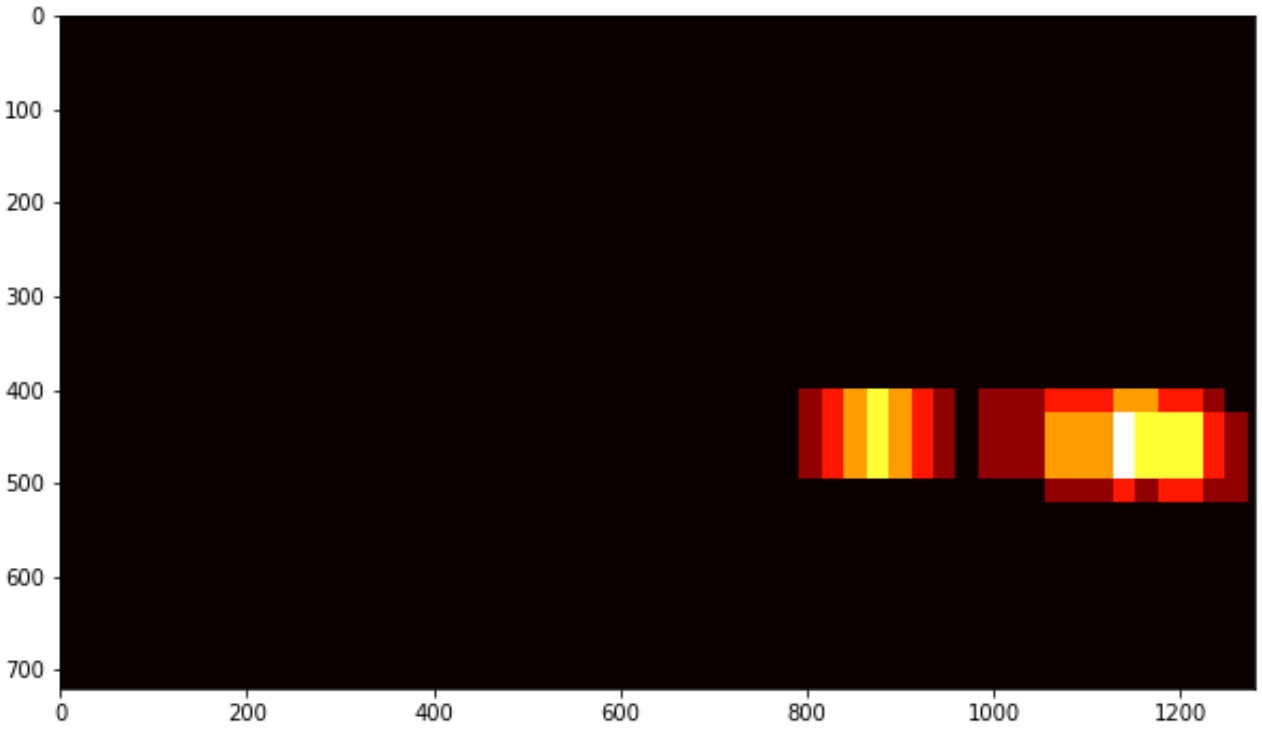

0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200



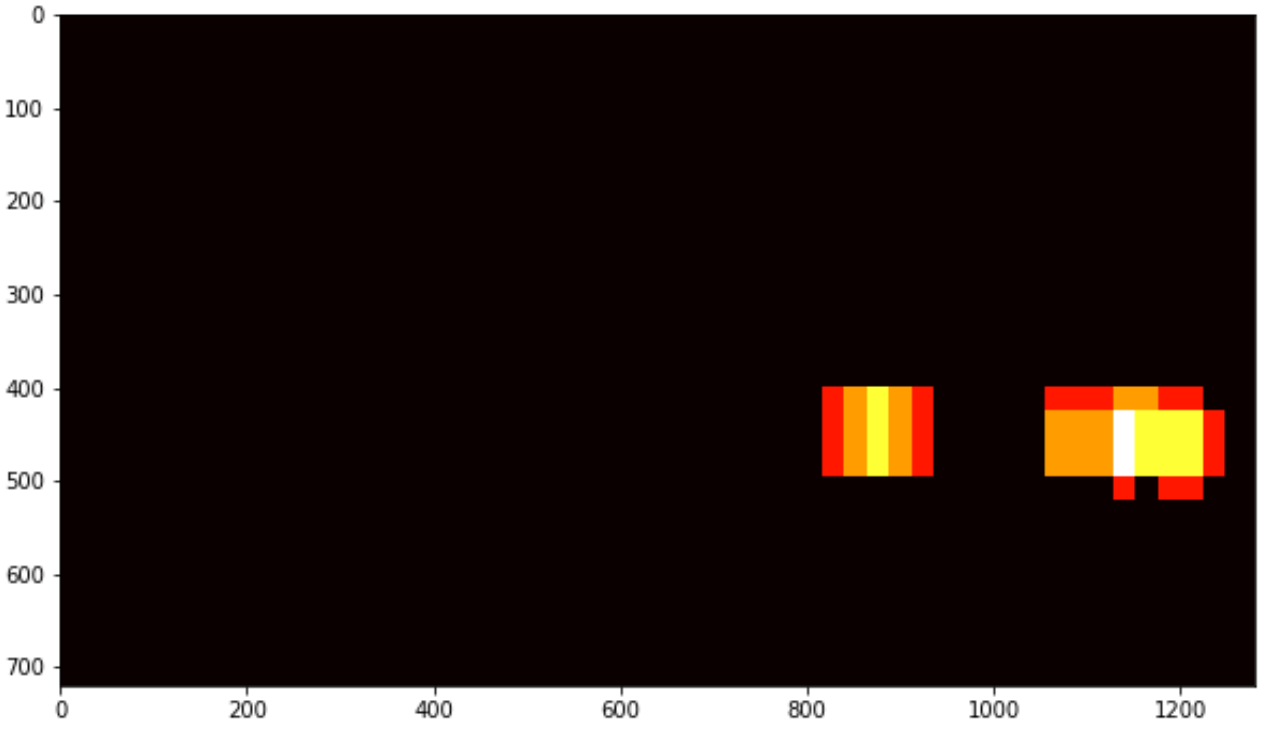
You can see the image as below that draw the rectangle with different sliding window and different scale.(code src/project.py line 308-310)



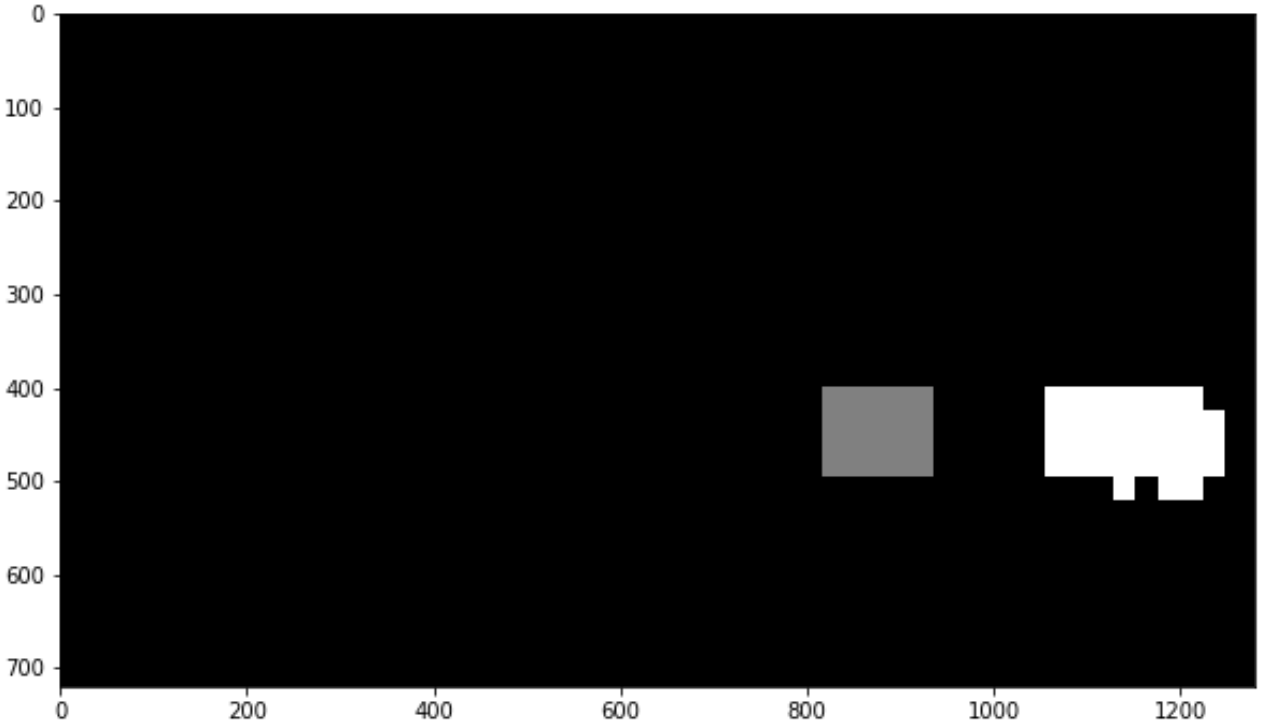
And I use heatmap to create a image as below.(code src/project.py line 468)



I use the threshold to increments the pixel value that reduce noise and higher level of heat. Below Image is the result of apply threshold.(code src/project.py line 469)



The `scipy.ndimage.measurements.label()` function collects spatially contiguous areas of the heatmap and assigns each a label:



And the final detection area is set to the extremities of each identified label:

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?

Results of project test images thought pipeline as below table.(code src/project.py line 418-477)(modify input image file name from 1 to 6)

test1	test2
	
test3	test4
	
test5	test6
	

The final result performs very good, it can identifying the near field vehiles for every test image without false positives.(code src/project.py line 418-477)

Video Implementation

1. Provide a link to your final video output. Your pipeline should perform reasonably well on the entire project video (somewhat wobbly or unstable bounding boxes are ok as long as you are identifying the vehicles most of the time with minimal false positives.)

Here's a link to my video result

https://youtu.be/5oCHHMc_LB8

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.

I use heatmap to solve false positive problem. and apply threshold to it.

In the video stream, I use previous frame to create heatmap and threshold. To solve false positive problem.

(code src/project.py line 489-497)

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?

I learned many techniques in this project. Extract feature, HOG, SVM, sliding windows, heatmap and assign each label.
If I have more time, I can make it more robust.
In HOG parameter part, I can discover more configuration.
In sliding window part, I can try more window size and scale.
In SVM part, I can adjust parameter or add color features.
In false positive part, I can try how many video frames to create heatmap that help robust.