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!

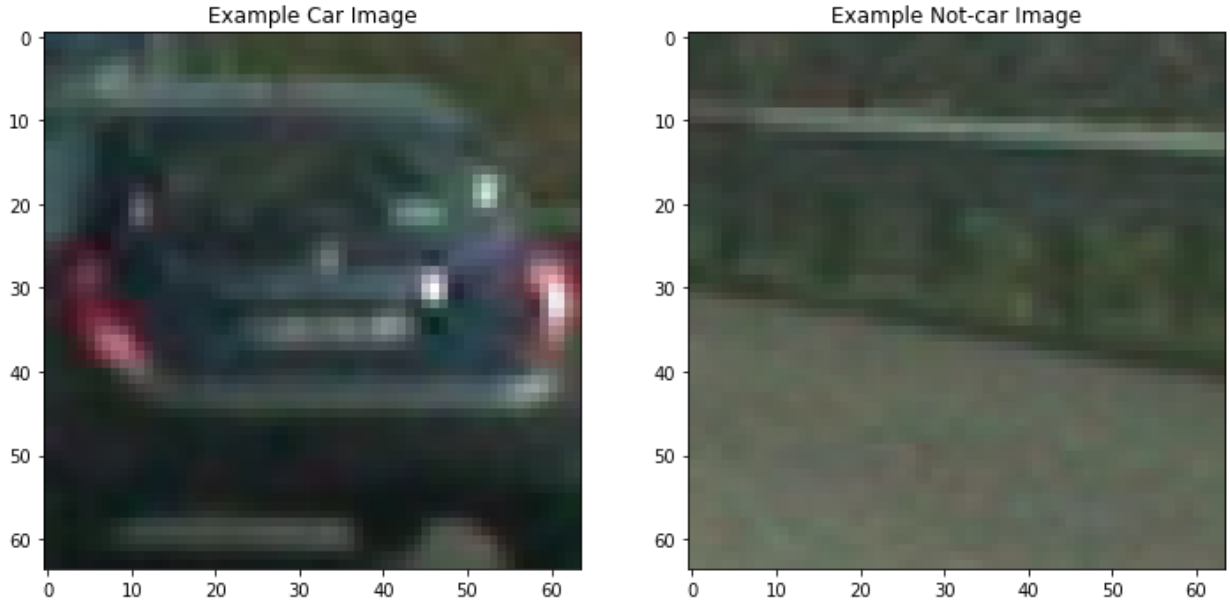
###Histogram of Oriented Gradients (HOG)

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

The code for this step is contained in the third code cell of the IPython notebook (core.IPynb).

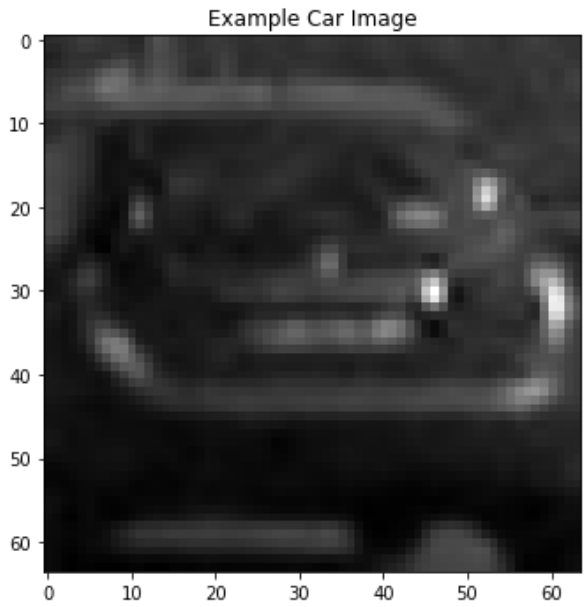
I started by reading in all the `vehicle` and `non-vehicle` images. I noted that:

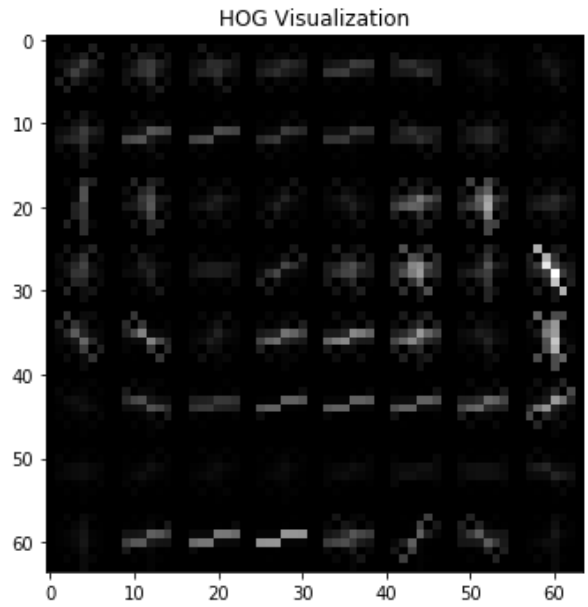
# of vehicle images:  8792
# of non-vehicle images:  8968

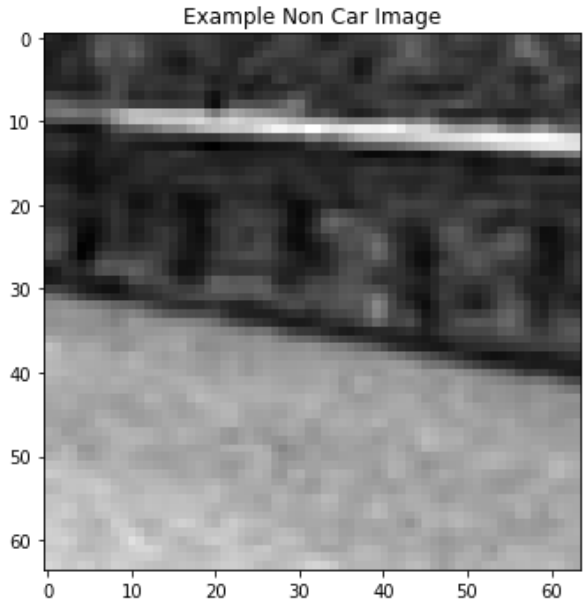


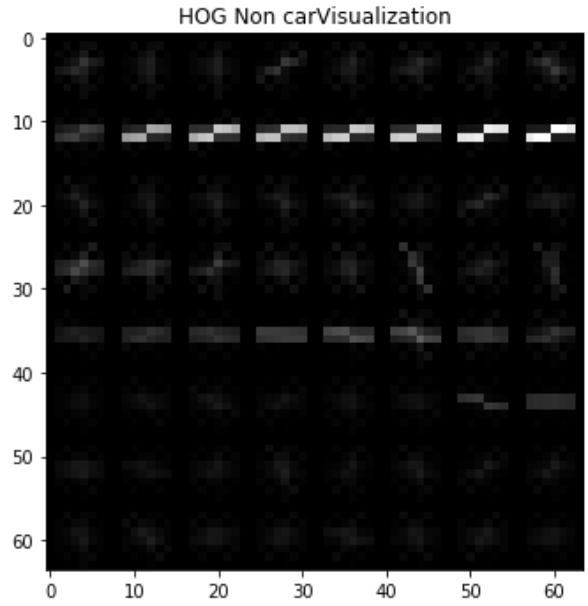

I also explored different `skimage.hog()` parameters (`orientations`, `pixels_per_cell`, and `cells_per_block`). I grabbed random images from each of the two classes and displayed them to get a feel for what the `skimage.hog()` output looks like. These are in code cells 4, 5 and 6

Here is an example using the `RGB` color space and HOG parameters of `orientations=9`, `pixels_per_cell=(8, 8)`and `cells_per_block=(2, 2)`:

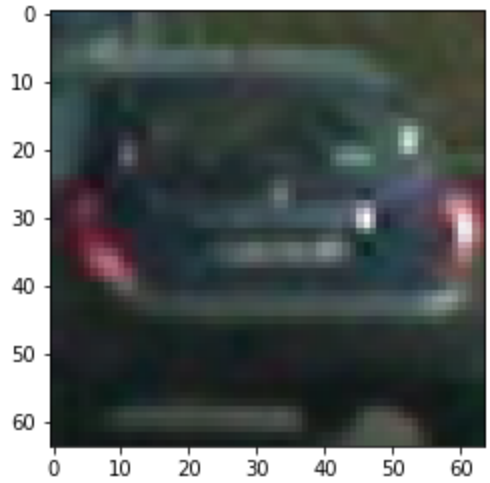


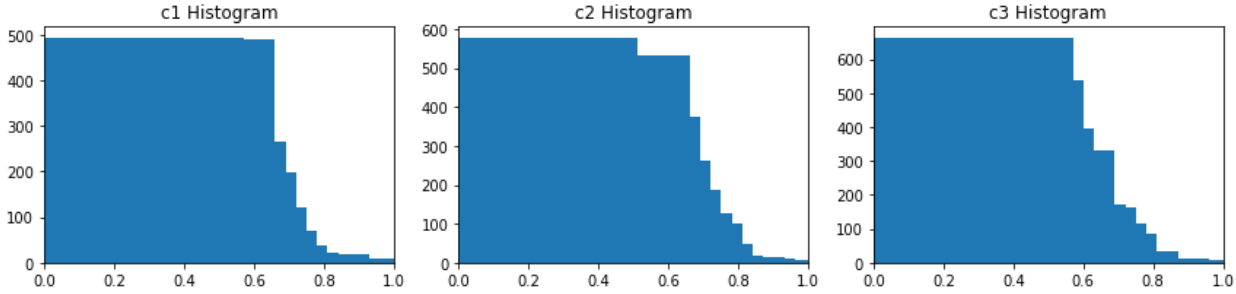



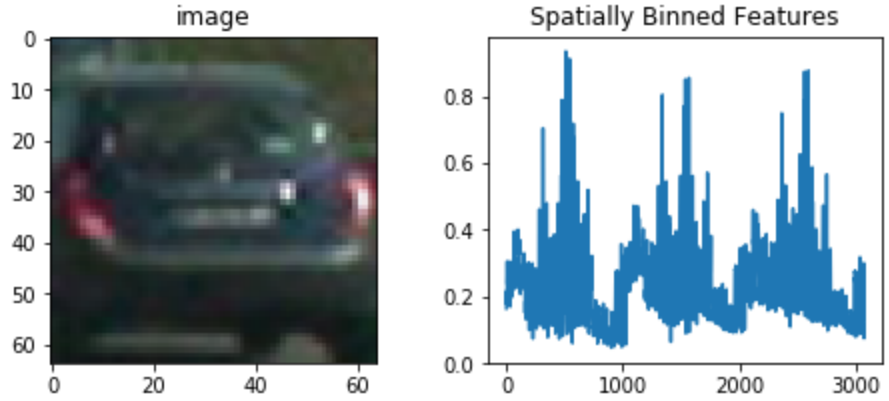




I also visualized the color histograms and the spatial bins (RGB):






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

I tried various combinations of parameters (code cell for all features on a sample of 1000 and then 2000 images from vehicles and non vehicles set. The best fit with over 99% accuracy was achieved with the following params:

color_space='YCrCb'

spatial_size=(32, 32)

hist_bins=32

orient=9

pix_per_cell=8

cell_per_block=2

hog_channel='ALL’

####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 a feature vector that combined spatially binned features, color histogram features and hog features (code cell 11) and got an accuracy of over 99% on the test set:

```
41.66 seconds to extract vehicle features...
40.87 seconds to extract non-vehicle features...
Feature vector length:  8460
22.3  seconds to train SVC..
Test accuracy of SVC:  0.9904
```

###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?

In code cell 12 I experimented with various search window and scale params.

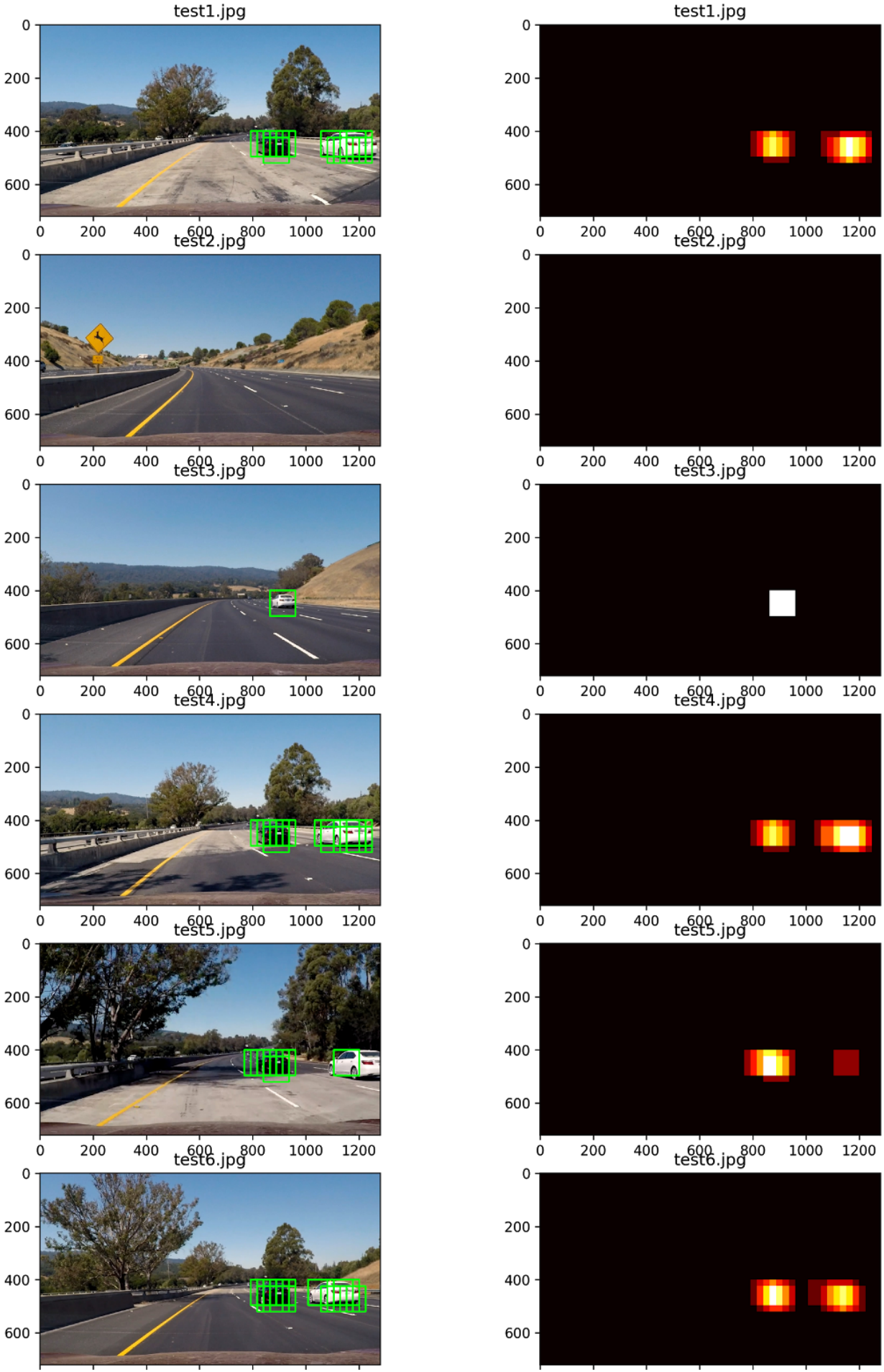
With overlap = 0.5 and xy_window as (64, 64) i got the following results on the test images:


It searched over 800 windows (some needlessly) and missed the white cars in several of these images and finds cars in tree. So next I edited the y crop to not include the tree and sky portion and set xy_window to 128. The results:

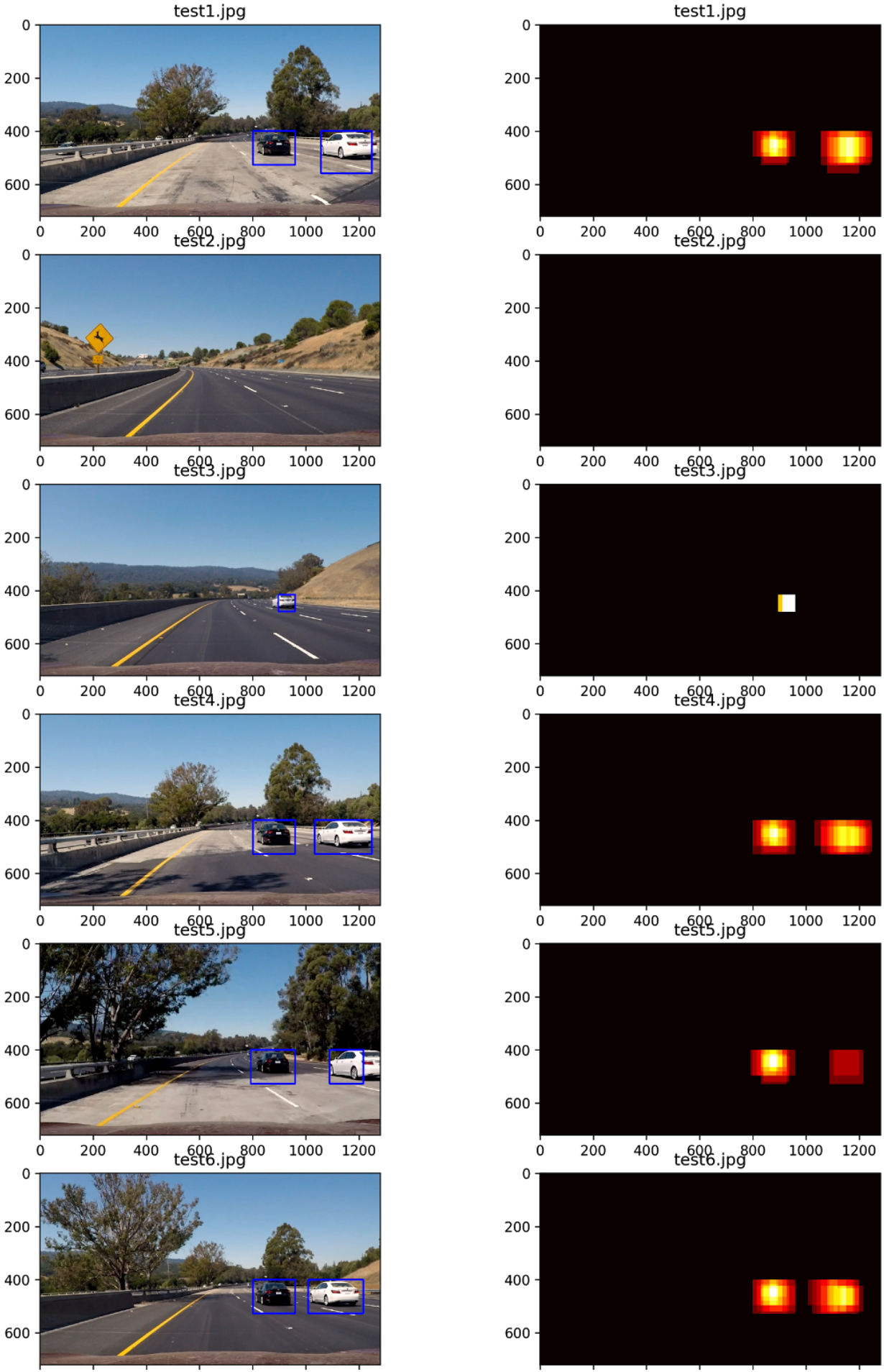

There are several false positives and it misses the cars that are far away (smaller scale)

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

Single scale (1.5) window search without thresholding and without using `scipy.ndimage.measurements.label()` to identify individual blobs in heat maps yielded these results:


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

Ultimately I searched on multi scales (1 to 3.5) using YCrCb 3-channel HOG features plus spatially binned color and histograms of color in the feature vector, which provided a nice result. I also added heat map calculations and thresholding the heat map for better results (cell 14). Here are some example images:


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

## 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

####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 recorded the positions of positive detections in each frame of the video. From the positive detections I created a heatmap and then thresholded that map to identify vehicle positions. I then used `scipy.ndimage.measurements.label()` to identify individual blobs in the heatmap. I then assumed each blob corresponded to a vehicle. I constructed bounding boxes to cover the area of each blob detected.

---

###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?

Using multi scale window search is a little slow and could be made faster by averaging over a few frames the position of a vehicle. But calculation over multiple scales does rule out almost all false positives. Another possible speedup is to search using smaller scale values near the horizon (just below) y values and larger as we get closer to the car.