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 are in the right place. This is the writeup.

###Histogram of Oriented Gradients (HOG)

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

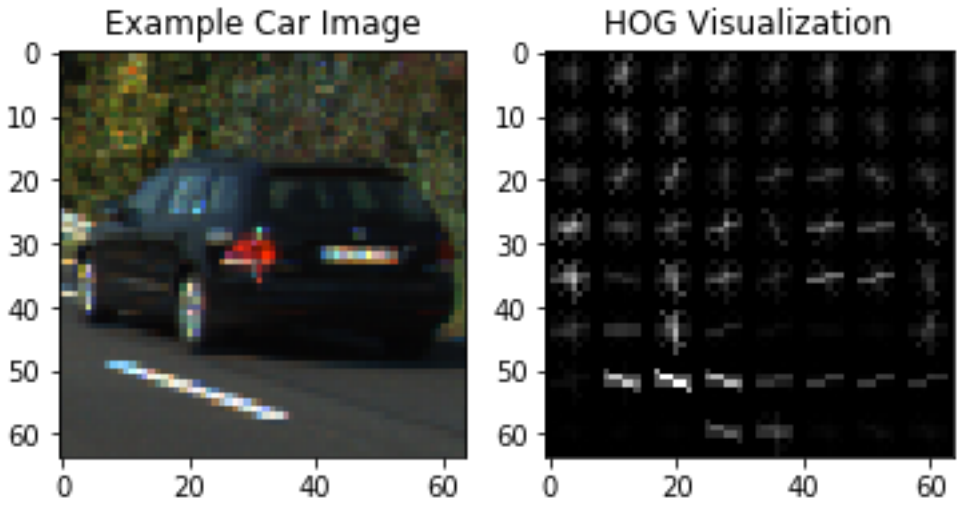
The code for this step is contained in the code cell of the IPython notebook. The cell is marked as Rubric 1 in the notebook (not using cell numbers as it may change due to additional documentation cells).
I started by reading in all the `vehicle` and `non-vehicle` images. Here is an example of one of each of the `vehicle` and `non-vehicle` classes:


car


noncar

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 following with the parameters below.

```
orient = 9
pix_per_cell = 8
cell_per_block = 2
```



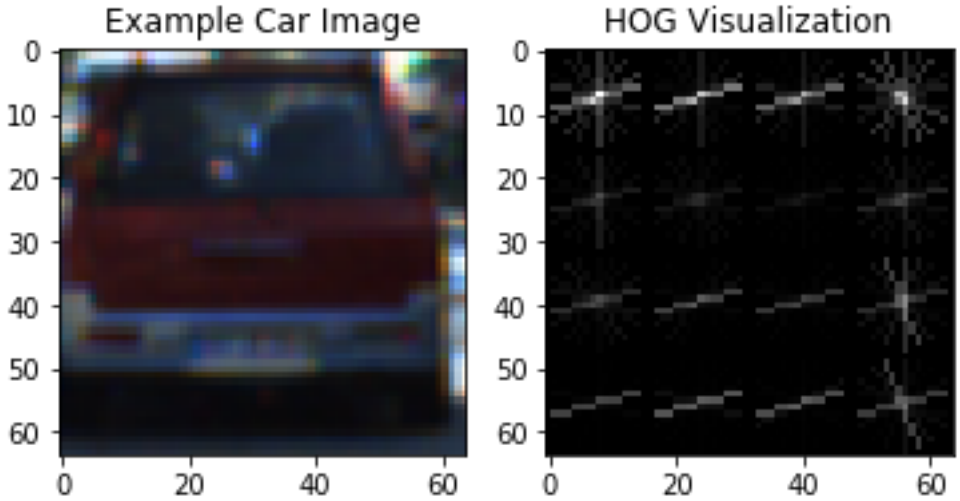
####2. Explain how you settled on your final choice of HOG parameters.

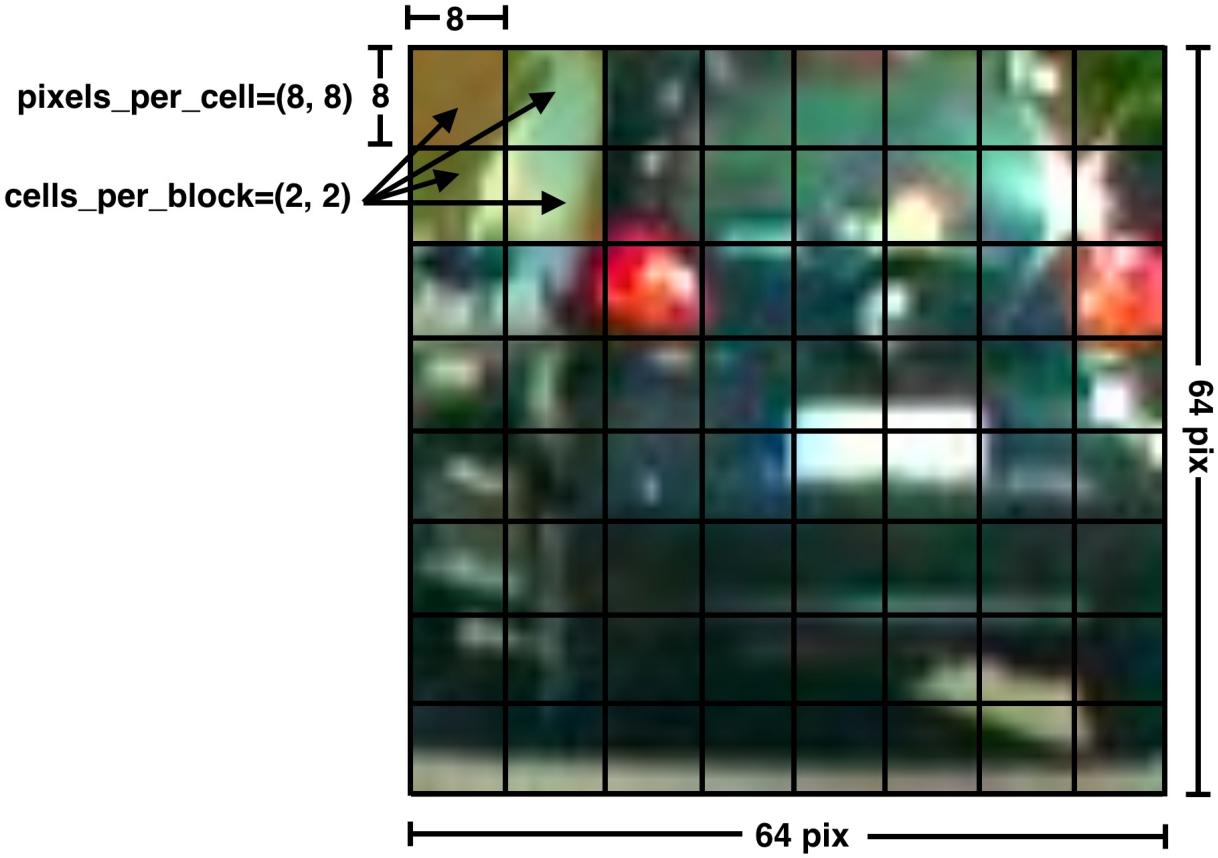
This was the most time-consuming part ☺

I didn't play around with the orient parameter as the defaults in the lesson and based on research https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients where it indicated that 9 was a good value.

Initially I started with the defaults shown in the udacity lesson.

```
orient = 9
pix_per_cell = 8
cell_per_block = 2
```
colorspace = RGB

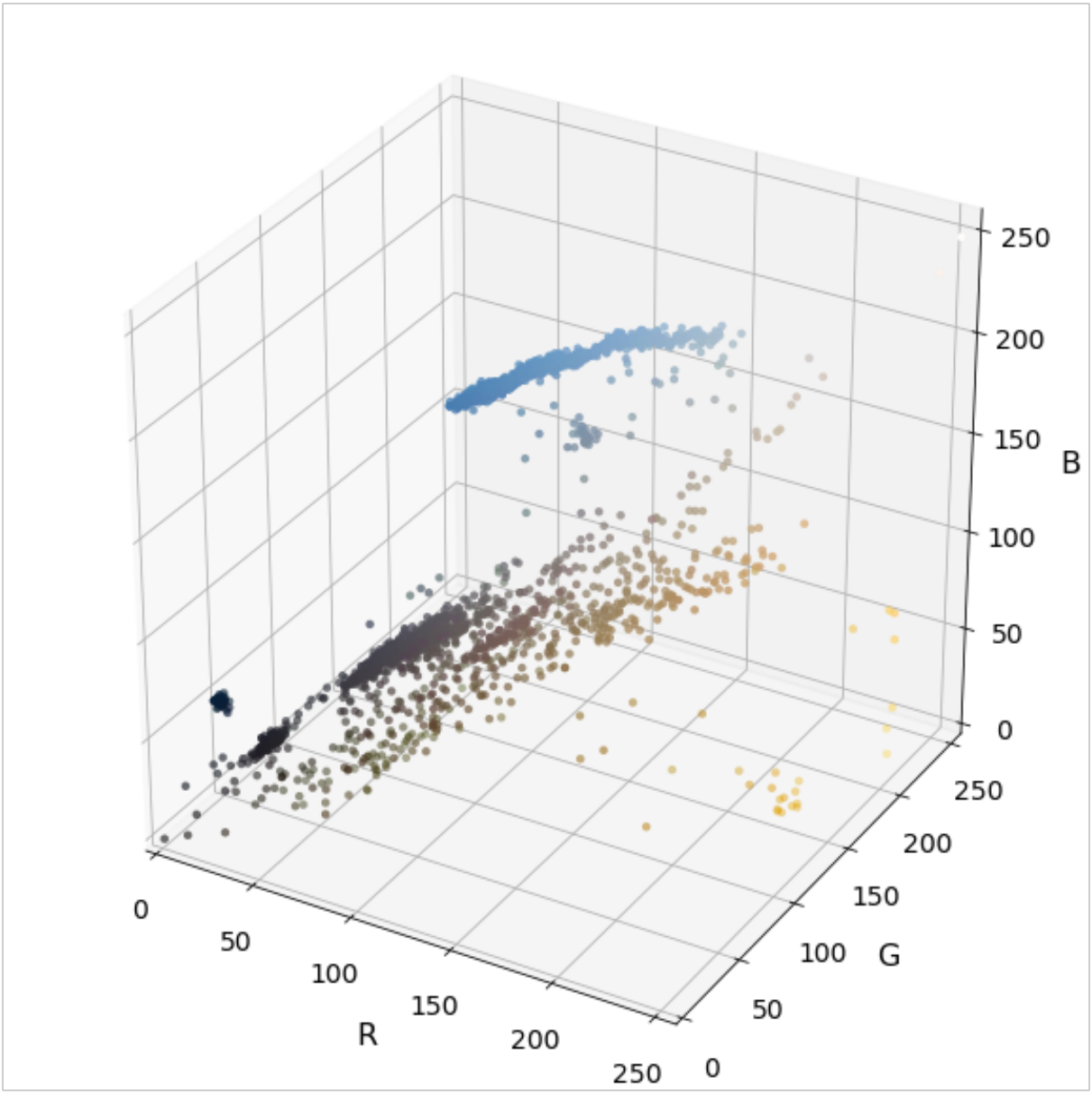


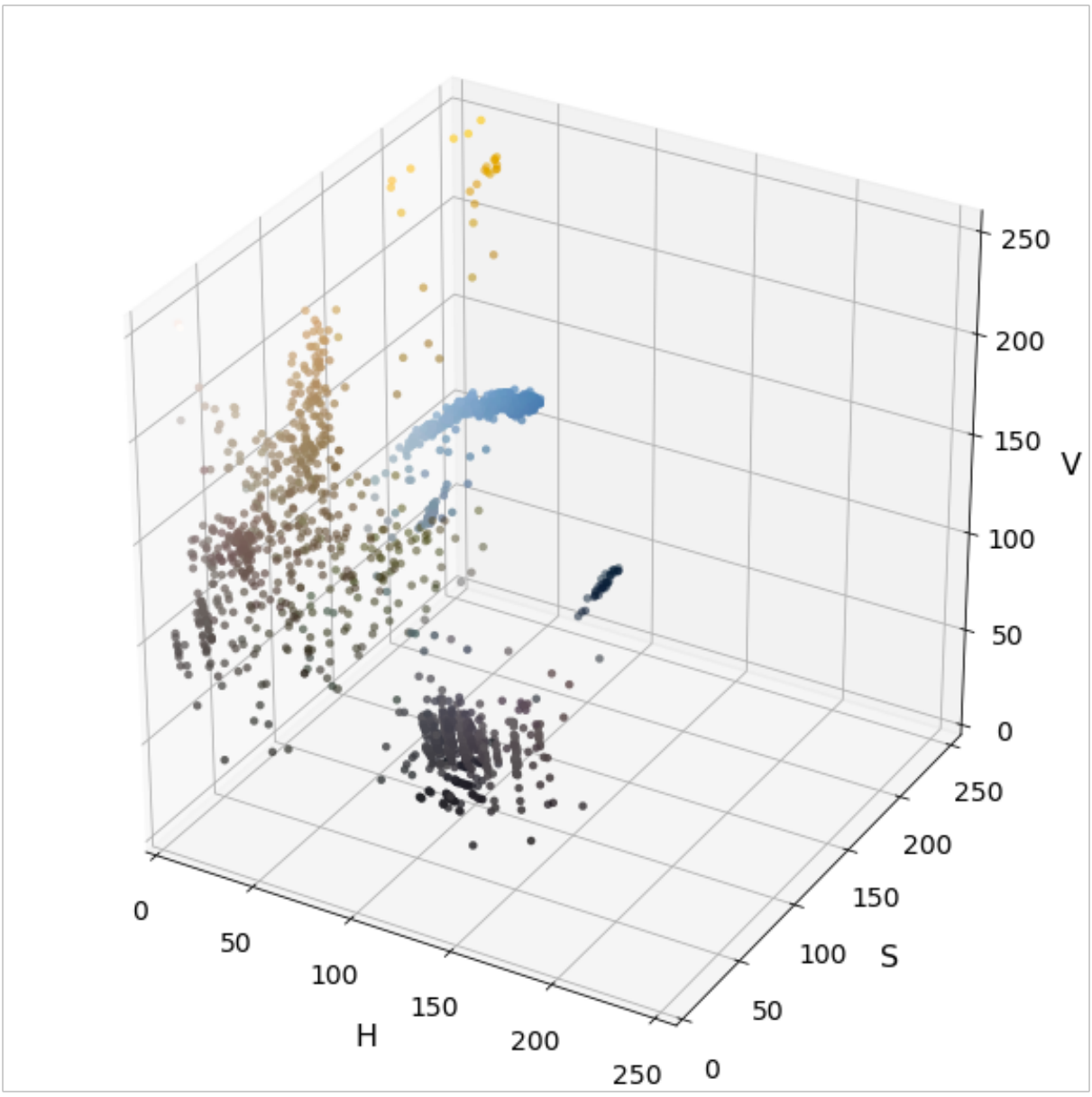


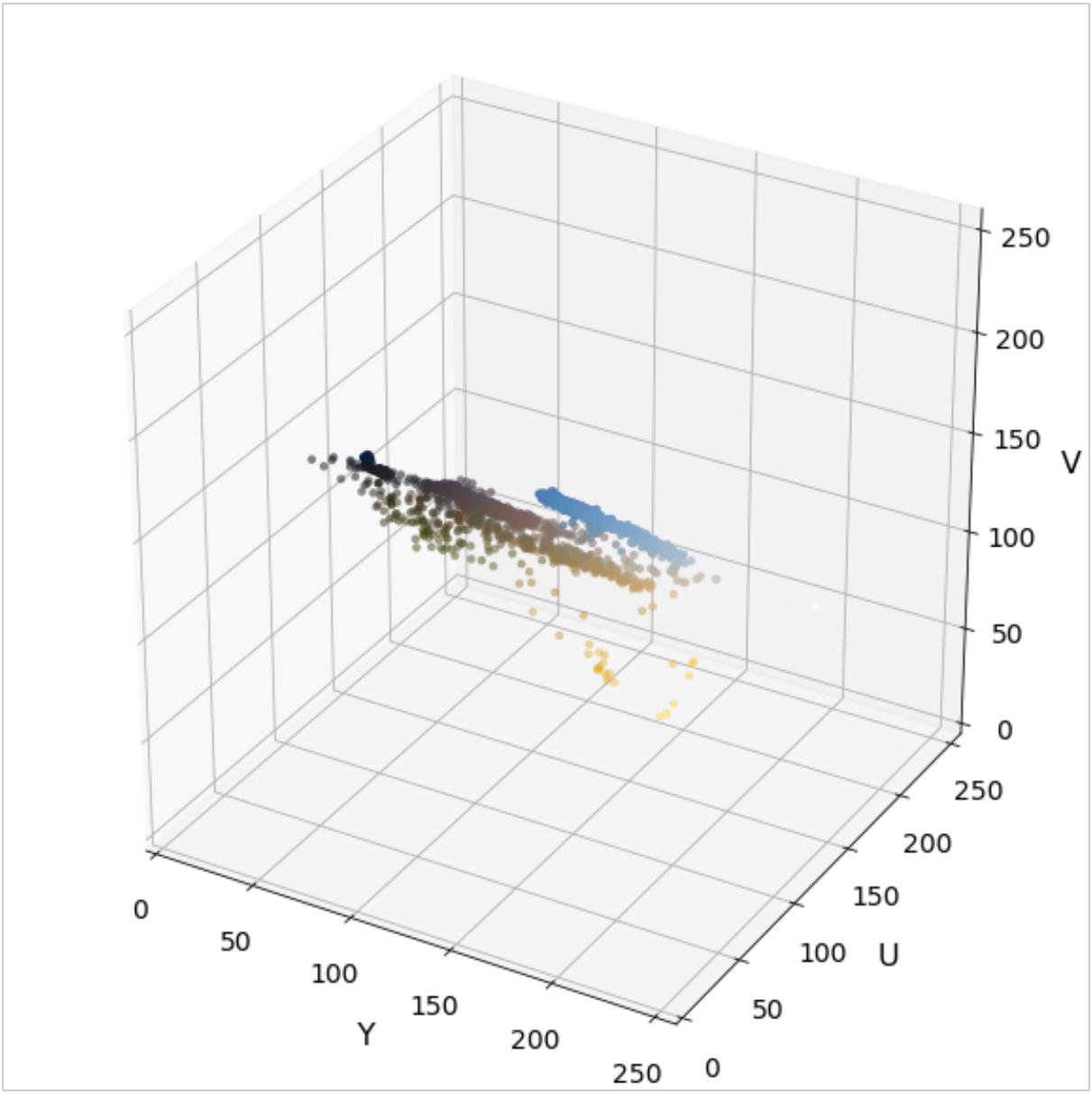
https://classroom.udacity.com/nanodegrees/nd013/parts/fbf77062-5703-404e-b60c-95b78b2f3f9e/modules/2b62a1c3-e151-4a0e-b6b6-e424fa46ceab/lessons/fd66c083-4ccb-4fe3-bda1-c29db76f50a0/concepts/d479f43a-7bbb-4de7-9452-f6b991ece599

With the above values I saw no detections after training.

Then I did a plot of the color spaces on a test image test2.jpg. Based on the results the YUV was the most promising with concentrated distinction of the spaces. I tried YUV as the colorspace and immediately I started to see good detections



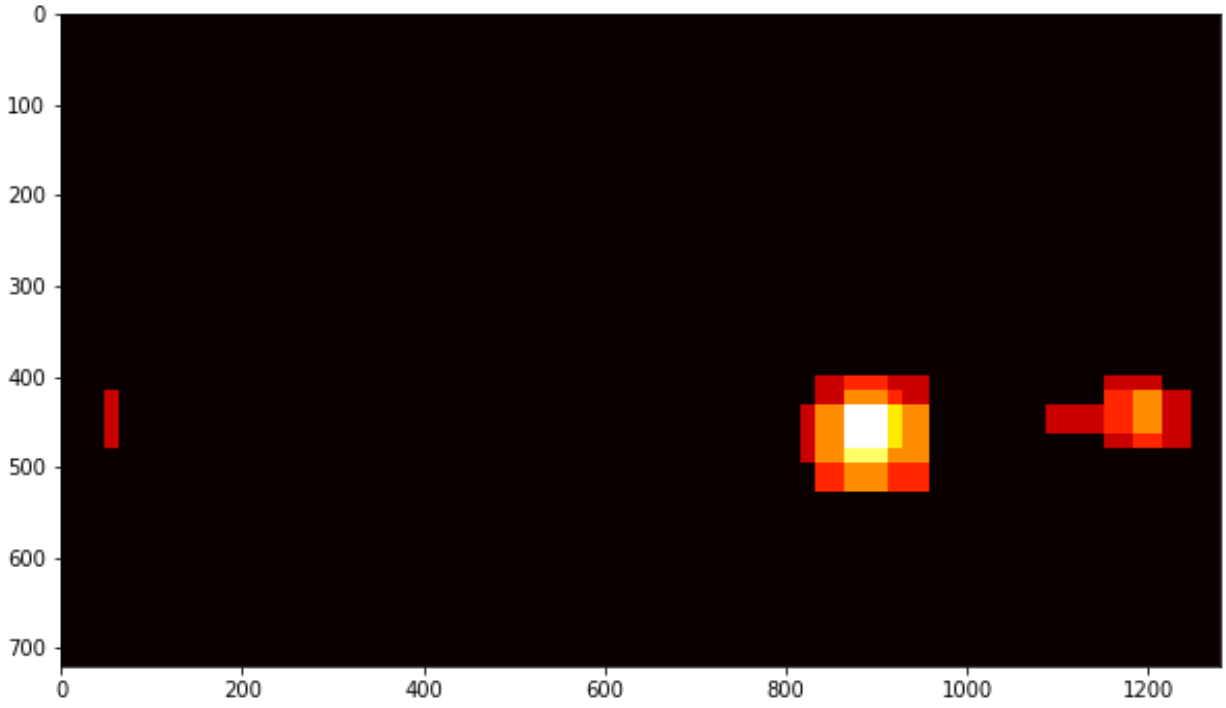

0
50
100
150
200
250
H
0
50
100
150
200
250
S
0
50
100
150
200
250
V



Once I started to get the detections, I also noticed a lot of false positives. I was surprised because in the lesson there weren't any false detections.

Then I increased the pixels_per_cell = 16 (Rubric 2 in the notebook) noting that a bigger block size should reduce false positives. This indeed worked out well.

After this I was I was still getting some false positives with hog_channel as 0.



false positives in heat map with HOG = 0

False Positives in hog = 0

The I started tweaking the hog = 1,2, etc. When I used HOG=ALL, the false positives decreased a lot.

It detected the test images correctly as sown below.

The final parameters are:

```
orient = 9
pix_per_cell = 8
cell_per_block = 2
```
colorspace = YUV

####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 defaults in cell #11. The training data was split 80% to 20% in cell#10.

###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?

Basically, I took the approach of “Educated Guess” approach rather than a random search.

The approach is that lower part of the screen we can expect larger car images therefore bigger scale. And the converse is also true.

I used a pixel step “overlapping” method or size 2. The logic is in cell#3 in function called find_cars. I used several staged “windows of search” based on an educated guess and manually hardcoding the window sections.

I used a test image and proceeded to draw lines in the image based on the staged window search. This method resulted in lot less false postitives

####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?

During testing with frames I noticed that I was still getting some false positives even after the heat_map elimination. Then I started another round of parameter tweaking. Since I didn't play with ORIENT, I started +/1 from the value 9. With ORIENT=11 the false positives in the below frames also got eliminated.

Accuracy with :

```
orient = 9
pix_per_cell = 8
cell_per_block = 2
```
colorspace = YUV

```
1.09 Seconds to train SVC...
Test Accuracy of SVC =  0.9823
My SVC predicts:  [ 0.  1.  1.  0.  1.  1.  1.  1.  1.  0.]
For these 10 labels:  [ 0.  1.  1.  0.  1.  1.  1.  1.  1.  0.]
0.00179 Seconds to predict 10 labels with SVC
```

```
orient = 11
pix_per_cell = 8
cell_per_block = 2
```
colorspace = YUV

```
Test Accuracy of SVC =  0.9842
SVC predictions:  [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]
For these 10 labels:  [ 0.  0.  0.  0.  0.  0.  0.  0.  0.  1.]
```

Video Implementation

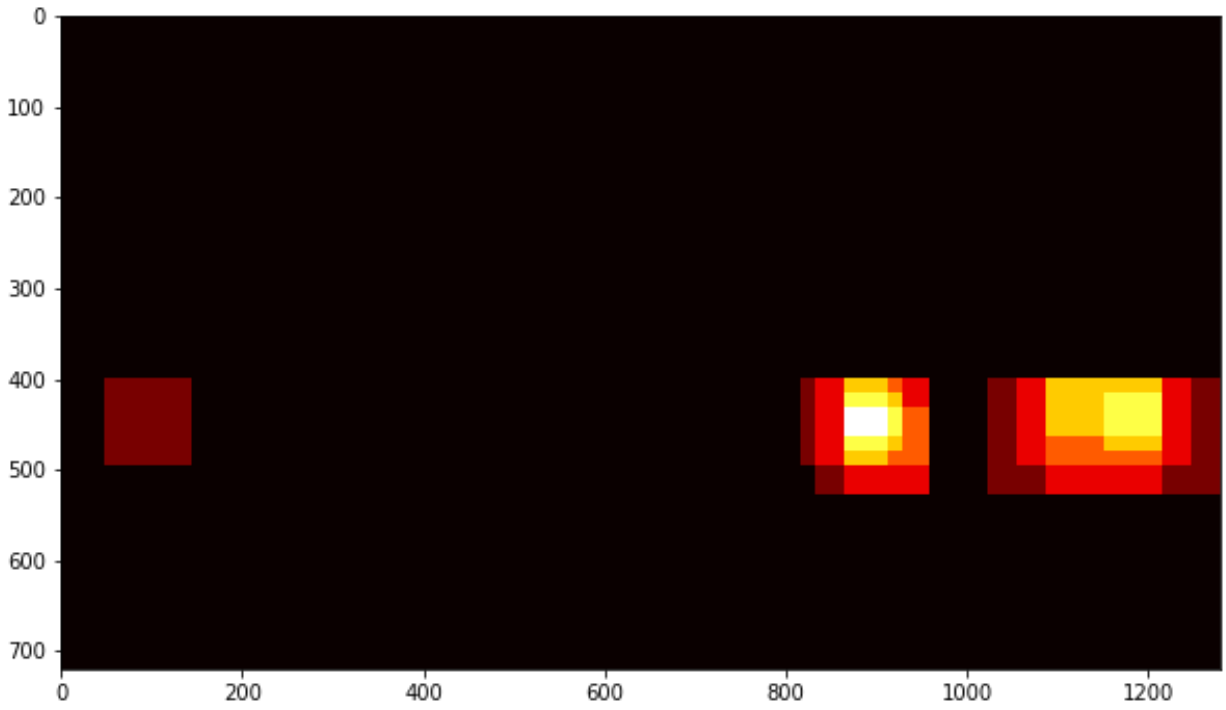
####1. Provide a link to your final video output. Your pipeline should perform reasonably well on the entire project video (somewhat wobbly or unstable bounding boxes are ok as long as you are identifying the vehicles most of the time with minimal false positives.)

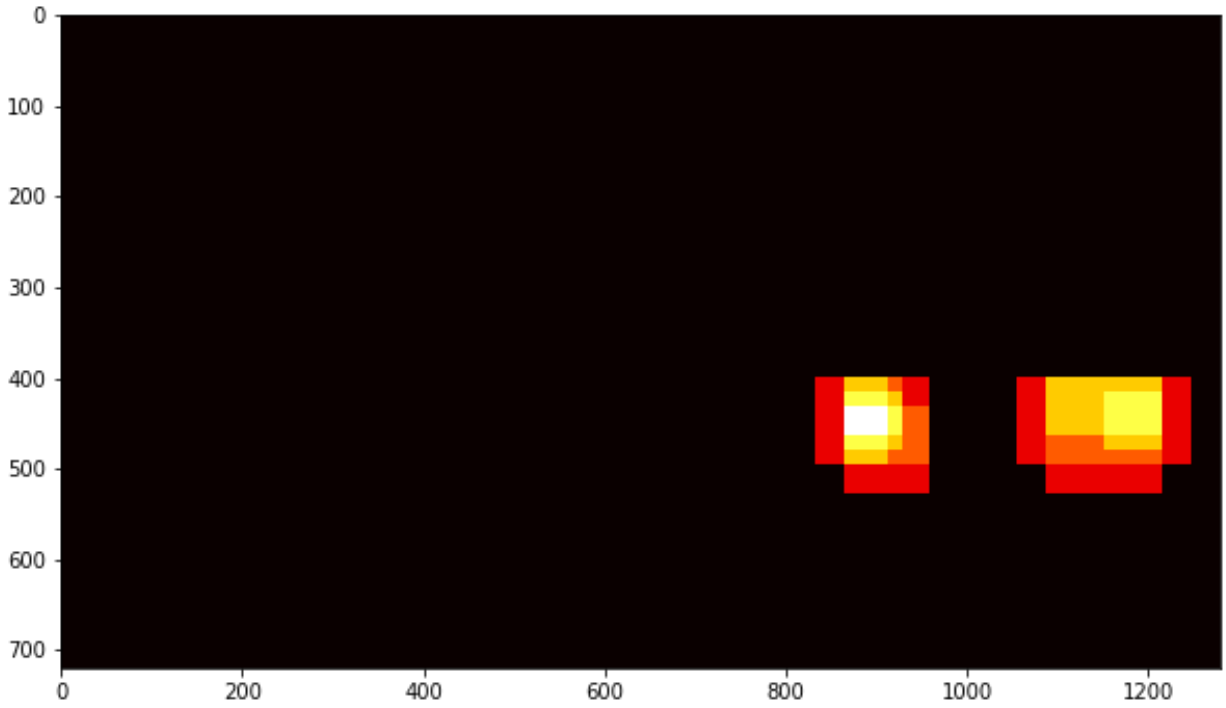
Video is part of archive file in the submission. Also to smooth the rectangles when then are no detections, I have used a history of rectangles to predict the detections in the next frame. I have used a class Run_Car() to store the last 15 historical detections.

####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.

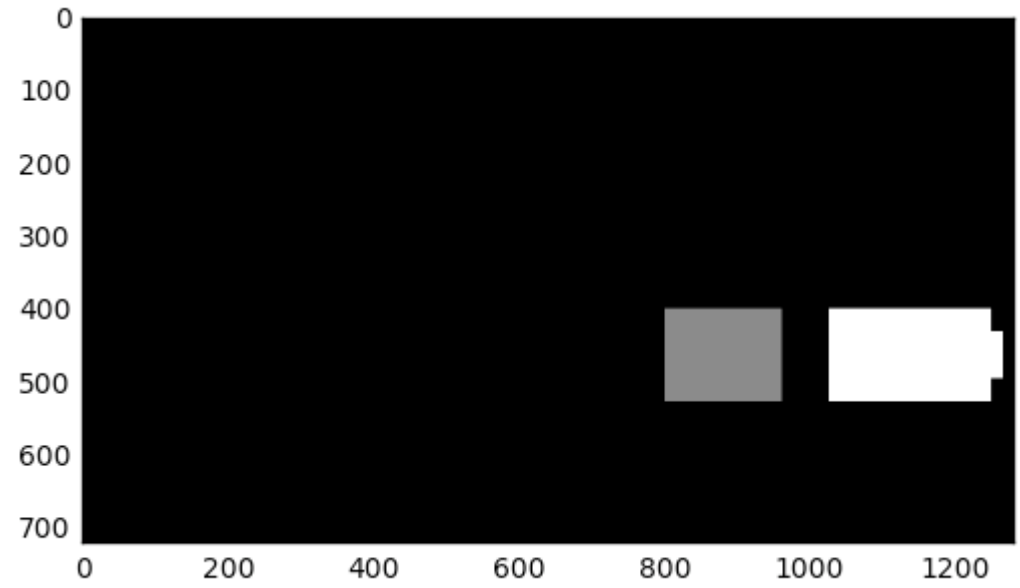
Heatmap is applied in cell 15. This is based on the udacity lesson.

Here are frames and their corresponding heatmaps:





Here is the output of `scipy.ndimage.measurements.label()` on the integrated heatmap from all six frames:



Here the resulting bounding boxes are drawn onto the last frame in the series:

###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?

A: The problems I faced were mainly in finding the right parameters for HOG. I am not sure if I chose the right approach. I feel I should invest in another code where it automatically detects and recommends the right parameters.

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.

A: What I am discovering is that findings the right parameters is almost an art ☺ I feel some investment in this area is needed on my part to perhaps automate some of the parameter selection. If the resolution changes or if the RGB/HSL/YUV changes based on lighting and weather conditions then the video will definitely fail (example sunset, rain).

This also bring up a bigger picture question. I may be totally wrong here. But It almost feels like, just a front camera is not enough for correct detections. There should be augmentation from additional cameras and other sensors. I wish these topics would have been included as part of the course. Example: What do we do during night time? Would the same camera with IR work? Or are LIDARS the only way?