Udacity Self-Driving Cars Nanodegree
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.

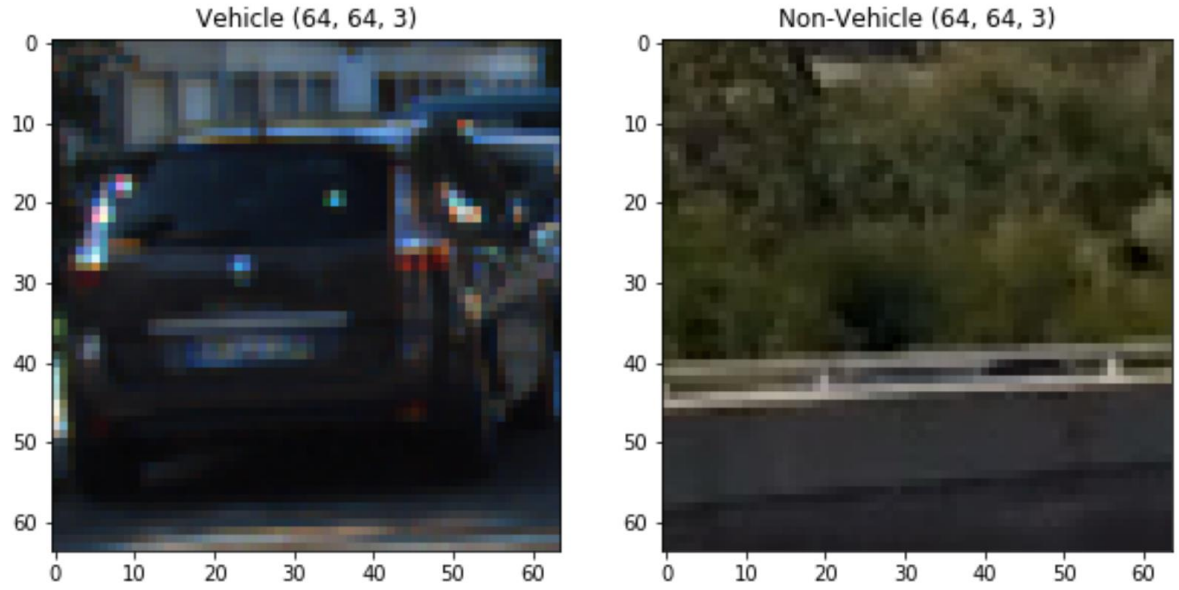
1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. Here is a template writeup for this project you can use as a guide and a starting point.

You're reading it!

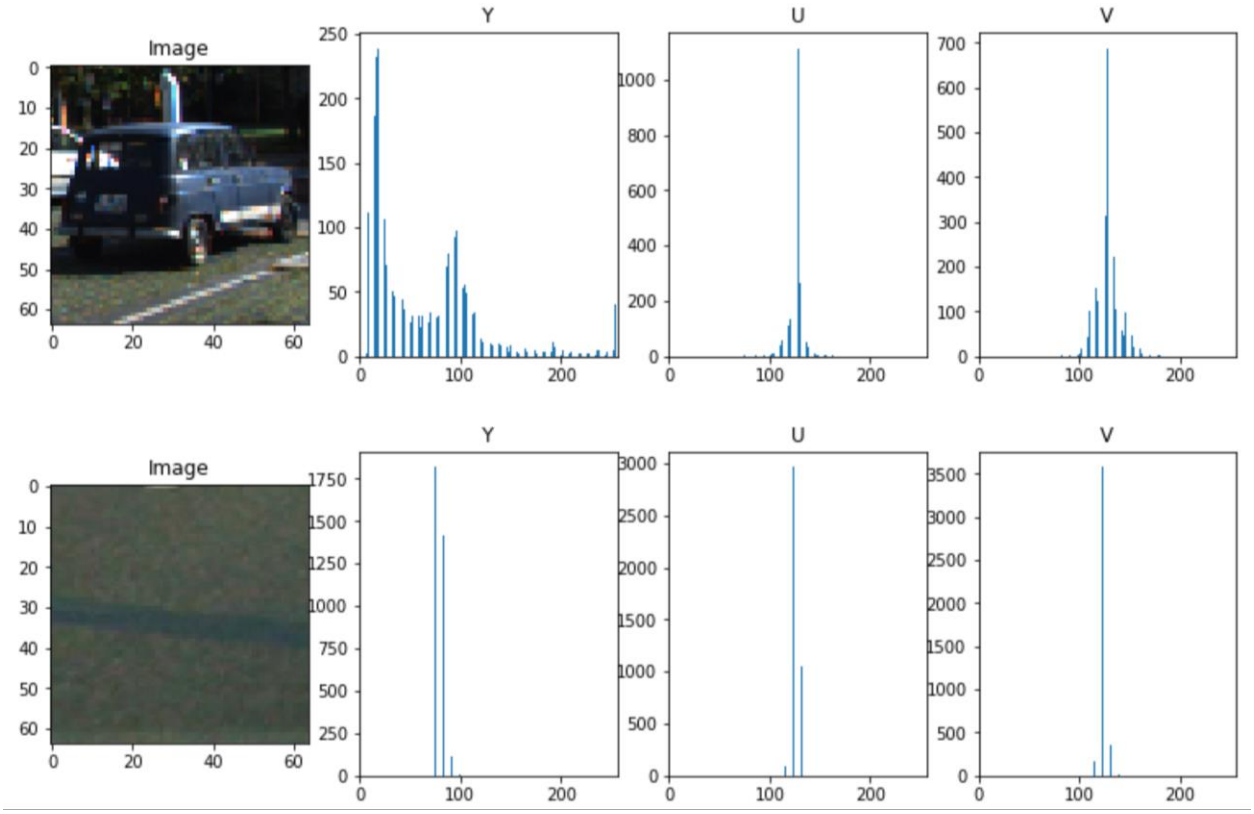
Histogram of Oriented Gradients (HOG)

1. Explain how (and identify where in your code) you extracted HOG features from the training images. (Refer cells 4 to 8)

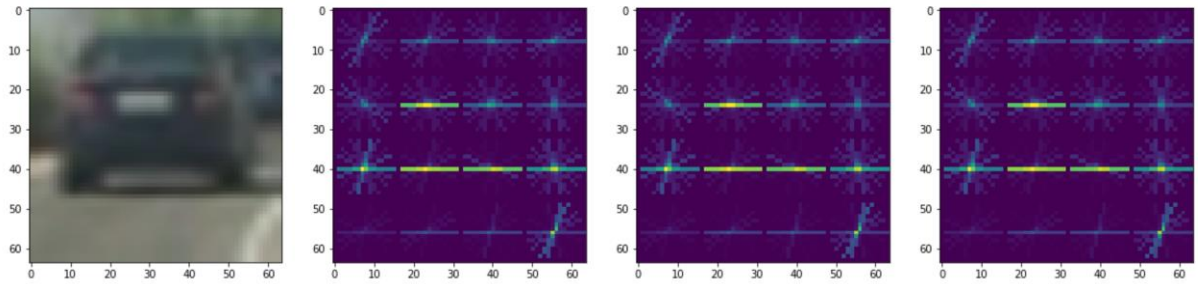
I started by reading in all the vehicle and non-vehicle images. Here is an example of one of each of the vehicle and non-vehicle classes:



I 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.
I selected the YUV color space. Below are the histograms of an image from each of the vehicle and non-vehicle classes.



Here is an example using the YUV color space and HOG parameters of orientations=9, pixels_per_cell=(16, 16) and cells_per_block=(2, 2):



2. Explain how you settled on your final choice of HOG parameters.
I tried various combinations of parameters like the LUV, YCrCb and HSV color spaces with different values of orientations, cells per block and pixels per cell and finally settled on the parameters mentioned in part (1). My selection was based on two factors: feature extraction time and classifier accuracy.

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 SVC and achieved an accuracy of 98.9% on my test data.

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 implemented the sliding window search method in code cell. This method returns the windows list. I decided to use the following windows after some hit and trials. The following points highlight my method.

- I applied the search technique only on the lower half of the image because that is where the cars will be. There were a lot of false positives when I was using more windows initially (searching for all cars).

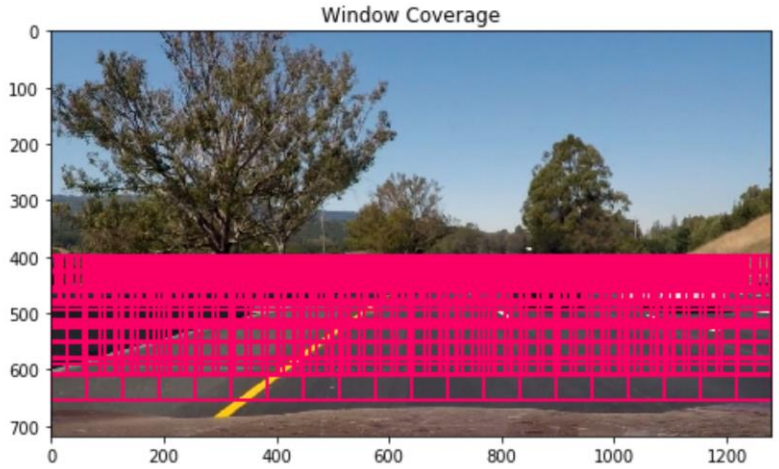
- The cars near the chosen horizon will be small and will increase in size as we move from the horizon towards the car. So it makes sense to search for 64x64 car in only one window starting from horizon and increase the window size as we move from horizon towards car.

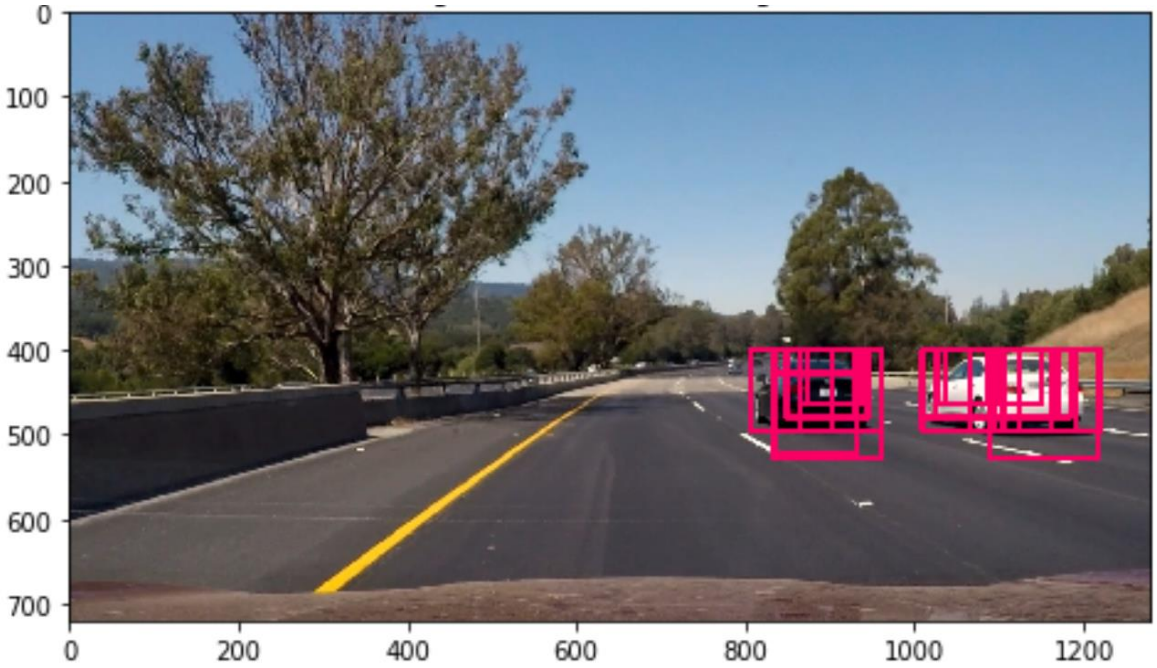
- We also need to resize the window back to 64x64 for different window sizes because while training the classifier we trained on features extracted from 64x64 image.

- The overlap selected was based on the fact that all possible points were covered. I searched for one bar of 64x64 window size near the horizon with more overlapping(85%) windows. Then I decided to search for one bar of 80x80 window size near the horizon with overlapping of 80%.

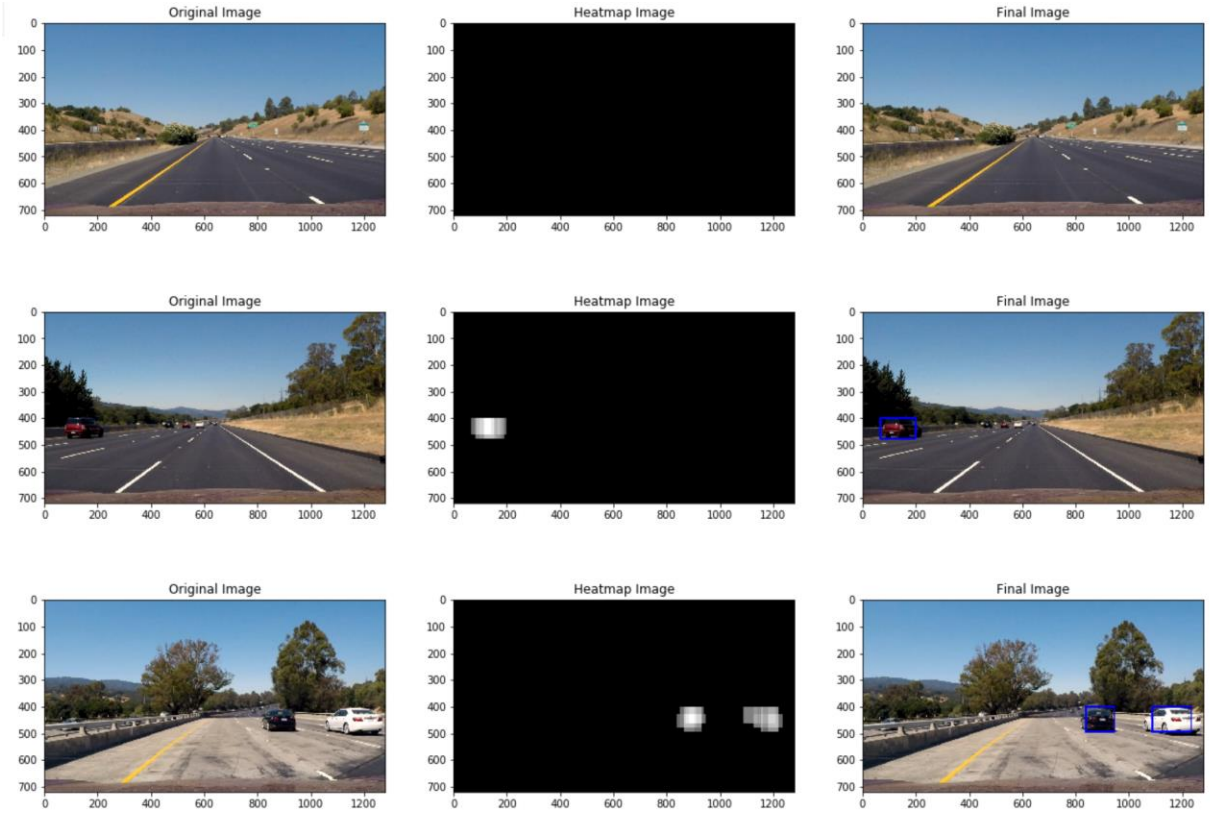
The final windows selected:

Window Size	Start – Y coordinate	End – Y coordinate	Overlap
64x64	400	464	85
80x80	400	480	80
96x96	400	612	70
128x128	400	660	50





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?
I decided not to use the spatial binning but instead pre-process the images like normalizing and scaling.
I used a heatmap to plot the final bounding box around the detected car.



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?

Problems faced:

- Despite using a heatmap and averaging the results of vehicle detection over 15 frames, I obtained a lot of false positives when I searched all windows over the image. This is why I decided to use the lower half of the image for detecting windows.
- I used the SVM with “rbf” filter initially took longer and had lower accuracy. Finally I arrived at the result by using a linear SVC.

Where will the pipeline most likely fail:

- The pipeline will fail to detect traffic on the other side of the road (which I believe is not much of an issue)

Future work:

- I feel that detecting vehicles on any portion of the image would be a useful feature.