CarND-Vehicle Detection

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
- Apply a color transform and append binned color features, as well as histograms of color, to your HOG feature vector.
- Normalize 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 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.

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 "CarND-VehicleDetection.ipynb "(cell 3).
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 with the HOG feature extraction, setting the parameter as: orientations = 9, pixels per cell = 8, and cells per block = 2 on the 300-example pool to evaluate the linear SVM accuracy. The result is as follow:

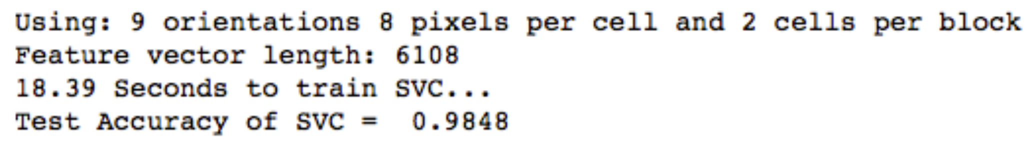
RGB	0.8875
HSV	0.9375
LUV	0.9375
HLS	0.875
YUV	1.0
YCrCb	0.975

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

I also tried increasing the orientation to 12 and increase the pixels per cell to 16. But the accuracy will not improve much while the speed to extract features are slowing down.

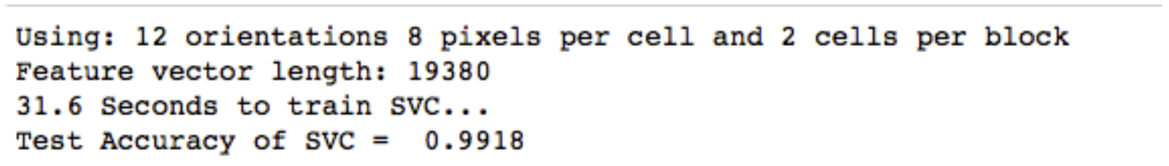
3. Describe how (and identify where in your code) you trained a classifier using your selected HOG features (and color features if you used them).

I found that introducing the color histogram and spatial features can increase the accuracy a bit without slowing down the speed too much. So I trained the linear SVM with a combined feature of hog, color histogram and spatial features. Result:



```
Using: 9 orientations 8 pixels per cell and 2 cells per block
Feature vector length: 6108
18.39 Seconds to train SVC...
Test Accuracy of SVC =  0.9848
```

Result 1



```
Using: 12 orientations 8 pixels per cell and 2 cells per block
Feature vector length: 19380
31.6 Seconds to train SVC...
Test Accuracy of SVC =  0.9918
```

Result 2

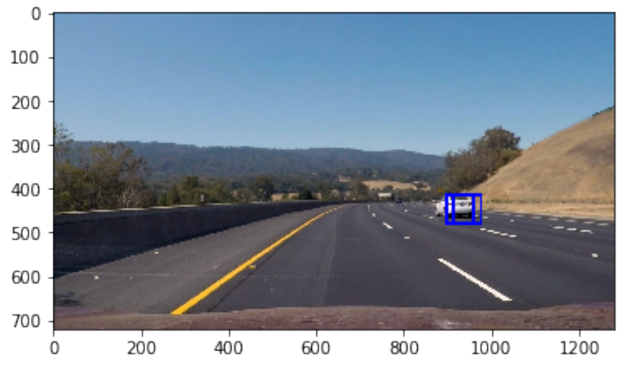
Sliding Window Search

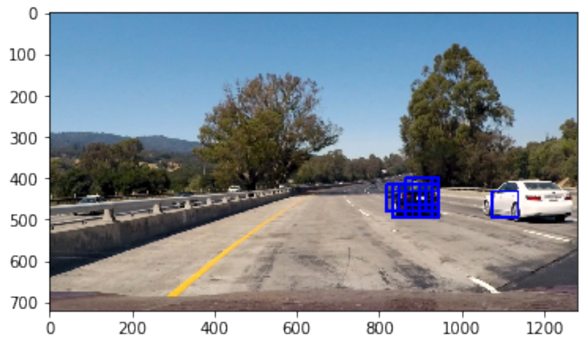
1. Describe how (and identify where in your code) you implemented a sliding window search. How did you decide what scales to search and how much to overlap windows?

First, since the vehicle will not likely appear in the sky, I limited the start and stop point on y-axis as 380,650. Then I tried combination of scale 1, 1.5 and 2 as well as individual scale. Finally, I found the scale 1.5 can detect the vehicle with good confidence while produce less false positive overall.

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 first fit the classifier with hog feature and color histogram feature, but there are a lot of false positive and the predict accuracy is under 99%(result 1). So I tune up the parameter of feature size and also included the spatial feature. Ultimately, I used all channel HOG feature in YUV color space, (64,64) spatial feature and 12 bins histogram of color feature, which provided result 2. Here are some example images:





Alternatively, I built convolution neural network(CNN) with Keras to directly use the model train the 64x64 data set images. The code can be found in cell 23-25.

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.)

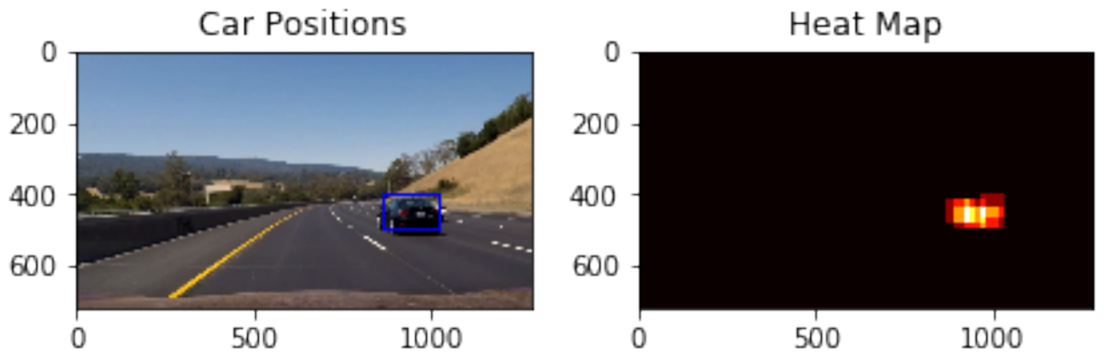
The video is attached in the zip file.

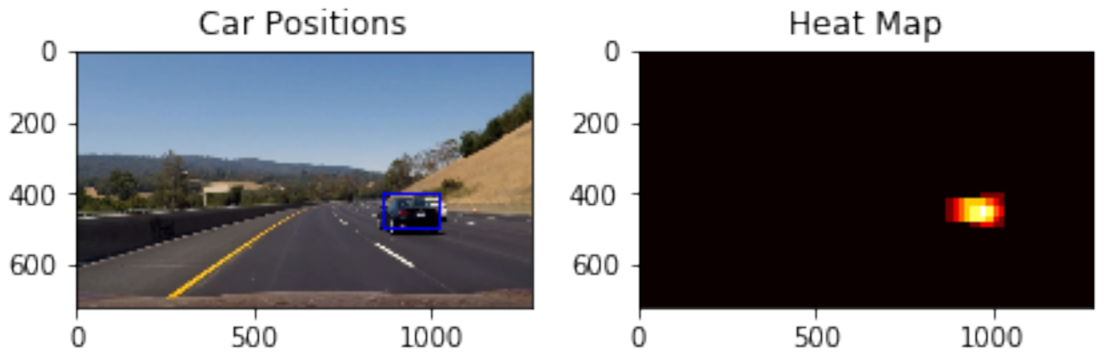
2. Describe how (and identify where in your code) you implemented some kind of filter for false positives and some method for combining overlapping bounding boxes.

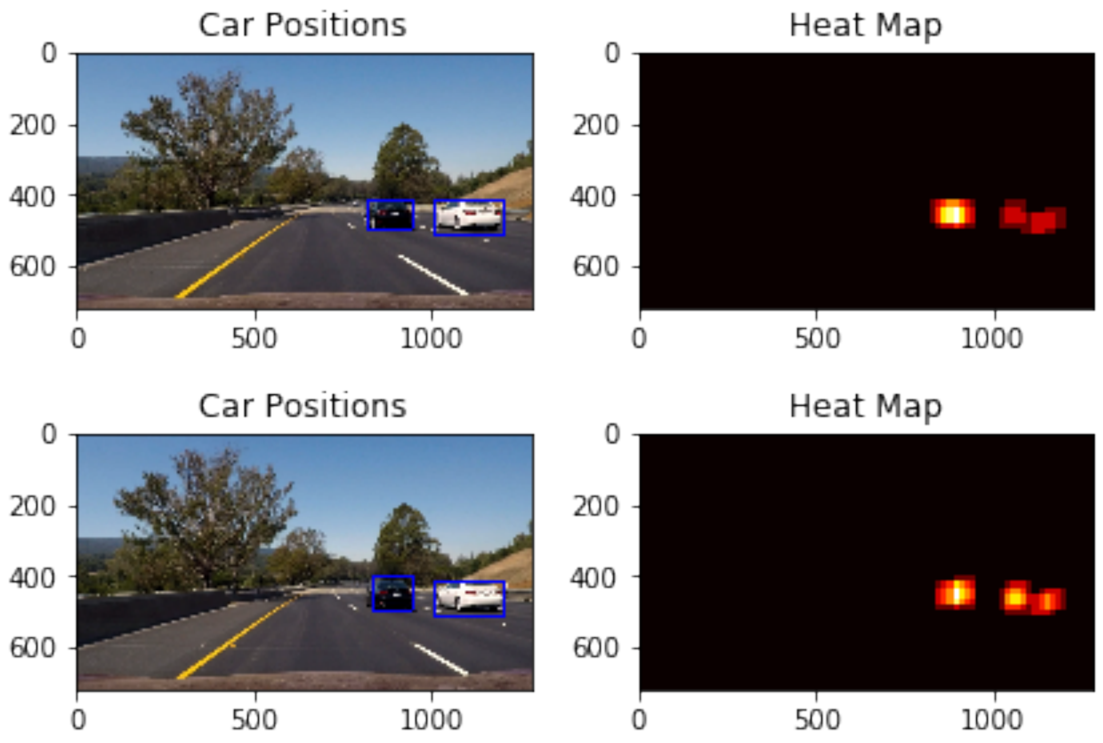
I recorded the positions of positive detections in each frame of the video. From the positive detections I created a heatmap and then apply thresholded on that heatmap to identify vehicle positions. I then used `scipy.ndimage.measurements.label()` to identify individual blobs in the heatmap. I then assumed each blob corresponded to a vehicle. I constructed bounding boxes to cover the area of each blob detected.
Here's an example result showing the heatmap from a series of frames of video, the result of `scipy.ndimage.measurements.label()` and the bounding boxes then overlaid on the last frame of video.

In addition, I used pickle module to record last 20 frames heat map as reference. And add last frame heat map to the heat map histogram to accumulate higher heat. Then apply the threshold to filter out the false positive.

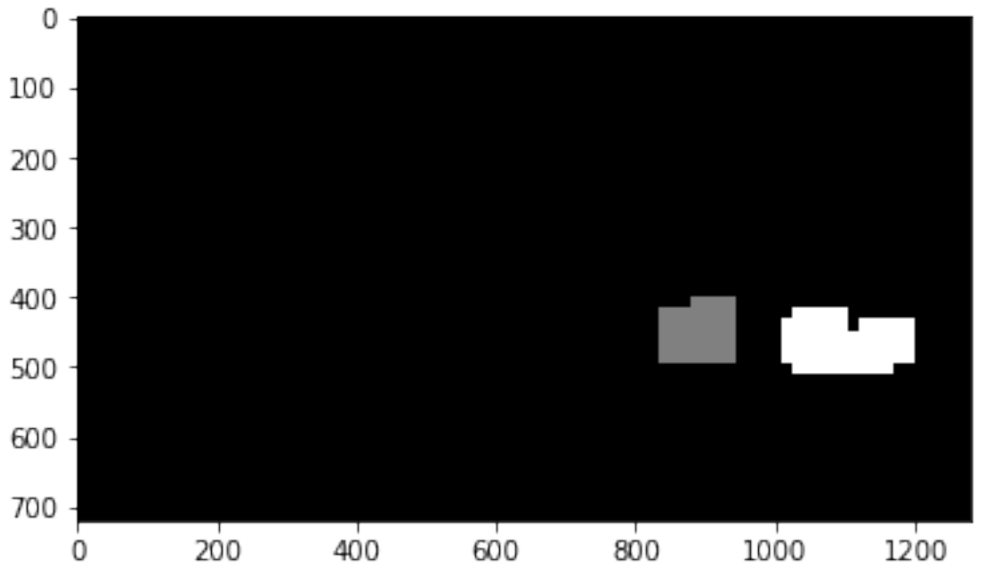
Here are some frames and their corresponding heatmaps:



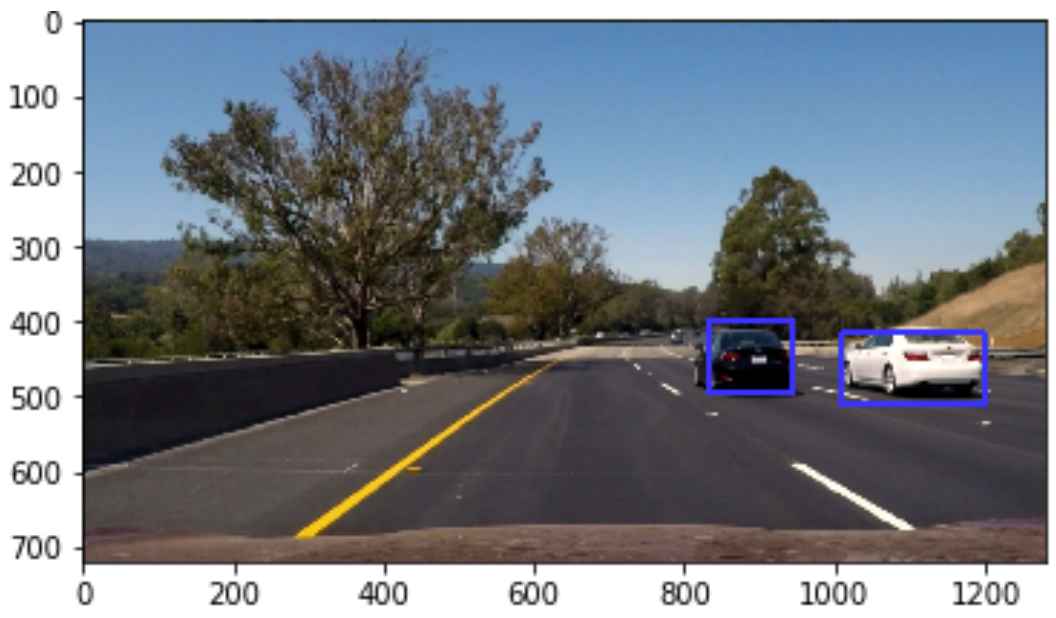




Here is the output of `scipy.ndimage.measurements.label()` on the integrated heatmap from last two frames:



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?

At first, I found the black vehicle does have any heat under strong brightness, so I tune up the feature vector length by increasing the orientation of hog and size of the spatial feature to obtain a better classifier. In addition, there are still few false positive at some bright place, I think it can be improved by adding one more color space hog feature to the classifier.