Vehicle Detection

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 second code cell of the IPython notebook. Different color spaces were experimented with, but I have used HLS eventually. Histogram features and spatial features are also appended to the HOG feature vector. The HOG implementation is the standard OpenCV implementation.

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

After trying various combination of values and seeing the resultant boxes being drawn on one of the test images after the SVM training, I could figure out which values were working and which weren't.

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).

A SVM is trained (code cell 11) on the HOG features + Histogram features + Spatial Features using a 'rbf' kernel. The performance is tested using a hold-out set (20% split).

Sliding Window Search

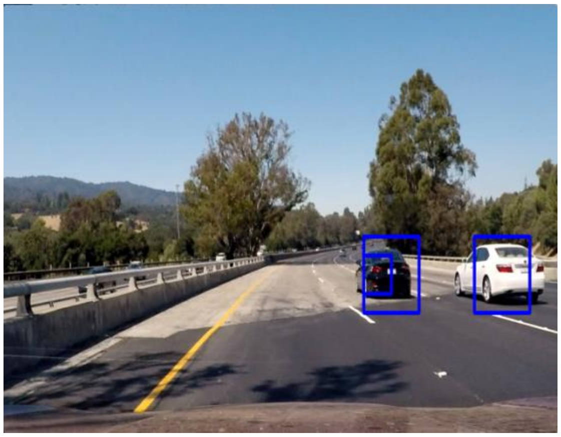
1. Describe how (and identify where in your code) you implemented a sliding window search. How did you decide what scales to search and how much to overlap windows?

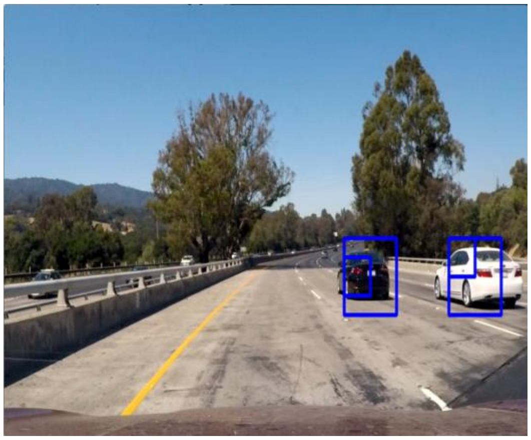
For the height of the image ranging from [390, 656] I use a window size of (128,128) using an overlap of 0.5 in both directions. I do another sliding window search using a window size of (64,64) for the image height ranging from [390, 500] also with an overlap of 0.5. Also, I use a threshold of 0.7 for the confidence value of the SVM. Only when the confidence is greater than this value, the prediction is set to 1. Also, when the confidence of the current search window is greater than the previous search window, the latter is replaced by the former (given the prediction value was 1).

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?

Using a larger threshold (explained in the previous question), the 'rbf' kernel for the SVM, more search windows and all the three features (HOG, Histogram, Spatial binning) helped optimize the performance of the classifier substantially. The result is shown below. The multiple boxes for the same object are removed in the video using a heat map for the last 25 frames.

Vehicle Detection





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.)

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.

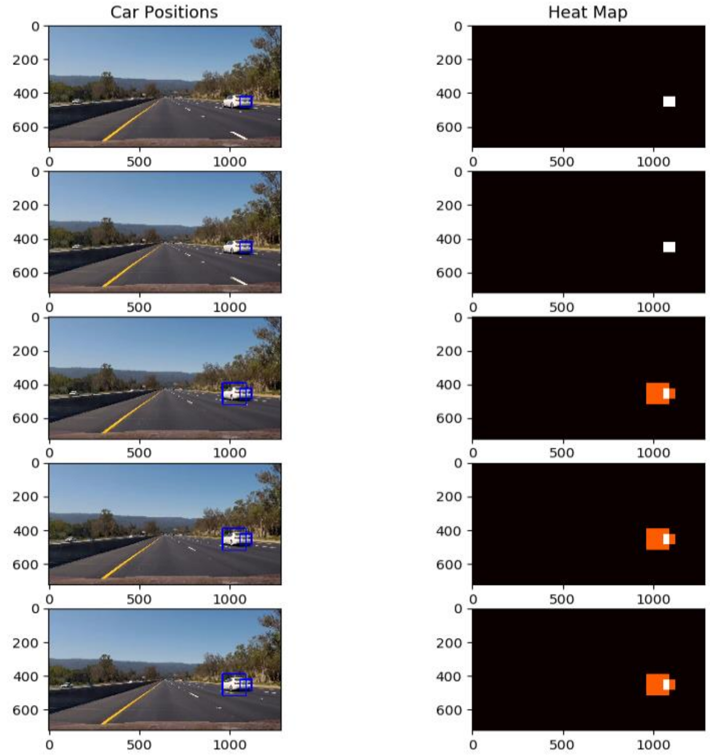
I recorded the positions of positive detections in each frame of the video using a threshold of 0.7 for the SVM prediction. When the number of frames seen were greater than 25, I created a heatmap using the last 25 frames and then thresholded that map to identify vehicle positions. I then used scipy.ndimage.measurements.label() 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. The heatmap for 5 consecutive frames from the test video is shown on the next page.

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?

The main problem I faced was of removing the false positives. This pipeline will likely fail in the case of more traffic where it probably won't be able to recognize all the cars quickly. Having more data can certainly help.

Vehicle Detection


Car Positions
Heat Map