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.

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

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 #15 of the jupyter notebook. The hog features are extracted by ‘get_hog_features’

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 using an HLS image transformation with HOG parameters of `orientations=6`, `pixels_per_cell=(8, 8)` and `cells_per_block=(1, 1):

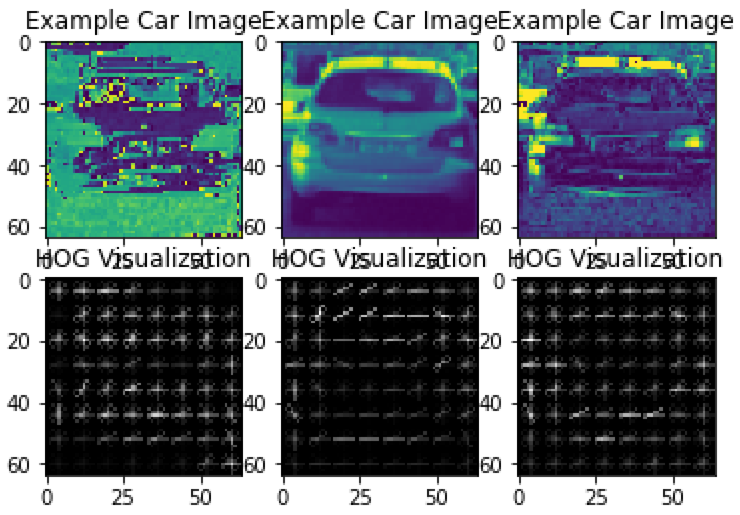

Figure 1. HOG features, car, HLS

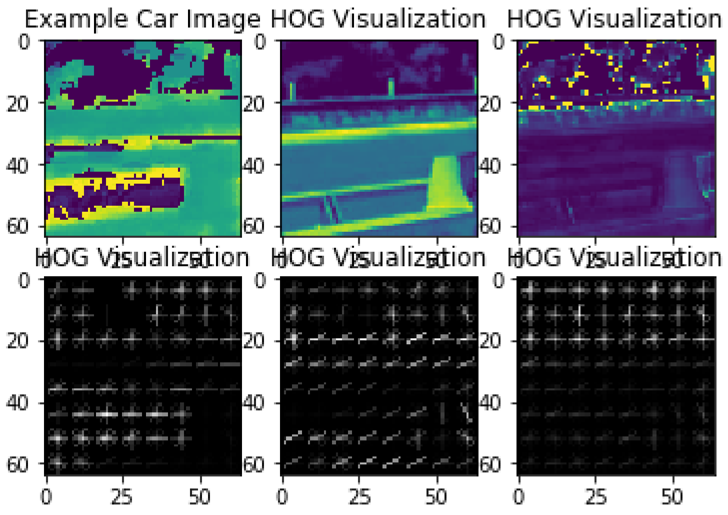

Figure 2. HOG Features, not car

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

I tried various combinations of parameters and because I was trying different combinations was interested in speed. I started with 6 orientations, 8 pixesl and 1 cell per block and combined with using only the hog features for the L channel, was able to achieve ~0.95 test accuracy, with a linear classifier with C = 0.001

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).

The classifier is built and trained in Code Cell 22. I trained a linear SVM using C parameter of 0.001. I had tried using GridSearch for rbf and linear classifiers with various gamma and C parameters. Ultimately, I narrowed my choice down to linear because the rbf kernels took too long to train and had negligible benefit in accuracy. I used GridSearch to further explore the C parameter for the linear kernel and found that 0.001 had a good result .

I ended up using color and spatial features as well. For color, I knew the white car was more difficult to classify based on the decision_function value output in comparison to the shadows. 32 bins seemed to work more effectively compared to 16 bins.

For spatial features, I tried to base my decision on the pixelated image and trying to judge any visual differences based on the spatial feature visualization. Visually, I could see none but knew that computation time was compromised at values greater than 16x16 so I stuck with 16x16.

The classifier uses all three featuers (1 channel HOG, 32 bin color histogram, and 16x16 spatial features)

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?

To implement a sliding windows search, I used the code from Lesson 21, Step 35 of Object Detection and modified it a bit so that it would work with the feature selection parameters and HOG channel selectors. In order to see how the parameters worked I checked the output in the lesson, and used the same image to check my results.

I settled on using smaller scales in the upper region of the area where I wanted to find cars (400 and more) and larger scales in the lower region, keeping in mind processing time. I settled on using 3 sweeps which seem to be able to classify vehicles both lower into the frame and higher up in the frame

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?

Ultimately, I searched on three scales using HLS 1-channel HOG features plus spatially binned (16x16) color and histograms of color (32 bins) in the feature vector, which provided a nice result. I played around with different color spaces, HOG parameters and color histograms and spatial binning. I tried to visualize each step so that I could tune the parameters.

For example, in cell 10, I looked at what the different layers of the color transformations might look like. I knew that the HOG features worked best on a grayscale image so HLS, YUV, or YCrCb might be good choices since the H or Y channels provided this kind of image. Visualization is shown below.

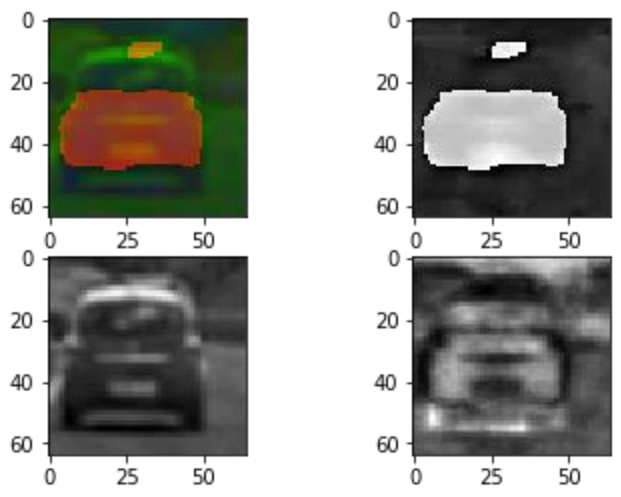

Figure 3. Image layer visualization

I also wanted to understand how spatial images varied using the parameters and what the feature vector might look like. In Cell 14 I explored what these images and vectors looked like so I could choose the dimension for spatial binning

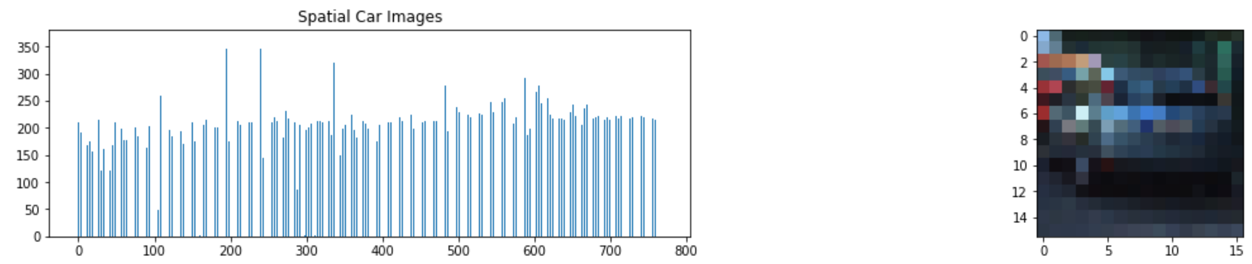

Figure 4. Spatial Binning Visualization

Finally I wanted to see what HOG features looked like for each of the image channels and wanted to see which channel might contribute the most for image classification. In cell 16, I attempted to do this and saw that the H channel of an HLS image seemed to be visually different than the H channel of a non car image.

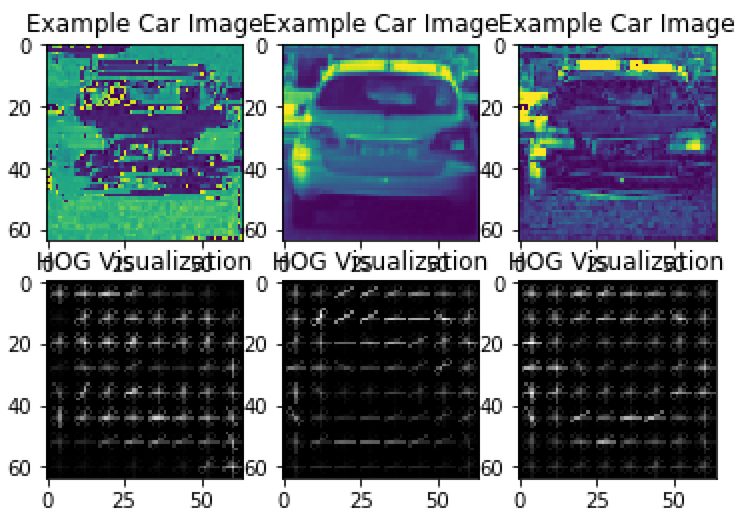

Figure 5. HOG of HLS conversion of car image

Here are some example images of identified vehicles:

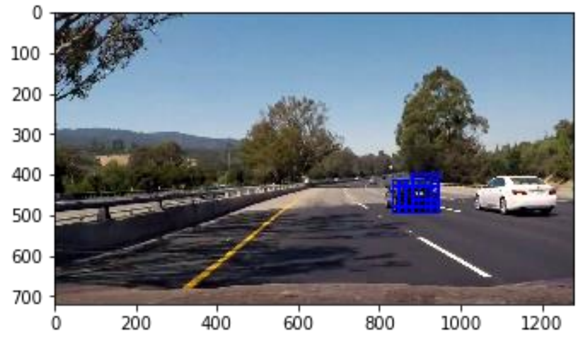

Figure 6. Search in upper portion

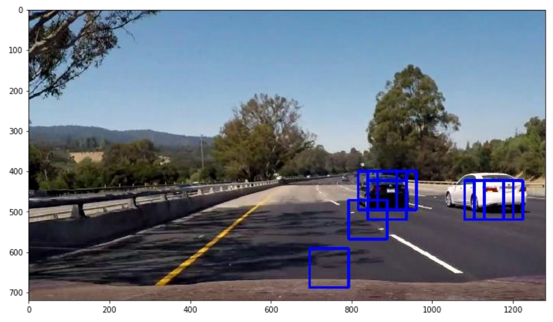

Figure 7. Search in middle portion

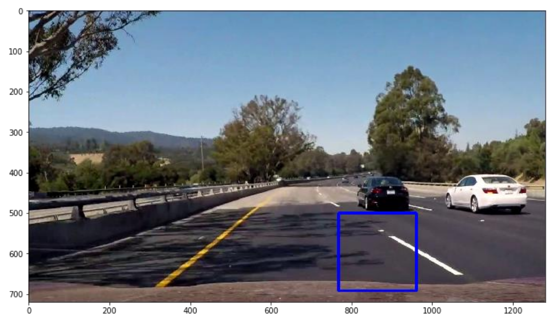

Figure 8. Search in Lower Portion

I then combined all the search areas and applied a threshold to the combined heat map to achieve a final image shown below. What I found was that even with just 1 HOG channel, 6 directions, and 1 cell per block, 16x16 spatial binning and 32 bucket color histogram, my classifier was still scoring around 97%. Adding more features would yield about 98% but slowed down the image processing quite a bit. As such, I settled on a lower number of parameters to gain speed. I was also finding that my classifier was weak on false positives in shadow areas and identifying the white car so I needed to tune the false positive rejection more.

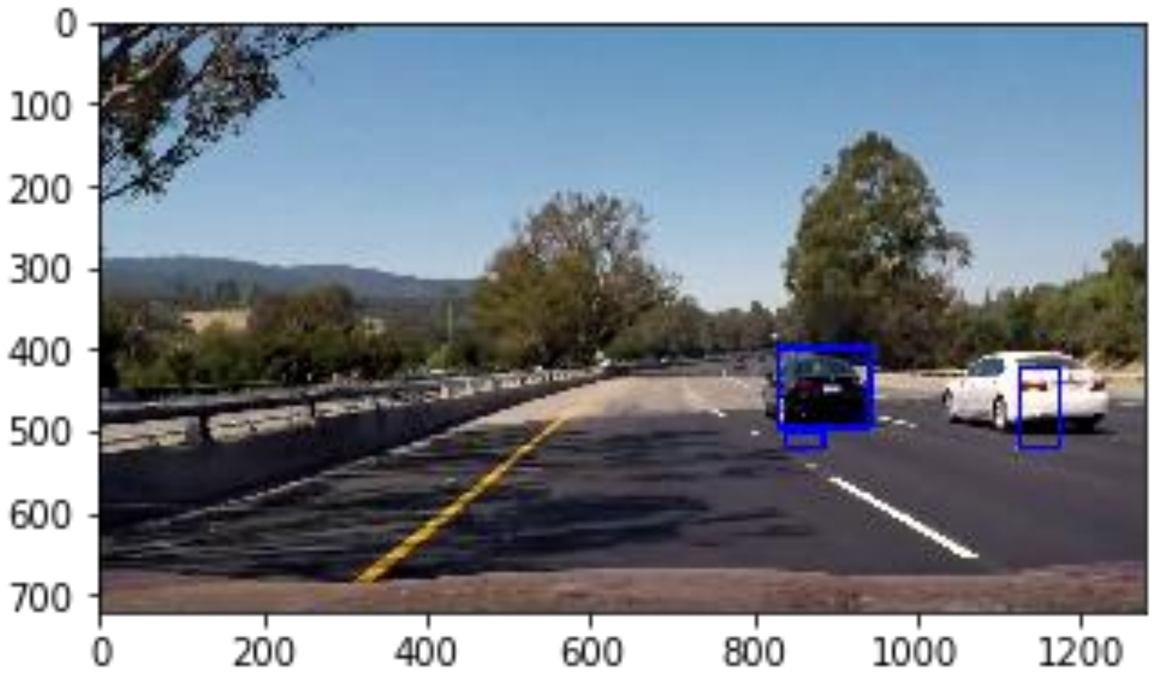

Figure 9. Combination of all search areas

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

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.

The video still exhibits false positive identification. As I mentioned the classifier confidence on shadows and the white car was relatively close while the darker cars are much more confidently identified.

I implemented a false positive rejection scheme in cell 19 by trying to use decision_function to filter out low grade scores. This is where I was finding that white car identifications and shadows have similar scores.

Then in cell 27, the video pipeline, I scan the image 3x using different box scales, starting points and overlap to try to find the vehicles. In here the decision_function is used to filter out low quality identifications. After the boxe coordinates have been returned, a heat function is used to filter out identification areas even more. I this case, I used a threshold of 3 to try to keep images of the white car.

I then used a Class to hold and x number of images for summing good identifications. I sum over 10 images and use a high threshold (100) in order to try to filter out shadows only appear in the video briefly vs the car which is persistent for longer times

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?

Right now the pipeline is failing on the mis-classifications of shadows as cars and the white car. Augmentation of the data set might help.

Also the false-positive rejection is highly sensitive and very much tuned for this brief video. Changes in image quality, terrain would make this vehicle identifier struggle.

Also, untested is if the car is travelling in the center or right line. I don't know how the pipeline will perform with identification of vehicles in the left lane