1. Explain how (and identify where in your code) you extracted HOG features from the training images. Explain how you settled on your final choice of HOG parameters.

The extraction of HOG features is realized in “extract_features” function.

The color space is set as “YCrCb’, the orientation is set as “9”, the pix_per_cell is set “8”, the cell_per_block is set as “2”, and the hog_channel is set as “ALL”.

The selection of these parameters is determined through the performances of support vector classifier, the current setting gives a classification accuracy of 99.1%. The other settings gives as best as 98.6%.

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

Except the HOG features, the color features are also used. They are realized in “bin_spatial” and “color_hist” functions. The spatial binning dimensions are set as (32, 32), the number of historical bins is set as 32.

The dataset includes 8,792 vehicle images and 8,968 non-vehicle images. Finally 8460 features are extracted for each image. All of them are normalized to be zero-centered with unit variance.

Random shuffle is applied and 80% of the dataset is used for training and the rest is used for testing.

The support vector machine (SVM) with linear Kernel function is used. The final classification accuracy is 99.1%.

3. 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?

The sliding window search is realized in ‘find_cars” function.

The function defines an image area of interest through “y_start” and “y_end”. The HOG features were extracted for each channel of this specific area. Sub-sampling is then used to extract small sub-images with HOG features known. The color features are also extracted for the small sub-images. All these features can then be used as the input of the trained SVM classifier.

The “y_start” is set as 400, the “y_end” is set as “656”. The scale is set as 1.5. The parameters used in the extraction of HOG features and color features are the same as previously introduced. These parameters are selected through the performances on the six testing images. The worst case I experienced is that no vehicles can be detected at all.

4. Show some examples of test images to demonstrate how your pipeline is working. How did you optimize the performance of your classifier?

The following are two examples of vehicle detection:

For the optimization of the classifier, one way is to adjust the hyper-parameters, another way is to introduce more data. First I tried the smaller dataset with 1196 vehicles and 1125 non-vehicles, the classifier can’t detect the vehicles in the testing images shown above. With the bigger dataset, the classifier performs much better.

5. 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 final video output can be found in the GitHub repository.

6. 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.

There are false positives and overlapping for the bounding boxes. The approach to solve this is to build a heat map and set up a threshold. Only when the number of rectangles is greater than the threshold, they are merged as one box for that vehicle. The functions built for this are “add_heat”, “apply_threshold” and “draw_labeled_bboxes”. The threshold is set as 1 in this project.

The six testing images for vehicle detection with FPs and duplication removed are shown as following:

7. 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?

We need to use the bigger vehicle and non-vehicle dataset. The classifier trained using the smaller dataset can't detect the vehicles in the testing images and videos well.

The pipeline may fail when the two vehicles around the self-driving car are overlapping. It may also detect the vehicles from the other direction if the threshold is set too low. On the other hand, if the threshold is set too high, some vehicles of interest can't be detected.

To make it more robust, more training data can be introduced. The classifiers from different algorithms (SVM, deep learning, classification trees and so on) can be applied together.