Writeup Report

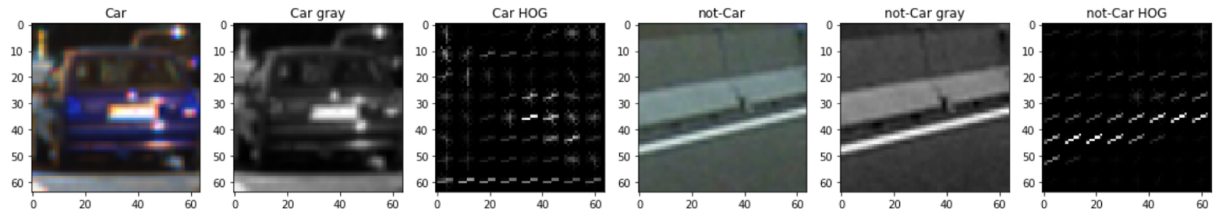
The notebook is divided into three major sections based on the notebook that follows the rubric points: 1. Histogram of Oriented Gradients (HOG) 2. Sliding Window Search and 3. Video Implementation. Let's dive into details.

1. Histogram of Oriented Gradients (HOG)

This section shows how I prepare data `[code cell 1]`, extract features, and train the classifier to track a vehicle.

In the extraction process, I first chose a **color space**, and then I computed and applied both color (**spatially binned color** and **color histograms**) and **HOG features**. Color related functions are in `code cell 3` while the HOG function and a test of it are in `code cell 4`. Here is an example of a car and a non-car images converted to grayscale before and after HOG.

```
Test index: 5381
```



The entire **feature extraction** process is stated in `code cell 5`, and `cell 2` contains all tuning parameters. These parameters such as

```
COLOR_SPACE = 'YCrCb'    # Can be RGB, HSV, LUV, HLS, YUV, YCrCb
HOG_CHANNEL = 'ALL'      # Can be 0, 1, 2, or "ALL"
SPATIAL_SIZE = (8, 8)    # Color: spatial binning dimensions
HIST_BINS = 16           # Color: number of histogram bins
ORIENT = 9               # HOG orientations
PIX_PER_CELL = 8         # HOG pixels per cell
CELL_PER_BLOCK = 2       # HOG cells per block
```

are chosen based on the class examples and my tuning experiments.

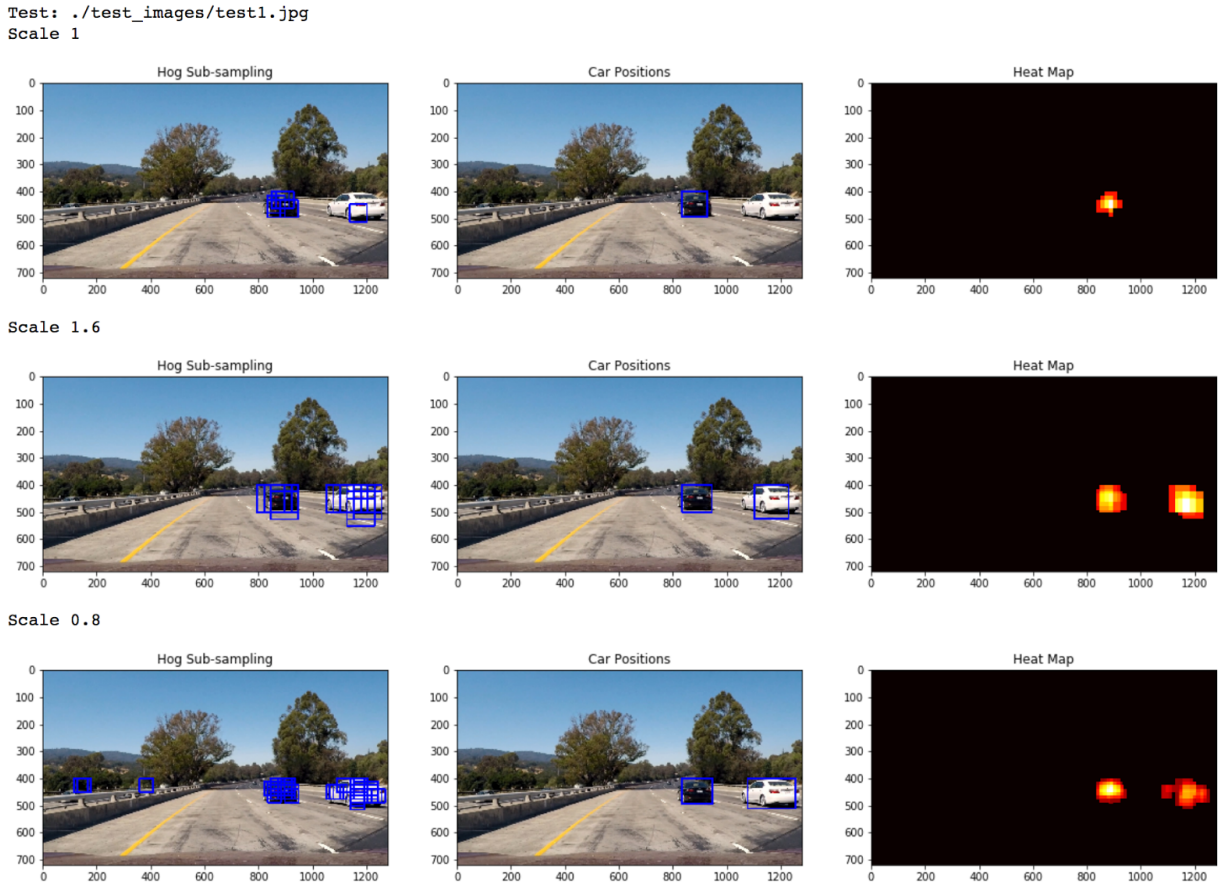
In `code cell 6`, I built a **classifier** to detect if an image has a car in it or not. Before training the classifier, I normalized the feature vectors of my training data as well as randomized all image data as they came from time-series videos. After these preprocessing, I applied a linear svc `LinearSVC()` to train a classifier. Here is the prediction result.

```
Using: 9 orientations 8 pixels per cell and 2 cells per block
Feature vector length: 5532
Linear SVC =>
13.15 Seconds to train SVC...
Test Accuracy of SVC =  0.9913
My SVC predicts:   [ 0.  1.  0.  0.  1.  0.  0.  1.  0.  0.]
For the 10 labels: [ 0.  1.  0.  0.  1.  0.  0.  1.  0.  0.]
0.00217 Seconds to predict 10 labels with SVC
```

2. Sliding Window Search

This section contains the pipeline of how I detected vehicles in two major steps: **Hog Sub-sampling** that finds cars in an image, and **Heatmap & False Positives** that filters out false positives using a heat-map.

The `find_cars` function in `code cell 7` extracts hog features once only and then it can be sub-sampled to get all of its overlaying windows. As mentioned in the class, each window is defined by a scaling factor where a scale of 1 would result in a window that's 8 x 8 cells then the overlap of each window is in terms of the cell distance. This means that a `cells_per_step = 2` would result in a search window overlap of 75%. The first column of the outputs below demonstrate examples that run this function multiple times for three different scale values `scale = 1, 1.6. 0.8`. This technique will be used to generate multiple-scaled search windows later.

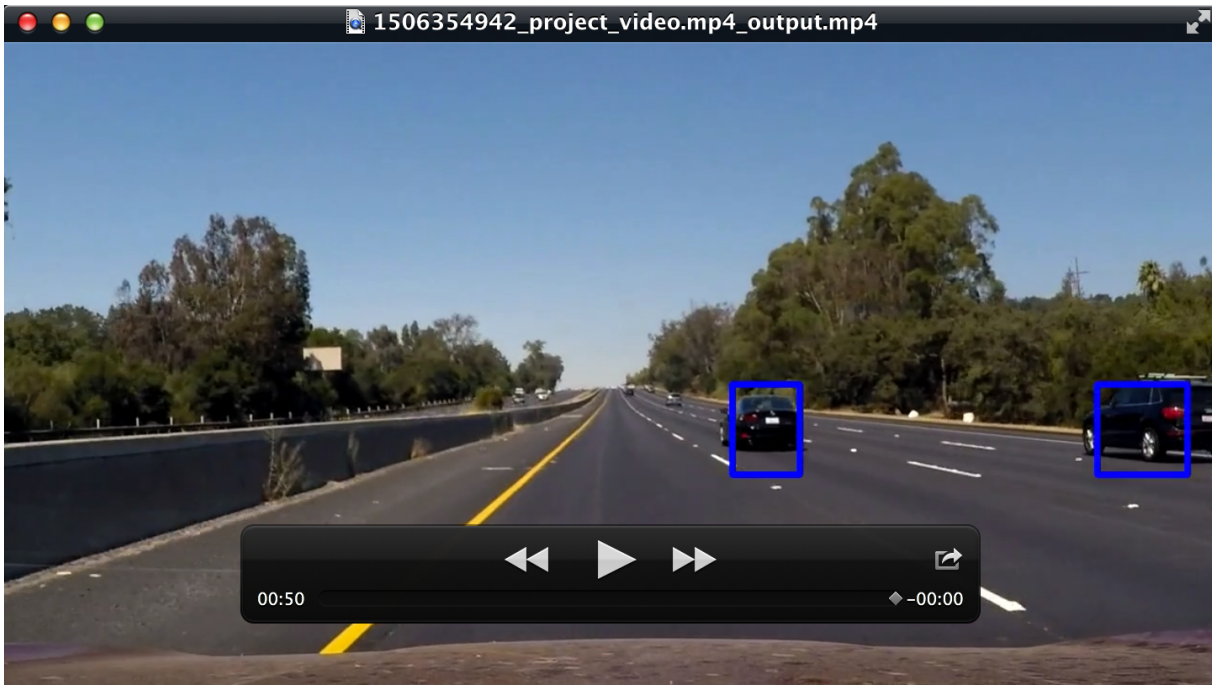


In the first column, we can find overlapping detections exist for each of the two vehicles, and in the frame of scale 0.8, I find some false positive detections on the guardrail to the left. To reduce false positive, I added search boundaries `X_START_STOP` and `Y_START_STOP` to skip detection on part of trees and the sky. The major filter step is in `code cell 8` where I built a heat-map from these detections in order to combine overlapping detections and remove false positives with a `THRESHOLD_HEAT`. The result is shown in column 2 and 3 above.

Note that for the video implementation, in addition to the two steps that I just mentioned, I also take multiple frames into consideration. The complete pipeline implementation and tuning parameters is in `code cell 9 and 10`.

3. Video Implementation

Here's a link on YouTube to my video result.



In the final video, I tuned search boundaries, scale of subsampling, threshold of heapmap and number of frames to optimize vehicle detection. The introduction of considering 6 frames is to average bounding area as well as reduce false positive detection. The parameters used in the video are as follows:

```
X_START_STOP = [[230, None], [600, 980]]
Y_START_STOP = [[380, 650], [390, 460]]
SCALE = [1.5, 0.8]

NUM_FRAME = 6
THRESHOLD_HEAT = 7
```

Discussion

The tuning of the parameters could lead to overfit for this particular video especially those fine tuning of the scale and boundaries. These are tuned either for experiments or based on the assumption that the car is driving on the similar road environment.