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.

Histogram of Oriented Gradients (HOG)

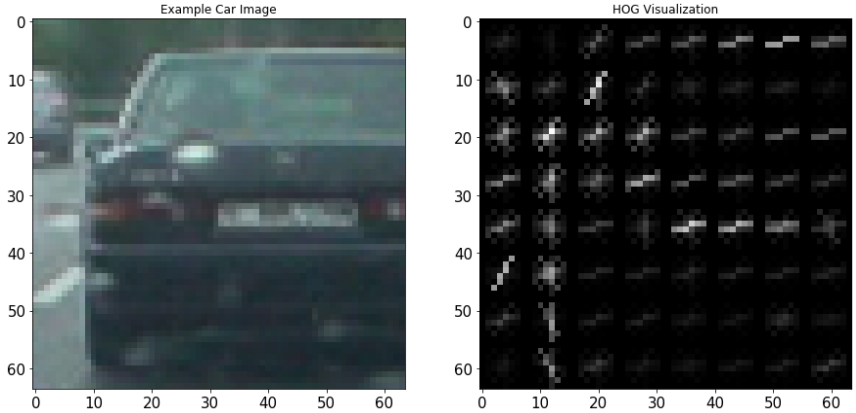
####1. Explain how (and identify where in your code) you extracted HOG features from the training images.

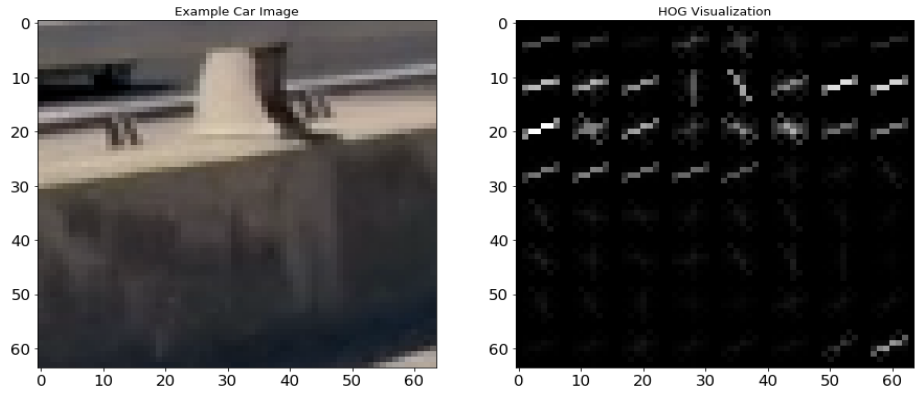
The code for this step is contained in the first code cell of the IPython notebook (get_hog_features)

I started by reading in all the `vehicle` and `non-vehicle` images. Here is an example of pictures of the `vehicle` and `non-vehicle` classes:

I then explored different color spaces. grabbed random images from each of the two classes and displayed them to get a feel for what the `skimage.hog()` output looks like.

Here is an example using the RGB color space and HOG parameters of `orientations=8`, `pixels_per_cell=(8, 8)` and `cells_per_block=(2, 2)`:

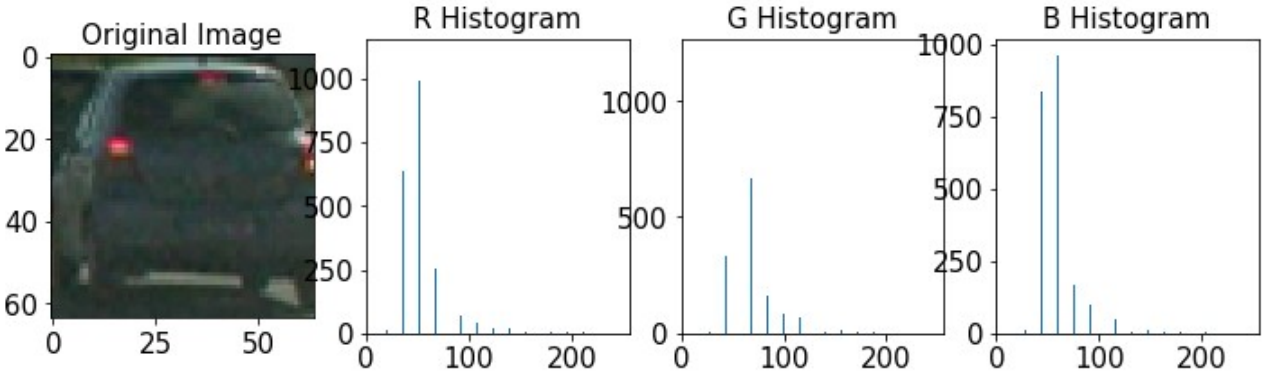




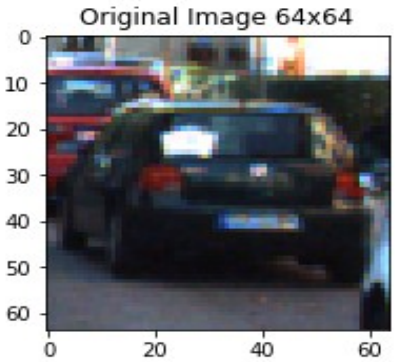
2. Define different color choices.

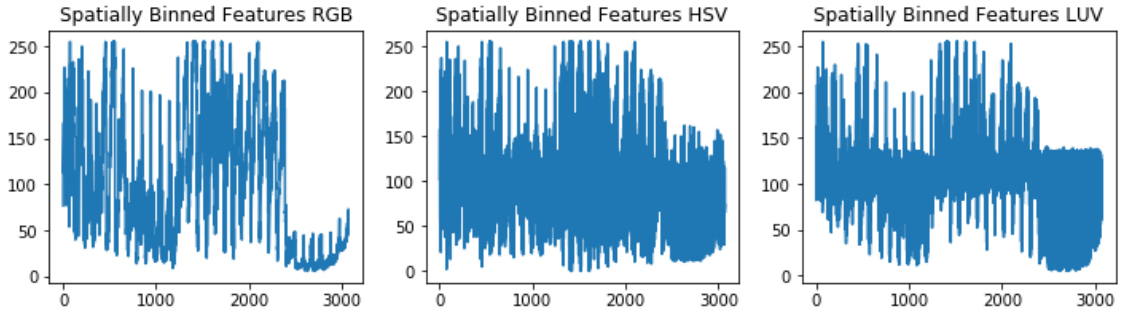
I tried various color features functions.

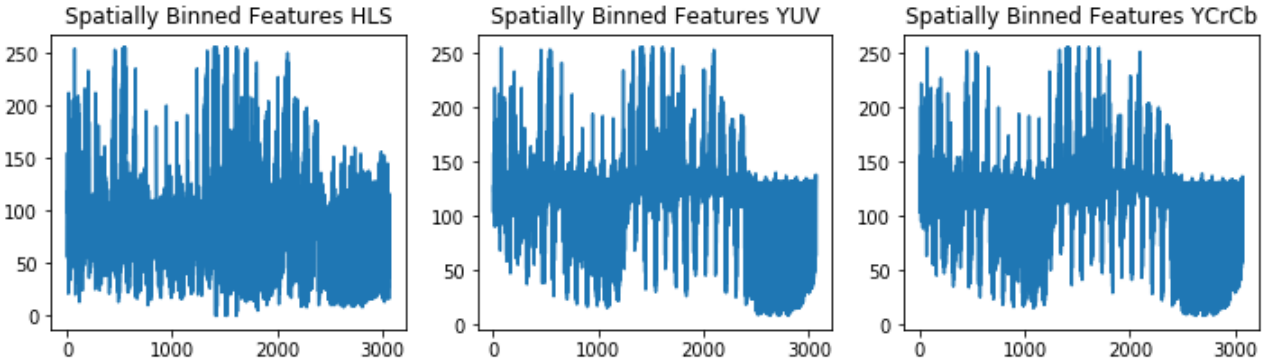
(1) compute Color Histogram features and visualizing the results

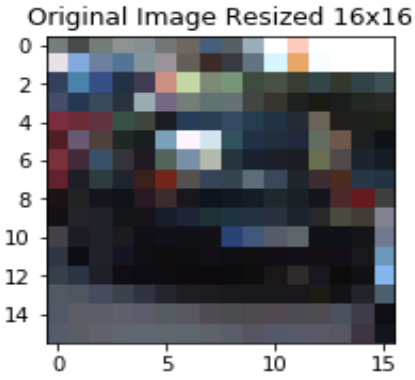


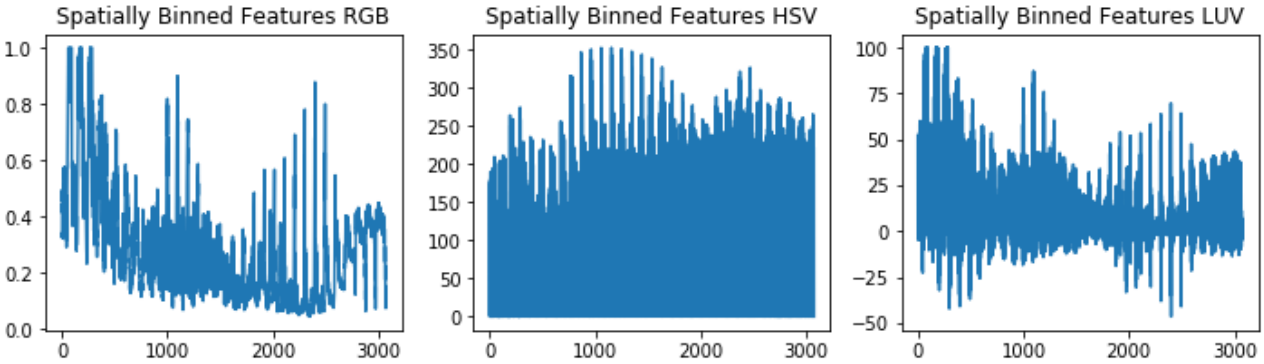
(2)computer Spatial Binning of Color features with different color space(HLS, HSV, LUV, YUV, YCrCb) and different size(32*32, 16*16) and visualizing the results

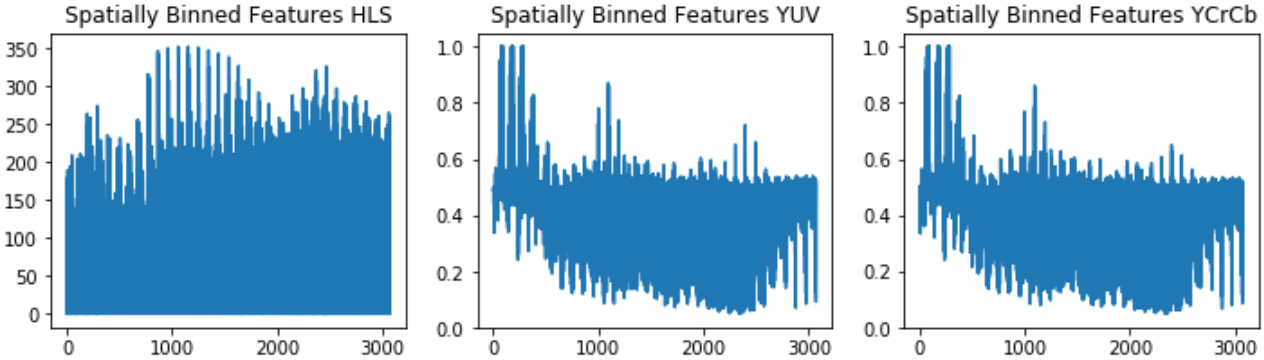












3. Describe how (and identify where in your code) you trained a classifier using your selected HOG features (and color features if you used them).

(1). I trained a linear SVM using HOG Support Vector Classifier and color space “YcrCb”,

the accuracy is Test Accuracy of SVC: 97.24%

```
  Predictions: [ 0.  0.  1.  1.  0.  1.  1.  1.  1.  1.]
       Labels: [ 0.  0.  1.  1.  0.  1.  1.  1.  1.  1.]

0.00189 seconds to predict 10 labels with SVC.
```

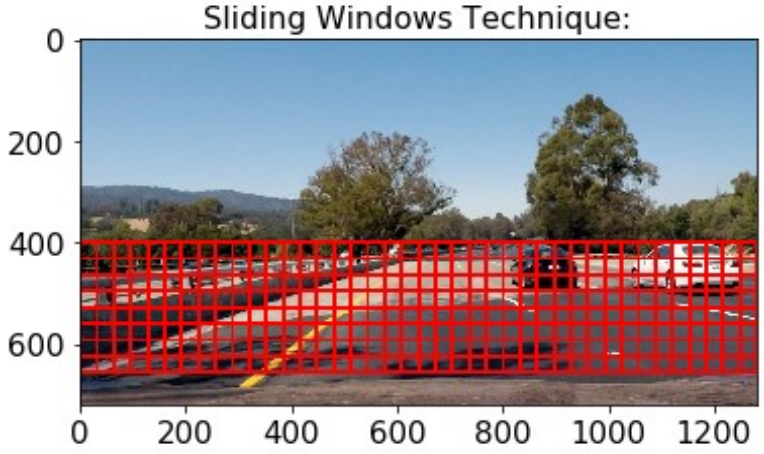
(2). I trained a linear SVM using Color Histogram Support Vector Classifier and color space “YcrCb”,

```
Test Accuracy of SVC: 97.58%
  Predictions: [ 1.  0.  0.  1.  0.  1.  1.  0.  1.  0.]
       Labels: [ 1.  0.  0.  1.  0.  1.  1.  0.  0.  0.]

0.00238 seconds to predict 10 labels with SVC.

After test different combination of color space of parameters of hog the final
variables I use are:
color_space = 'YCrCb' # Can be RGB, HSV, LUV, HLS, YUV, YCrCb
orient = 10  # HOG orientations
pix_per_cell = 8 # HOG pixels per cell
cell_per_block = 2 # HOG cells per block
hog_channel = 'ALL' # Can be 0, 1, 2, or "ALL"
spatial_size = (32, 32) # Spatial binning dimensions
hist_bins = 64    # Number of histogram bins
spatial_feat = True # Spatial features on or off
hist_feat = True # Histogram features on or off
hog_feat = True # HOG features on or off
Test Accuracy of SVC = 99.01%
```

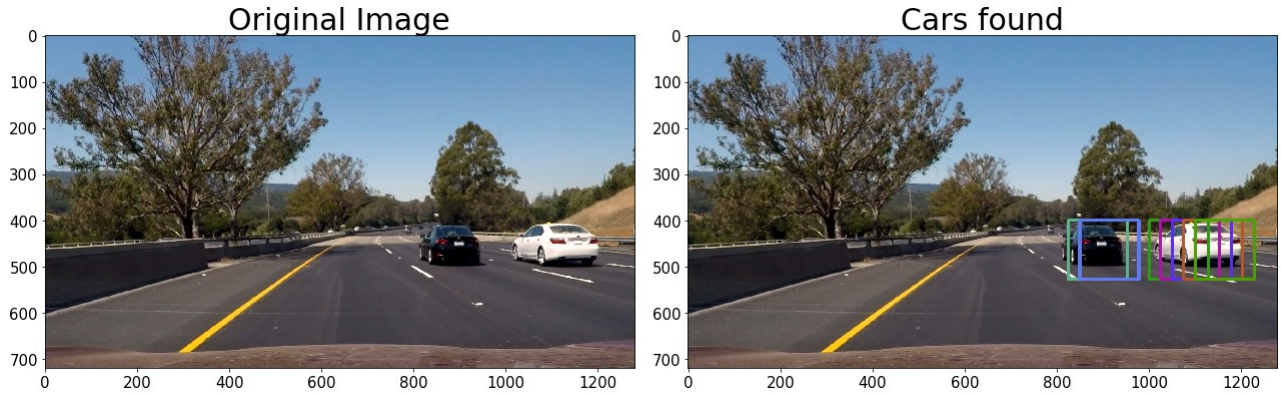
###Sliding Window Search

####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?

I decided to search random window positions in y_start_stop=[400, 656], at scales xy_window=(64, 64),

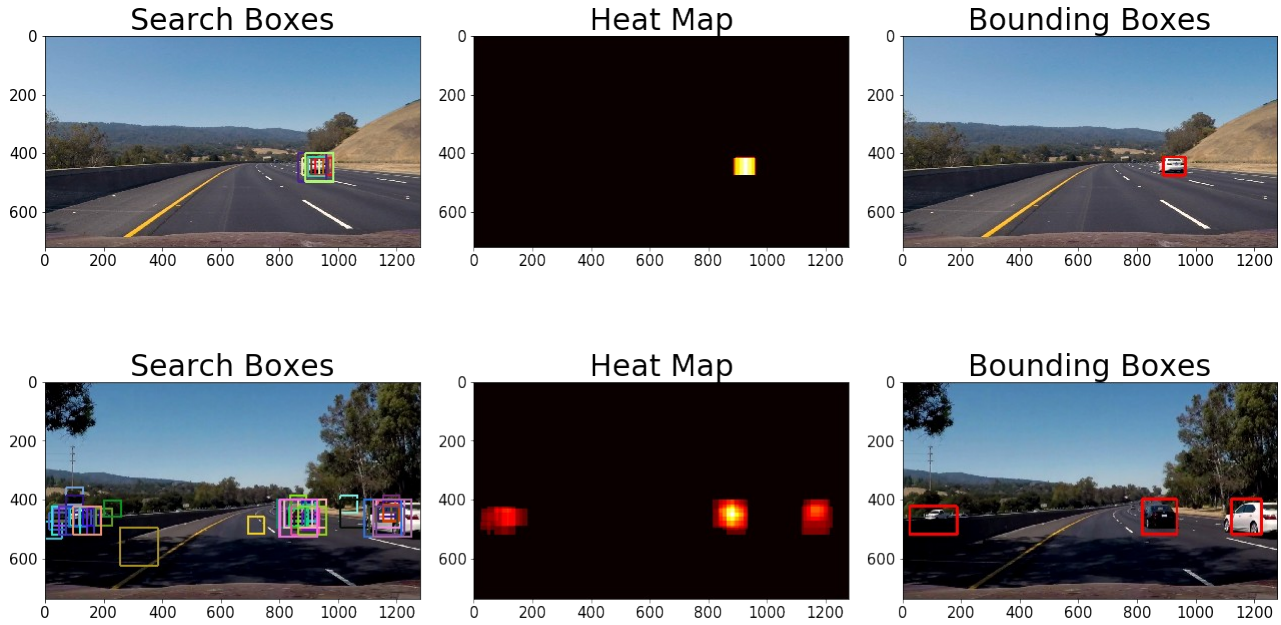


##########2 I test the windows on the test images:



####3. 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 four scales using YCrCb 3-channel HOG features plus spatially binned color and histograms of color in the feature vector, which provided a nice result. Here are some example images:



####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 thresholded (apply_threshold(heat, 2)) that map to identify vehicle positions. I then assumed each blob corresponded to a vehicle. I constructed bounding boxes to cover the area of each blob detected (draw_labeled_bboxes(np.copy(img), labels).

###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?

I find there are two difficults in this project. First, To detect the vehicle. I need to anaylsis more combinations to find a more accurate detection. Because When I apply the find_cars function, there are lots of wrong detections. Secondly, I applied four scales to tracking, I tried to use 3 scales, find some cars could not detected, while when I used 5 scales, it results in lots of wrong detections.