# Vehicle Detection Project- 5

**The goals / steps of this project are the following:**

1. Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images and train a classifier Linear SVM classifier
2. 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.
3. Implement a sliding-window technique and use your trained classifier to search for vehicles in images.
4. 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.
5. Estimate a bounding box for vehicles detected.

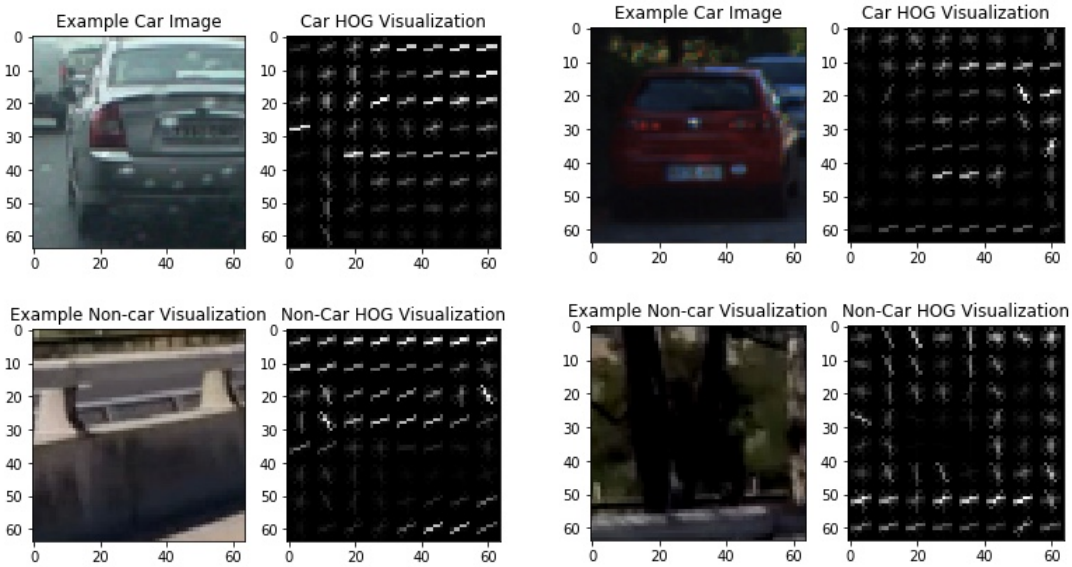
---

# 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 code cell #1.I started by unzipping and reading in all the vehicle and non-vehicle images as provided in the Rubric.
I've got:

Number of vehicle: 8792
Number of non-vehicles: 8968

I then grabbed random images from each of the two classes and displayed them to get a feel for what the skimage.hog() output looks like. The below example uses the HOG parameters of orientations=9, pixels_per_cell=(8, 8), cells_per_block=(2, 2), and hog_channels = "ALL":



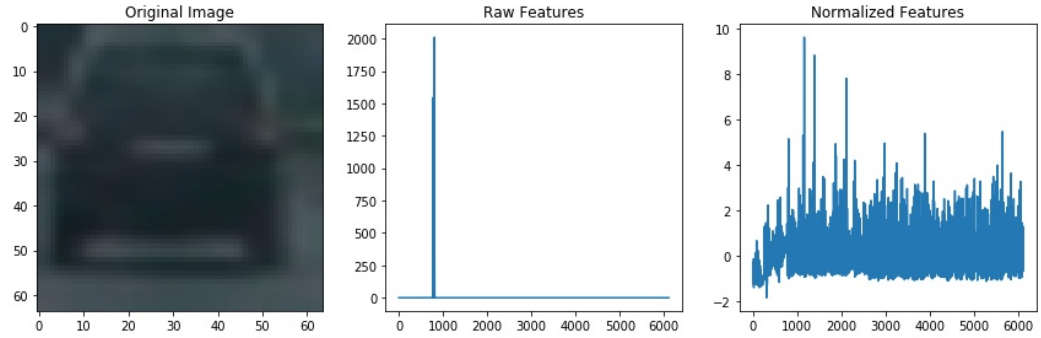

**2. Explain how you settled on your final choice of HOG parameters.**
The below code-cell lists all the various types of functions used to extract features from the images which are to be combined and normalized for training a Linear SVM.
After experimenting various combinations of features, and HOG parameters, I chose the following HOG parameters for my final classifier:
color space = 'YCrCb'
orientations = 9
pix_per_cell = 8
cell_per_block = 2
hog_channel = "ALL"

I chose these parameters based on its test accuracy and overall performance. The following features were chosen, and combined into a feature vector.

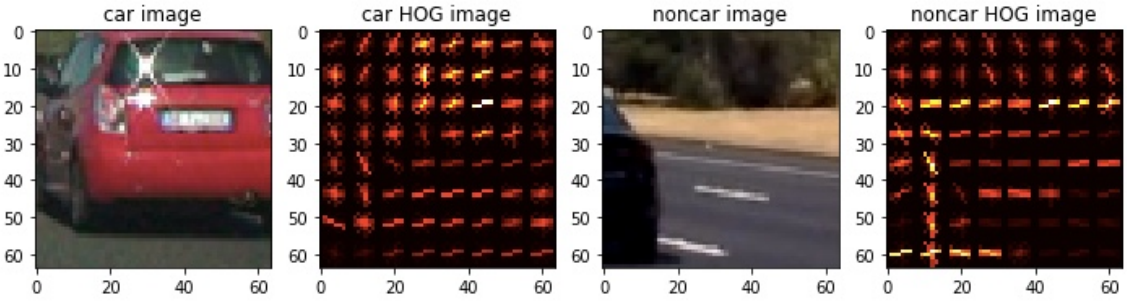
spatial_size = (32, 32)
# Spatial binning dimensions hist_bins = 32
# Number of histogram bins spatial_feat = True
# Spatial features on or off hist_feat = True
# Histogram features on or off hog_feat = True
# HOG features on or off

**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 function 'train' is used to train a Linear SVM model with the parameters and 2000 images mentioned above using HOG, Spatial, and histogram features. This features are also normalized before training the model. As you can see below, the accuracy of the model is 99.25%.

```
Using: 9 orientations 8 pixels per cell and 2 cells per block
Feature vector length: 6108
35.64 Seconds to train SVC...
Test Accuracy of SVC =  0.9925
My SVC predicts:  [ 0.  1.  0.  0.  1.  0.  0.  1.  1.  1.]
For these 10 labels:  [ 0.  1.  0.  0.  1.  0.  0.  1.  1.  1.]
0.01142 Seconds to predict 10 labels with SVC
```




Also shown below is a randomly selected image of a car along with its Raw and Normalized 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?**
The below 'find_cars' method (based on the method from the lessons) is used to identify cars in a given image/video frame using the SVM classifier trained earlier. This method returns boxes based on the scales that are sent to it as parameters. An overlap of 2pixels is set. The images following the codecell below shows the output. These are the scales used for the search area:

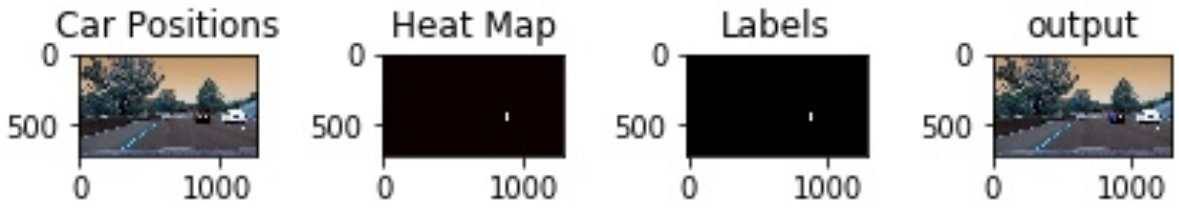
| Y-Range | Scale |
|---|---|
| 380, 510 | 1.0 |

380, 570 1.5
380, 630 2.0
380, 690 2.5

From the output, its noticeable that the function does return false-positives. In the later heatmap section, I use the heatmap thresholds to filter out false positives.

# Heat Map

The Code-cell below generates a heatmap of all the detections from the 'find_cars' method. I created a heatmap 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 output below shows the heatmap of a test image.




# 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.)**

The cells below generate the video output of the entire processing pipeline.

**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.**

As described above, from the positive detections I created a heatmap 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.

# 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?**

As noticed in the final video, the pipeline could use a smoothening feature to track the cars. The boxes can be made to move uniformly. The pipeline also identifies cars in the oncoming lane (Maybe a good or bad things based on the scenario).

Another improvement that I plan to implement is to try out other classifiers(NN based) and also implement decision tress to better filter out false positives.