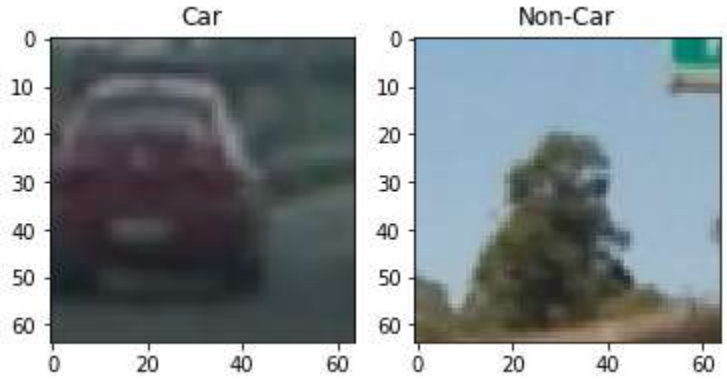

Vehicle Detection Project

The goals / steps of this project are the following:

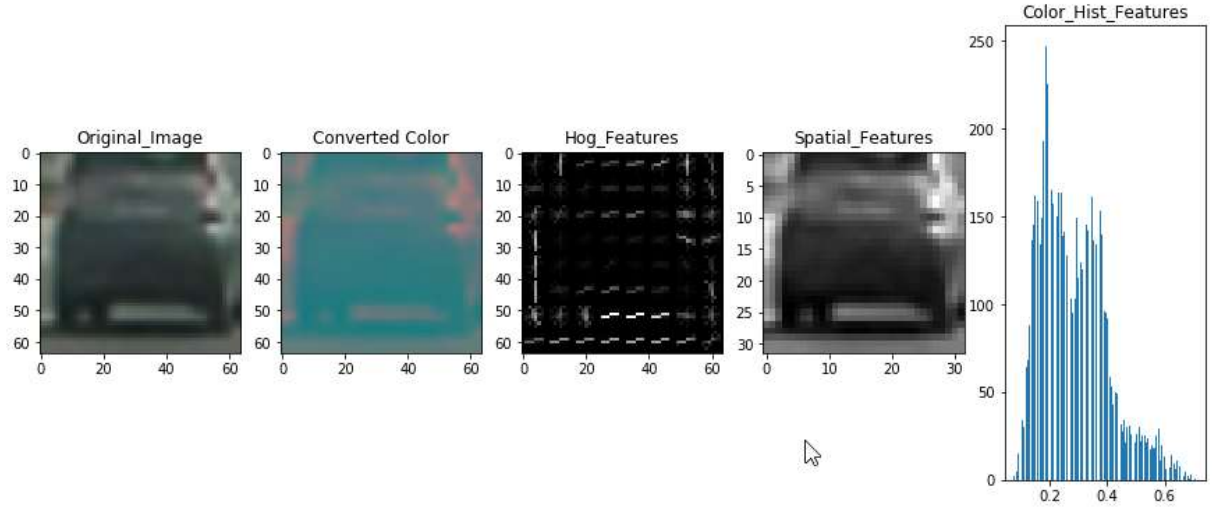
- Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images and train a classifier Linear SVM classifier
- Optionally, you can also apply a color transform and append binned color features, as well as histograms of color, to your HOG feature vector.
- Note: 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.

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

I started by reading all car and non car images from the database downloaded from the Udacity website/
The code for this step is contained in the first code cell of the IPython notebook (or in lines # through # of the file called `some_file.py`).
Here's an example of an image from each set.



I then explored different color spaces and different `skimage.hog()` parameters (`orientations`, `pixels_per_cell`, and `cells_per_block`). I grabbed a random image from the car set and tested color spaces, hog feature function, bin_spatial function and also a color histogram . The results are as below



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

I tried various combinations of parameters color spaces and using single color space in hog and finally settled in using all the color channels for HOG which gave a significant boost to detection.

####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).

I trained a linear SVM using all the car and non car images from all the folders from the udacity website. I fit a StandardScaler to normalize the different feature values. At the end I landed with around 8640 features with the help of which I trained my model. I got an accuracy of 99.16% while testing

###Sliding Window Search

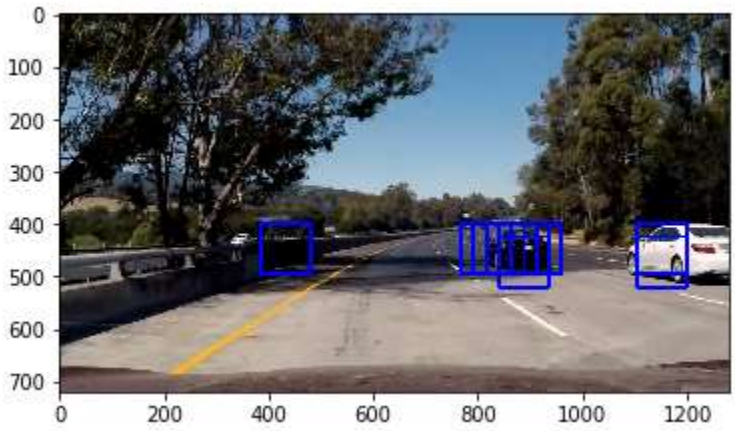
####1. Describe how (and identify where in your code) you implemented a sliding window search. How did you decide what scales to search and how much to overlap windows?

I first implemented a get hog feature function on the entire image and then defined the ystart and ystop in the visible region of the road. With a window size of 64 by 64, I stepped through the search region by 2 cells and extracted the hog features for that

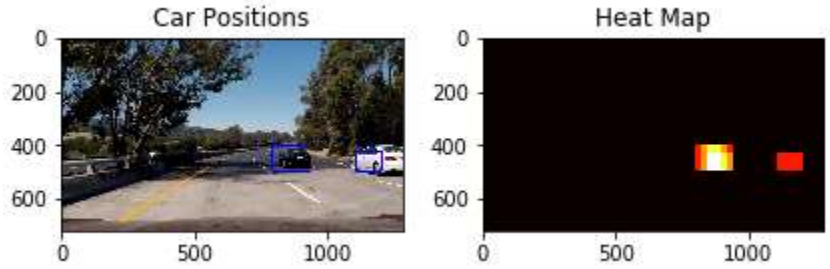
region along with spatial and histogram features. After combining these features, if the prediction is true, I drew a rectangle around the features

####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 two scales using YCrCb 3-channel HOG features plus spatially binned color and histograms of color in the feature vector, which provided a nice result. Heres an example of the feature extraction



After feature extraction, I found out that a heat map is necessary to remove the false positives from the prediction. The results from the heat map and resulting predicted image is as follows:



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.)

####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.

With the inclusion of a heat map mask of the image, I defined a class 'car'. I appended heat maps for consecutive 15 frames and defined a threshold of 6 frames . Only cars identified in atleast 6 frames are identified as True Positives and rectangle boxes are drawn around them. That defined the pipeline of the image processing required for vehicle detection

---

###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?

My pipeline still detects some of the cars in the opposite lane which is not necessarily a bad thing as the direction in which they travel can be used to detect cars moving in opposite lane. One of the major improvements I could do is to determine the yellow line (divider) to identify the left most boundary of the search window so that cars in the opposite lane are not identified or they could be identified as cars in the other lane.

I could also add the lane detection algorithm to the pipeline.