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.

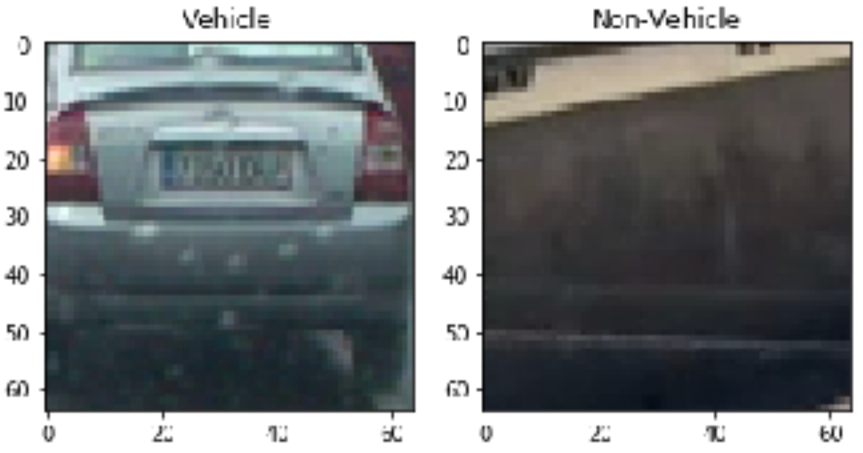
Writeup / README

Histogram of Oriented Gradients (HOG)

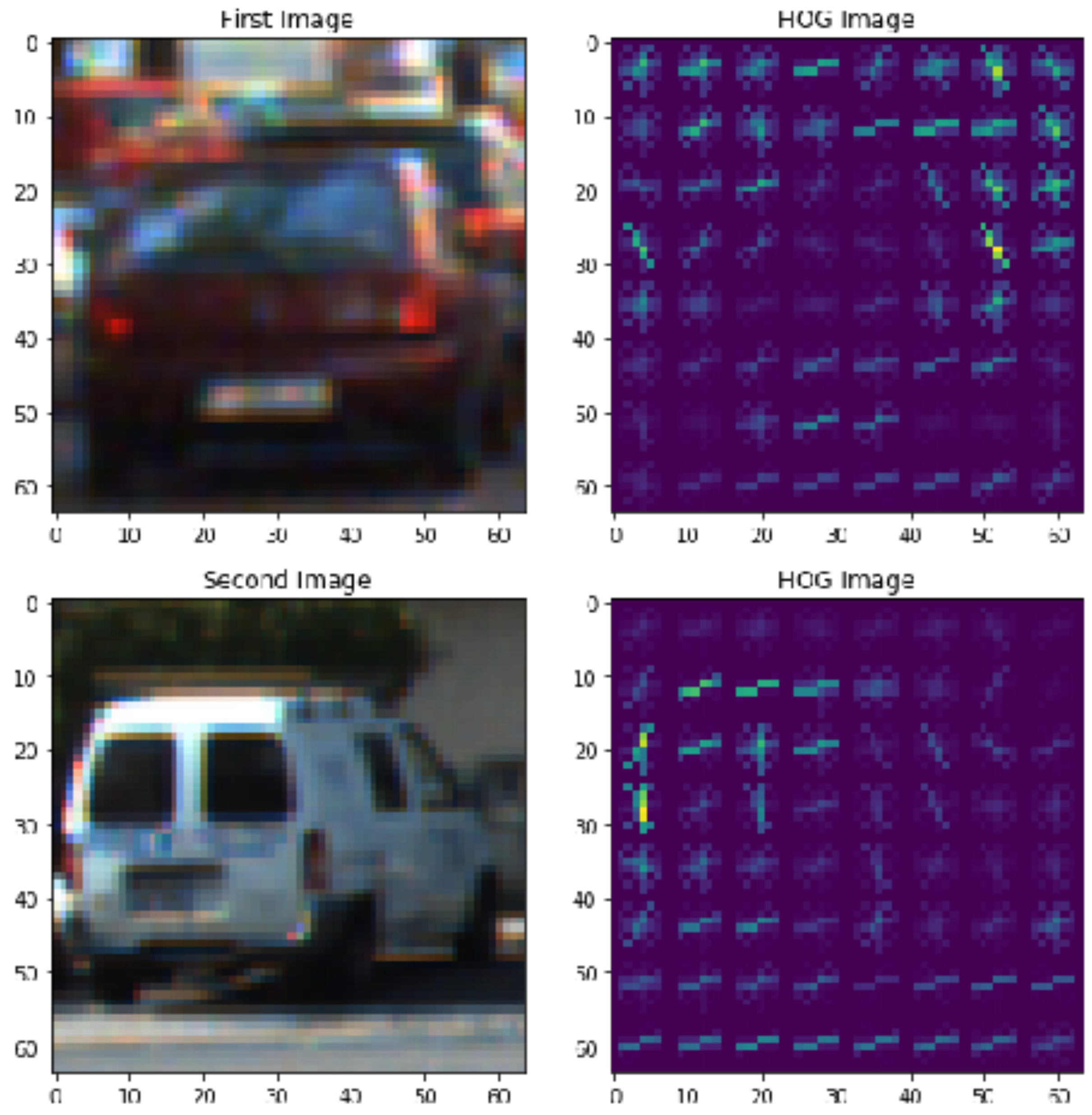
1. Explain how (and identify where in your code) you extracted HOG features from the training images.

- The code for this step is contained in the 5th code cell of the IPython notebook.

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



- I then explored different color spaces and different skimage.hog() parameters (orientations, pixels_per_cell, and cells_per_block). I 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 YUV color space and HOG parameters of orientations=9, pixels_per_cell=(8, 8) and cells_per_block=(2, 2)



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

I tried various combinations of parameters and found linear SVM classifier test set accuracy of 99.1% for following combination of HOG parameters. Below HOG params and color gave best results on the test images and test video.This experiment was done in the Udacity provided course material quiz section.

colorspace = 'YUV'
orient = 9
pix_per_cell = 8
cell_per_block = 2
hog_channel = “ALL"

It took 86.25 seconds to extract cars(8792 images) and notcars(8968 images) features includes both HOG and colour features, histogram and spatial bins of colours using number of bin = 32 and spatial size = (32,32) respectively.

It took 11.4 seconds to train the linear SVM classifier.

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)

Code Cell 5-10 covers implementation of process that leads to training classifier. It includes following steps,

1. Extract HOG and color features from Images using HOG parameters mentioned above, number of histogram bins = 32 and spatial binning of colors with size (32X32) mentioned above.

2. Pythons sklearn package provides StandardScaler() method to normalise the data.To apply StandardScaler() need to first have data in the right format, as a numpy array where each row is a single feature vector.

3. I used a linear SVM classifier to train the extracted features. Training time of classifier was 11.49 secs. and SVM classifier has test accuracy of 99.1%.

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?**

Cell 12 covers Hog Sub-sampling Window Search that allows to extract the Hog features once. Find_cars fn in code cell 12 extract Hog and color features and make predictions.

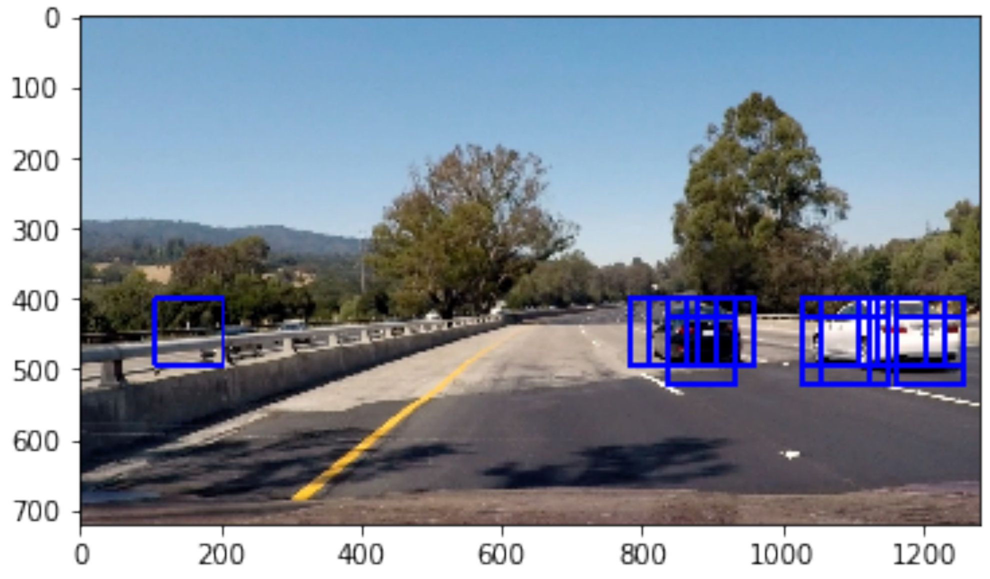
The find_cars fn. has to extract hog features only once and then can be sub-sampled to get all of its overlaying windows. Each window is defined by a scaling factor where a scale of 1 would result in a window that's 8 x 8 cells then the overlap of each window is in terms of the cell distance. This means that a cells_per_step = 2 would result in a search window overlap of 75%.

I have used two scaling factors 1.5 and 0.95 to generate multiple-scaled search windows for better predictions.

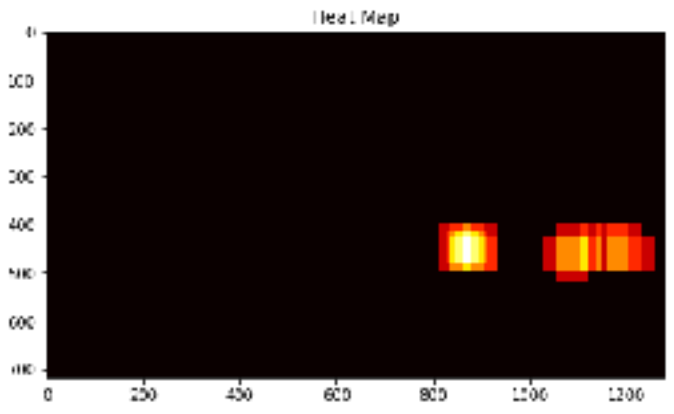
To capture distant images of vehicle (bw y value of 400-480) small scale would be required to bring zoom out effect and predictions are accurate and that is why 0.95 is used to cover such cases. For close vehicles, 1.5 scale is used.

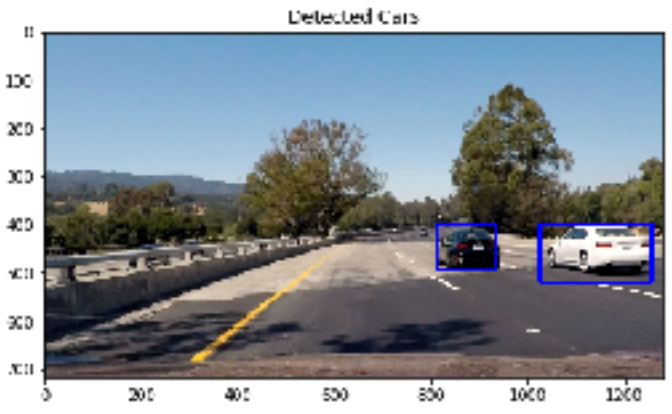
2. Show some examples of test images to demonstrate how your pipeline is working. What did you do to optimise the performance of your classifier?

Below image shows the boxes that are predicted as vehicle by linear SVM classifier



Heat Map is generated after false positive is removed and combined bounding boxes to generate final boxes representing detected vehicles.





To optimise performance of the classifier, different subchannels of various colour spaces were tried like YUV, HLS, HVS, GRAY etc. with HOG parameters kept constant orient = 9, pixels per cell = 8, cell per block =2.

I tried YUV with orient = 12, pixels per cell 10 cells per block 2. classifier accuracy jumped above 99.5 but did not turned out well on test video.

Thus, YUV(ALL channels) was chosen with orient = 9, pixels per cell = 8, cell per block =2 and it gave better results on test video.So these HOG parameters were kept to train the classifier.

Also, increased the training dataset by considering GTI images along with KITTI images. This gave well balanced dataset for cars and notcars labels.

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.)

Output_Project_video.mp4 is present in the project repository.

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.

Heatmap and Thresholding is used to combine the bounding frames detected as vehicle and to remove the false positives. label()

function is used to identify number of vehicles in the image. These functions are defined in cell 13.

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 found only a small portion of vehicles being detected by the pipeline. This was improved by increasing the training dataset.

Also I found performance of my pipeline was very slow. This was improved by making visualise flag and vector flag as false in the hog() method of extracting HOG features.

There are changes required to make the vehicle detection algorithm more robust, input being multiple scales. Also vehicle detection is limited to training dataset used to fit the classifier.

There is improvement required to make the detection smoother. This can be done as an improvement in the next iteration by taking average of boxes of last few frames as resultant box of the current frame.