Vehicle Detection and Tracking

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.

Note :

Ipython notebook is uploaded here :
https://github.com/linux-devil/vehicle-detection-tracking/blob/master/vehicle-detect.ipynb

Datasets:

I use two datasets. First is project dataset. It is splitted into cars images and non-car images. Here is examples of dataset images:

```
dataset has cars: 8792
none cars: 8968
```

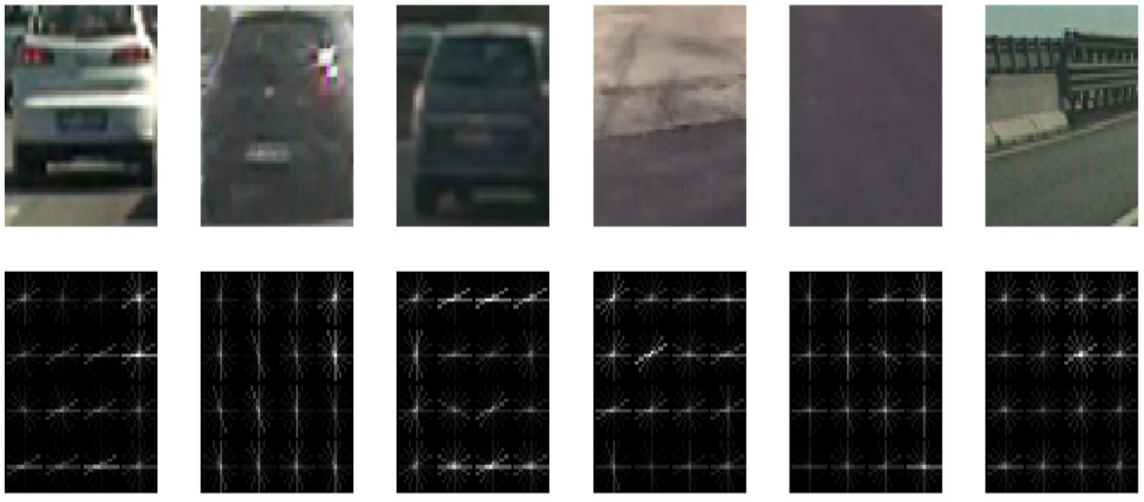
Histogram of Oriented Gradients (HOG)

After playing with picture pixels, histogram and HOG features I decided to use only little amount of HOG features. I tried various combination of HOG parameters and finally selected on following setting which gave better SVM classification accuracy.

Following are my parameters :

1. color_space = 'YUV' # Can be RGB, HSV, LUV, HLS, YUV, YCrCb
2. orient = 9 # HOG orientations
3. pix_per_cell = 8 # HOG pixels per cell
4. cell_per_block = 2 # HOG cells per block
5. hog_channel = 0 # Can be 0, 1, 2, or "ALL"
6. spatial_size = (16, 16) # Spatial binning dimensions
7. hist_bins = 32   # Number of histogram bins
8. spatial_feat = True # Spatial features on or off
9. hist_feat = True # Histogram features on or off
10. hog_feat = True # HOG features on or off

Following is hog image for car and non cars:



SVM classifier

Decided to use SVM classifier with rbf kernel and default sklearn parameters as linear kernel gave slightly less accuracy.

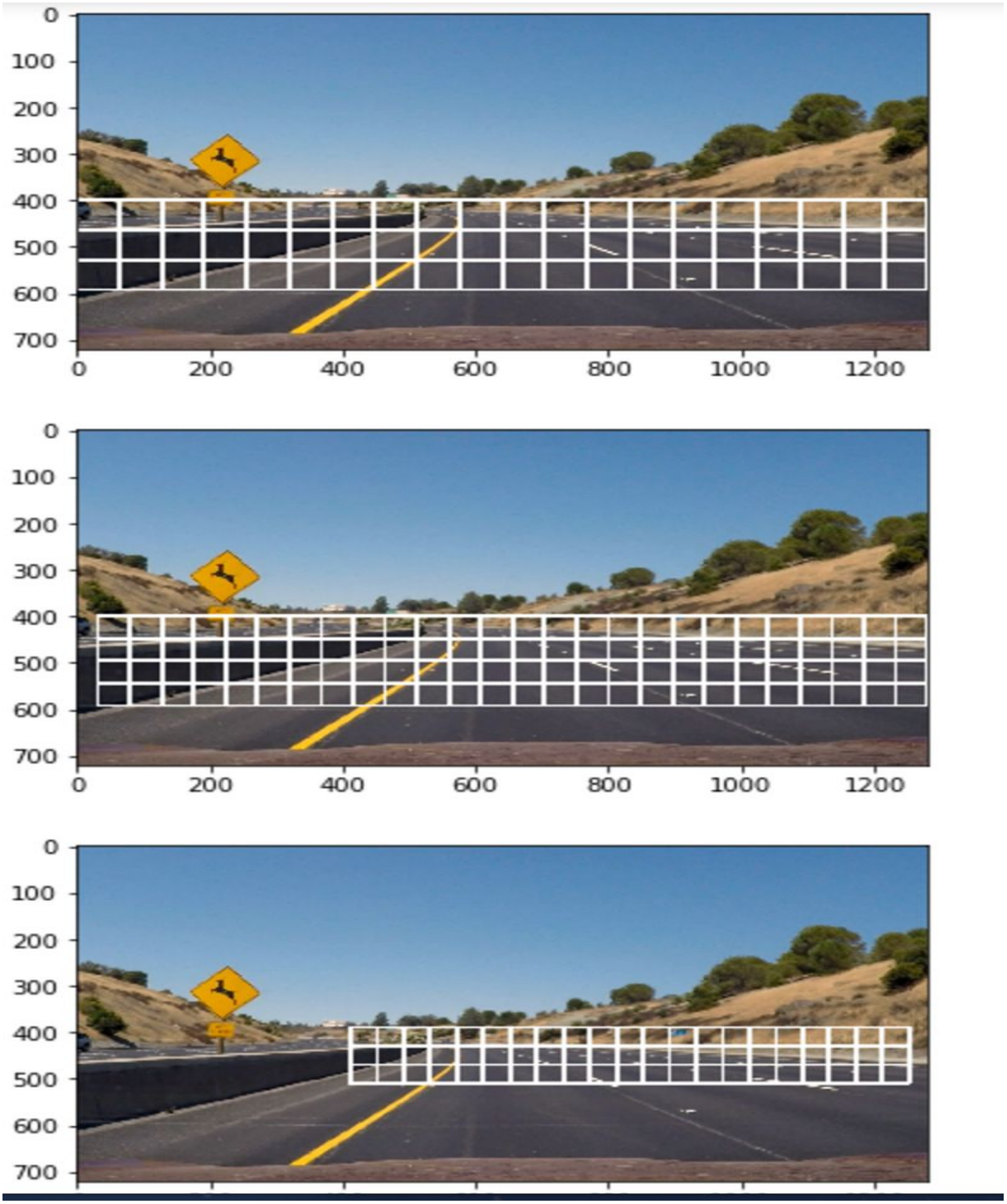
Using: 9 orientations 8 Color space YUV pixels per cell 2 cells per block
Feature vector length: 2628
Test Accuracy of SVC = 98.9

I also tried adding flipping multiple images in the samples and following is the accuracy obtained :

Car samples: 17584
Notcar samples: 17936
Using: 9 orientations 8 Color space YUV pixels per cell 2 cells per block
Feature vector length: 2628
16944.64 Seconds to train SVC...
Test Accuracy of SVC = 0.997

Sliding windows

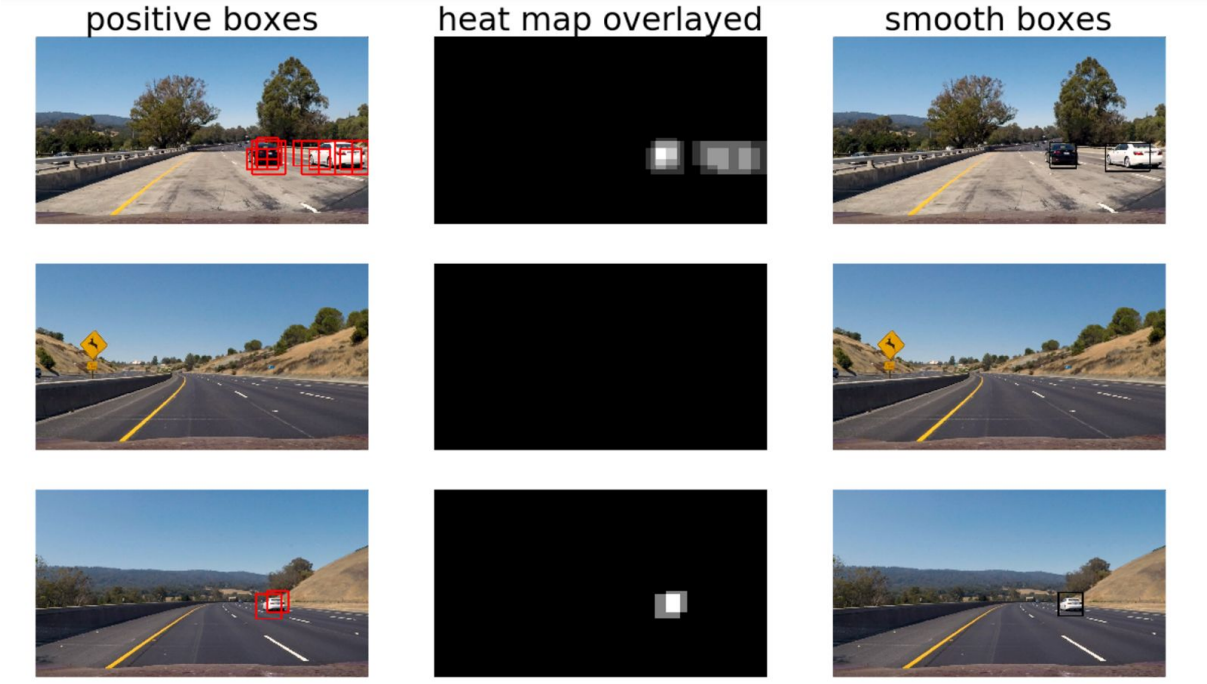
I use sliding window technique for searching cars in an input image. Iterating over image area that could contain cars with approximately car sized box and try to classify whether box contain car or not. As cars may be of different sizes due to distance from a camera we need a several amount of box sizes for near and far cars. I use 3 square sliding window sizes of 128, 96 and 80 pixels side size. While iterating I use 50% window overlapping in horizontal and vertical directions.


0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

Estimation of car positions and sizes

After sliding window application we have positive boxes - sliding window positions that were classified as containing car. We need to estimate real cars positions and sizes based on this information.

We need to join overlapped boxes around peaks to convert many overlapped boxes into smaller amount of not or slightly overlapped ones. Idea is take first box (called average box) form input boxes and join it with all boxes that is close enough (here for two boxes: they need to overlap by 20% of area of any one of two) After joining two boxes we need to update average box (here just increasing size to cover both joining boxes). Loop while we are able to join furthermore. For left boxes repeat all procedure. As a result we get list of joined boxes, average boxes strengths as the count of boxes used to get a merged box, the number of boxes it was joined to.



Video processing

We just need to accumulate positive boxes over number of last frames and then apply same algorithm here with higher threshold. Function average_boxes accumulates positive boxes over last frames

average_boxes(positive_boxes, 20)

20 - Threshold count

Video link : linux-devil/vehicle-detection-tracking

Conclusion

I think this is interesting approach for starting in this field. But it is not ready for production use. I think convolutional neural network approach may show more robustness and speed. As it could be easily accelerated via GPU. Also it may let to locate cars in just one try. For example we may ask CNN to calculate number of cars in the image. And by activated neurons locate positions of the cars. In that case SVM approach may help to generate additional samples for CNN training.

Detecting cars with SVM in sliding windows is interesting method but it doesn't generalizes well and produces lot of false positives in some situations. Also sliding windows slowes computation as it requires many classifier tries per image. Again for computational reduction not whole area of input image is scanned. So when road has another placement in the image like in strong curved turns or camera movements sliding windows may fail to detect cars.