Vehicle detection

The goals / steps of this project are the following:

- Perform a Histogram of Oriented Gradients (HOG) to extract features and also use color features on a labeled training set of images and train a classifier Linear SVM classifier
- Data is normalized and selected for training and testing
- A sliding-window technique is implemented and used on trained classifier to search for vehicles in images.
- Pipeline is ran on a video which creates a heat map of recurring detections frame by frame to reject outliers and follow detected vehicles.
- Estimate a bounding box for vehicles detected.

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Writeup / README

Histogram of Oriented Gradients (HOG)

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

Features were extracted in the second cell block. I decided to use spatial, histogram and HOG features. A lot of the code is from the Udacity course.

Function for getting spatial features bin_spatial is in cell 2 lines 4-8, histogram features function color_hist is in cell 2 lines 11-19 and hog features are extracted with a function get_hog_features in cell 2 lines 21-25. Function called extract features in cell 2 lines 27-52 put these three feature groups together.

Images are read in cell 1 lines 58-59. In cell 2 lines 62-70 features are extracted from the images. In cell 3 lines 7-9 the normalization is done. 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`). After trying various parameters, the default ones gave quite good results which can be seen in cell 2 lines 55-60.

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

First I tried the features separately and got reasonable results, but I got the best results when combining the features together - spatial, histogram binning and HOG features. When testing spatial, histogram or HOG separately, then different parameters were successful, but when combining them together spatial = 32, histbin = 32, hog_channel = “ALL”, hog orient = 9, hog pix_per_cell=8 and hog cell_per_block=2 gave the best result. For example, if only HOG would have been used, meaning if I would not have used spatial and histogram features, then I would have chosen used YCrCb color space as it gave good results. But when combining HOG with spatial and histogram features, RGB gave better results than YCrCb.

I tried various combinations of parameters and the ones which gave the best results can be seen in cell 2 lines 55-60 gave the best results. The classifier got a result of 98.8% which seemed good enough for me to proceed.

3. Describe how (and identify where in your code) you trained a classifier using your selected features.

I trained a linear SVM using spatial, histogram and HOG features. Data was normalized in cell 3 lines 7-9 using StandardScaler from scikit. The data was randomized and split into training and test set in cell 3 lines 16-17 using train_test_split from scikit which makes the split by sampling randomly. The training command itself is in cell 4 line 8 where LinearSVC is fitted. The accuracy is tested on the testset in cell 4 line 12.

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?

Initial part of sliding windows was done in cell 5 lines 7-46 with function slide_window. This returns the windows from which to search for vehicles. I got good results when overlap was 75%, but this depends on a lot of variables and might differ with a different setup, meaning it is not a definite rule of thumb.

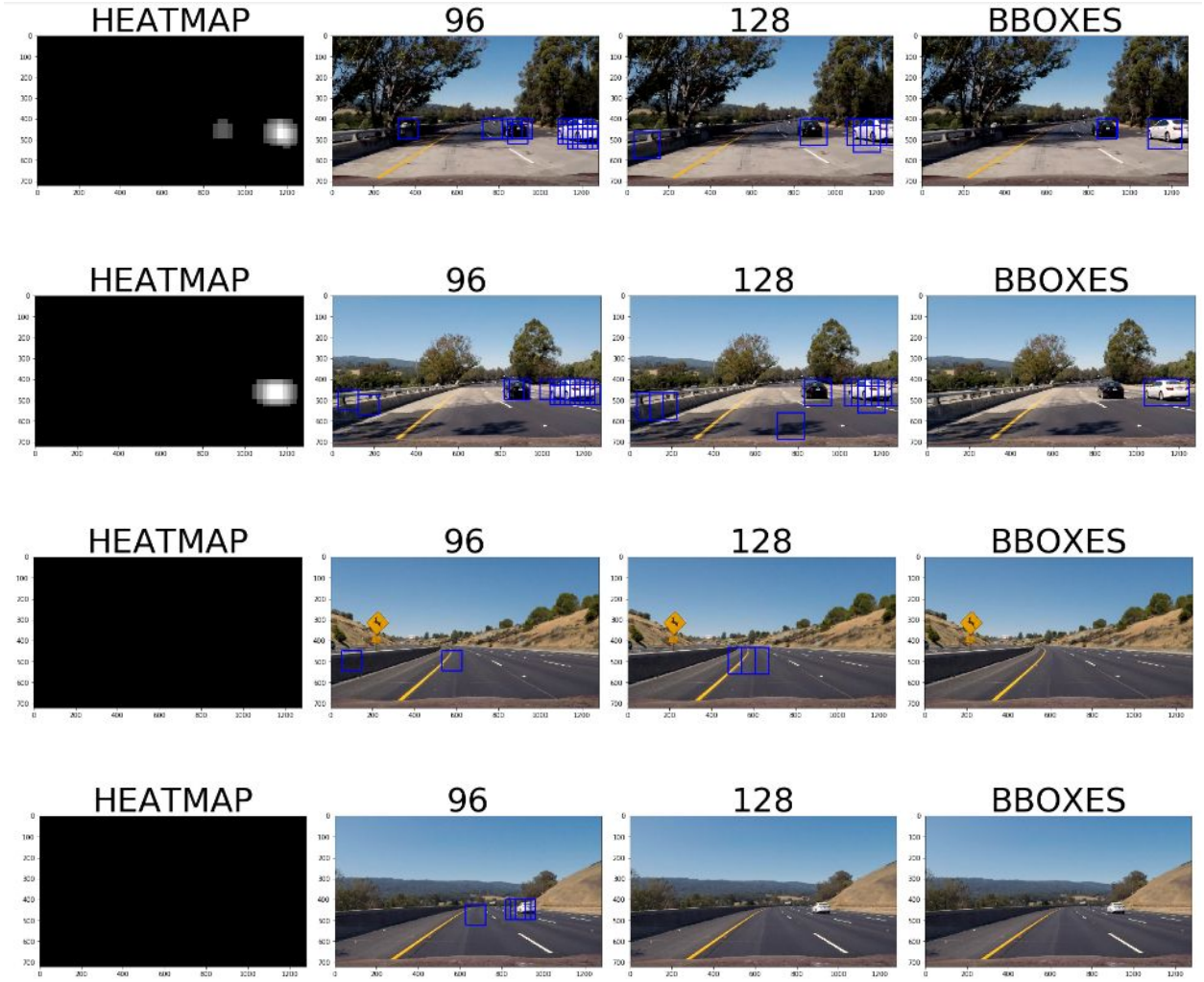
As can be seen in cell 7 lines 3-6 I began searching for windows from pixel 400 in y direction and with with window size 96 proceeded up to pixel 592 and with window size 128 proceeded up to 700 as I did not make sense to look for lower. In the x direction I used almost the whole width, with window size 96 I only limited it so that after resizing it would be divisible by 16, so that I would have the same amount of windows for HOG and other features.

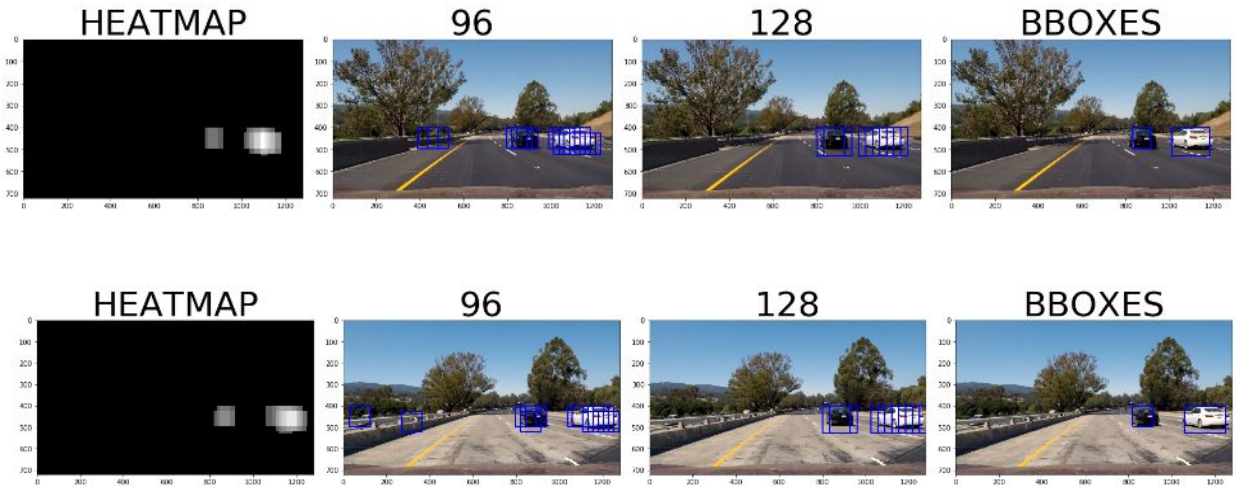
Since I wanted to speed up the process I decided to extract HOG features only once for the image. This is done in cell 5 function search_windows lines 101-139. For doing this I made sure that the subimage dimensions were divisible 16 so that I could skip two cells to get 75% overlap and I would get the same amount of windows as with slide_window which was still used to extract windows for spatial and histogram binning feature generation.

Windows resulting from slide_window function are fed into search_windows function (cell 5, lines 95-177) into which also the hog feature extraction is performed. Search_windows function extracts the features from all windows, runs the classifier on them and returns those windows where vehicles were detected.

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?

Finally I searched on two scales - (96, 96) and (128, 128) sized windows with the setup described above. Smaller windows resulted in a lot false positives and bigger windows rarely detected anything, so I did not use them to improve the speed of the algorithm. The pipeline without centroid tracking can be seen in cell 6. Below are some example images. Results from image “96” and “128” are combined together to make the heatmap and then bounding boxes are generated from the heatmap.





Finally, I also created two classes - Centroid in cell 7 lines 64-117 and TrackImage in cell 7 lines 120-198. Centroid is used to keep track of the labels which are appearing thanks to heatmaps and TrackImage is used to keep track of heatmaps over multiple frames.

The performance of the classifier was improved by averaging the heatmap over multiple frames and putting those bounding boxes on the heatmap which had high confidence during previous frames. This can be seen in cell 7 function track_vehicles_with_centroid_tracking lines 179-182 - first I add those to heatmap about which I was confident before with function add_confident_to_heatmap in line 180, then I add the current frame to heatmaps with function add_frame_to_heatmaps in line 181 and afterwards I use heatmaps from multiple previous frames with function combined_heatmaps in line 182.

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.)

Here's a link to video.

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 added the heatmaps over multiple frames and I kept track of the centroids of the labeled boxes with class Centroid in cell 7 lines 64-117. If the boxes appeared over

multiple frames their confidence increased (cell 7, line 189) and they were displayed (cell 7, line 194-197). If they were not present in the succeeding frames, their confidence decreased (cell 7, line 173) so that they disappeared or were deleted completely (cell 7, line 175).

The centroid confidence increased only if the new label center were close enough to previous ones (cell 7, lines 188). Also, centroid confidence decreased more slowly if a centroid had become confident enough (cell 7, line 102).

This approach helped to get rid of false positives.

For combining overlapping bounding boxes I used the function combine_overlapping_bboxes in cell 7 line 197 which uses labels functionality from scikit.ndimage.measurements label function to stitch multiple bounding boxes together. This approach helped me to combine overlapping bounding boxes.

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?

The classifier accuracy was 98.8%, but there were still a lot false positives which messed up the results quite a lot, meaning I had to build a lot of filtering into TrackImage and Centroid class than I had imagined at first. It would probably fail in different lighting conditions. The classifier accuracy should be better which could probably be easily improved with more data and a more powerful classifier. I wanted to focus on the sliding-window, heatmap and tracking approach as I had used a good classifier for a similar task in the past already and that would not have been such a challenge.

I could have used the false positives received during testing on video to add them to my training data for the algorithm to become more robust. This method is called hard negative mining if I am not mistaken.