Vehicle Detection & Tracking Project

Writeup

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.

Rubric Points

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

Writeup / README

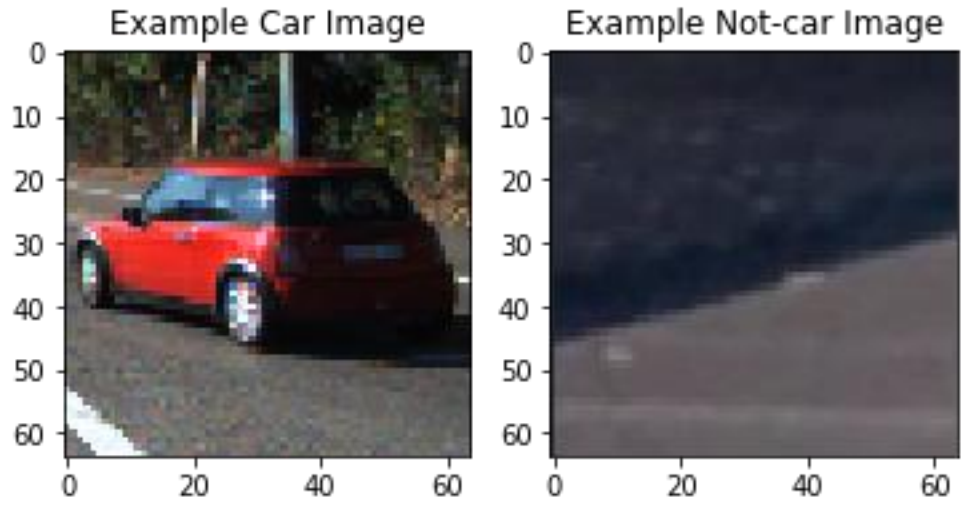
1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. Here is a template writeup for this project you can use as a guide and a starting point.

You're reading it!

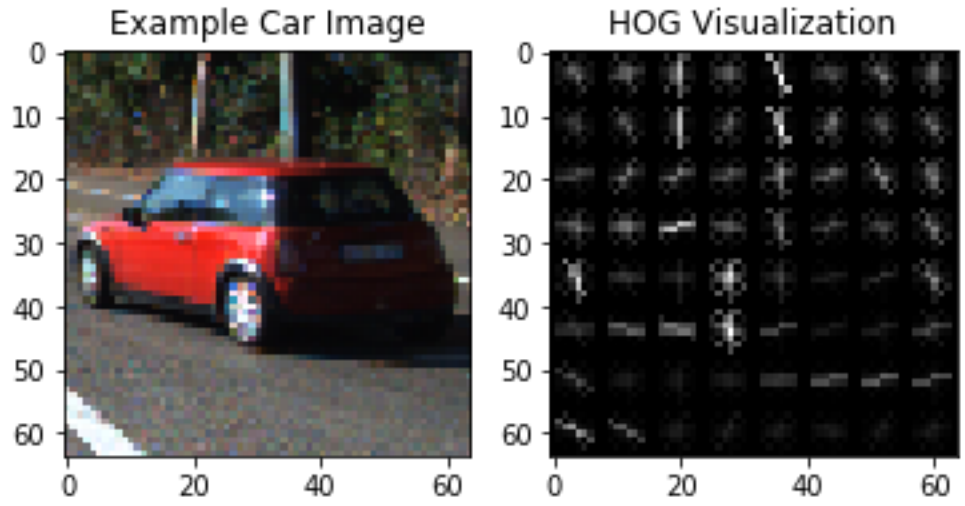
Histogram of Oriented Gradients (HOG)

1. Explain how you extracted HOG features from the training images.

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

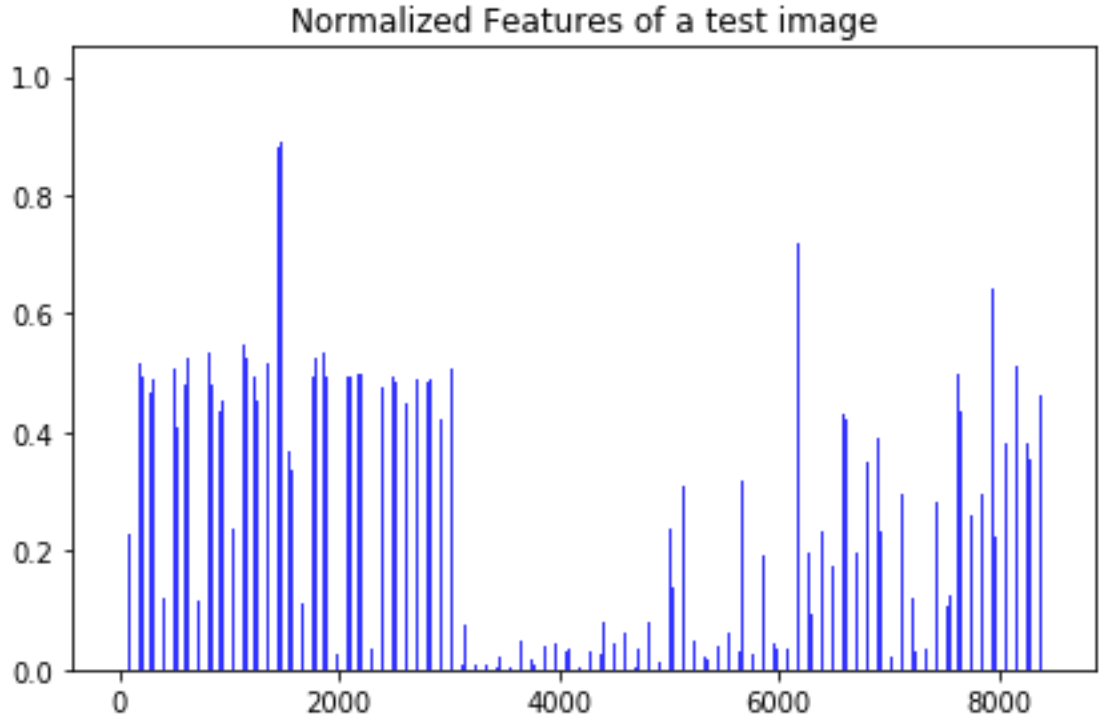


I then explored different color spaces and different `skimage.hog()` parameters (`orientations`, `pixels_per_cell`, and `cells_per_block`). I grabbed random image from car classe and displayed them to get a feel for what the `skimage.hog()` output looks like. Here is an example using the `Gray` color 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 out that for the project color space = 'YCrCb', orient = 9, pix_per_cell = 8, cell_per_block = 2 works fine. Here is the combined features set of a test image.



3. Describe how you trained a classifier using your selected HOG features (and color features if you used them).

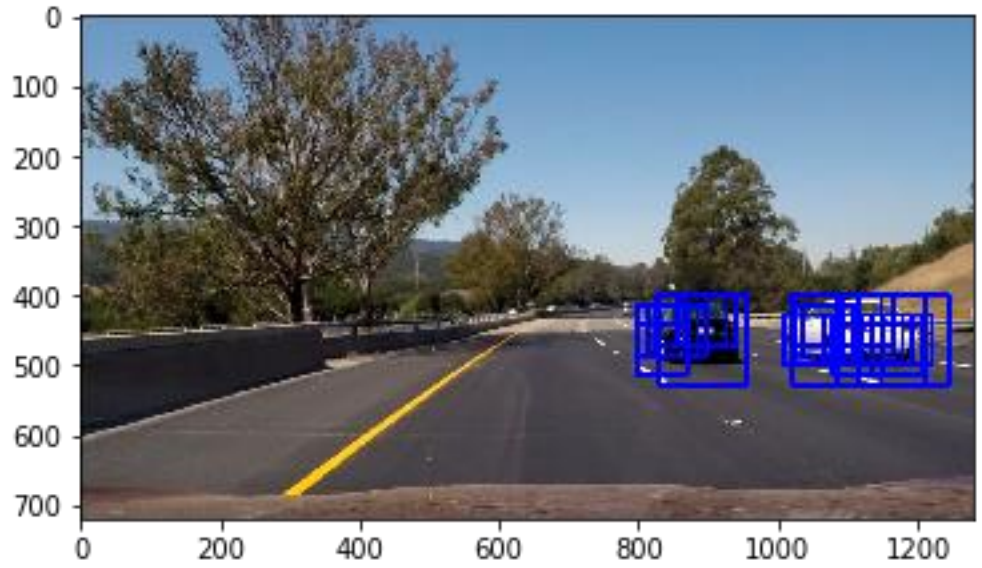
I trained a linear SVM using the combination of three features which are HOG, spatial bins and color histogram. The accuracy of the test set which is 20% of the entire images set is 98.82%

Sliding Window Search

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

I performed search operation on two kinds of window size which are (100,100), (60,60) using YCrCb 3-channel HOG features plus spatially binned color and histograms of color

in the feature vector and kept the ROI to (x:(700,max-x-axis),y:(400:700) for bigger window and x:(700,max-x-axis),y:(400:700) for smaller window), which provided good result. Here is one example:

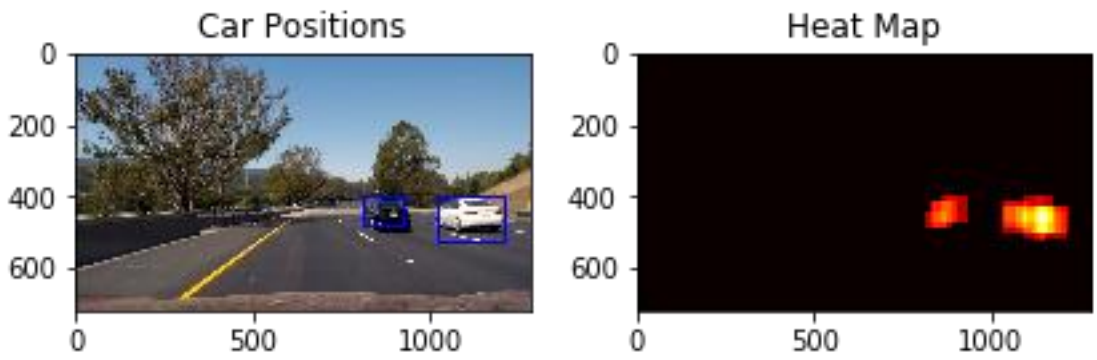


Video Implementation

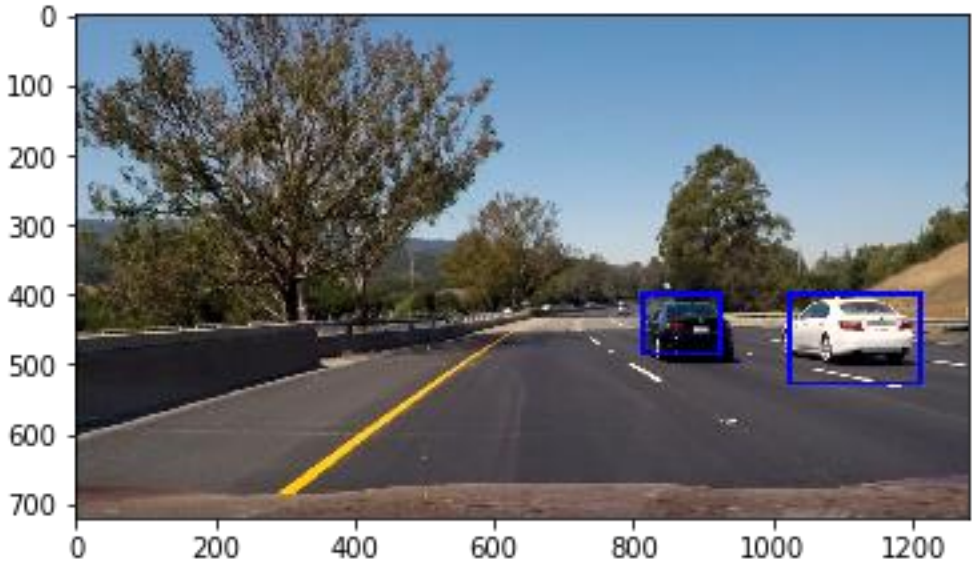
1. Describe how you implemented some kind of filter for false positives and some method for combining overlapping bounding boxes.

I recorded the positions of positive detections in each frame of the video. From the positive detections I created a heatmap and then thresholded that map to identify vehicle positions. For averaging, I have used a deque of maxlength=10. 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.
Here's an example result showing the heatmap from a series of frames of video, the result of `scipy.ndimage.measurements.label()` and the bounding boxes then overlaid on the last frame of video:

Here are a frame and its corresponding heatmap:



Here the resulting bounding boxes are drawn onto the frame:



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?

For this project, I used two window sizes to detect the position of the cars, and to keep the window size consistent I used averaging on the heatmaps with the help of a deque of maxlength 10. I have only taken windows in the fourth quadrant of the image, that is if there are cars in the left lanes, they won't be detected. The reason behind this is that for the entire video the cars are seen in right corner, thus performed search in fourth

quadrant of the image. So, for a different video, the search window will have to be changed accordingly.