Writeup Template

You can use this file as a template for your writeup if you want to submit it as a markdown file, but feel free to use some other method and submit a pdf if you prefer.

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.

Rubric Points

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

Writeup / README

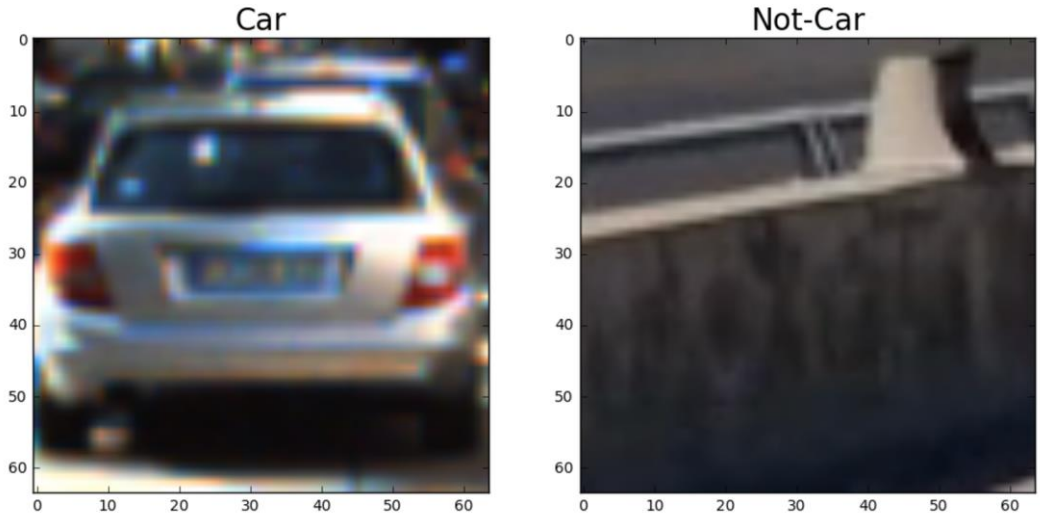
1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf. Here is a template writeup for this project you can use as a guide and a starting point.

You're reading it!

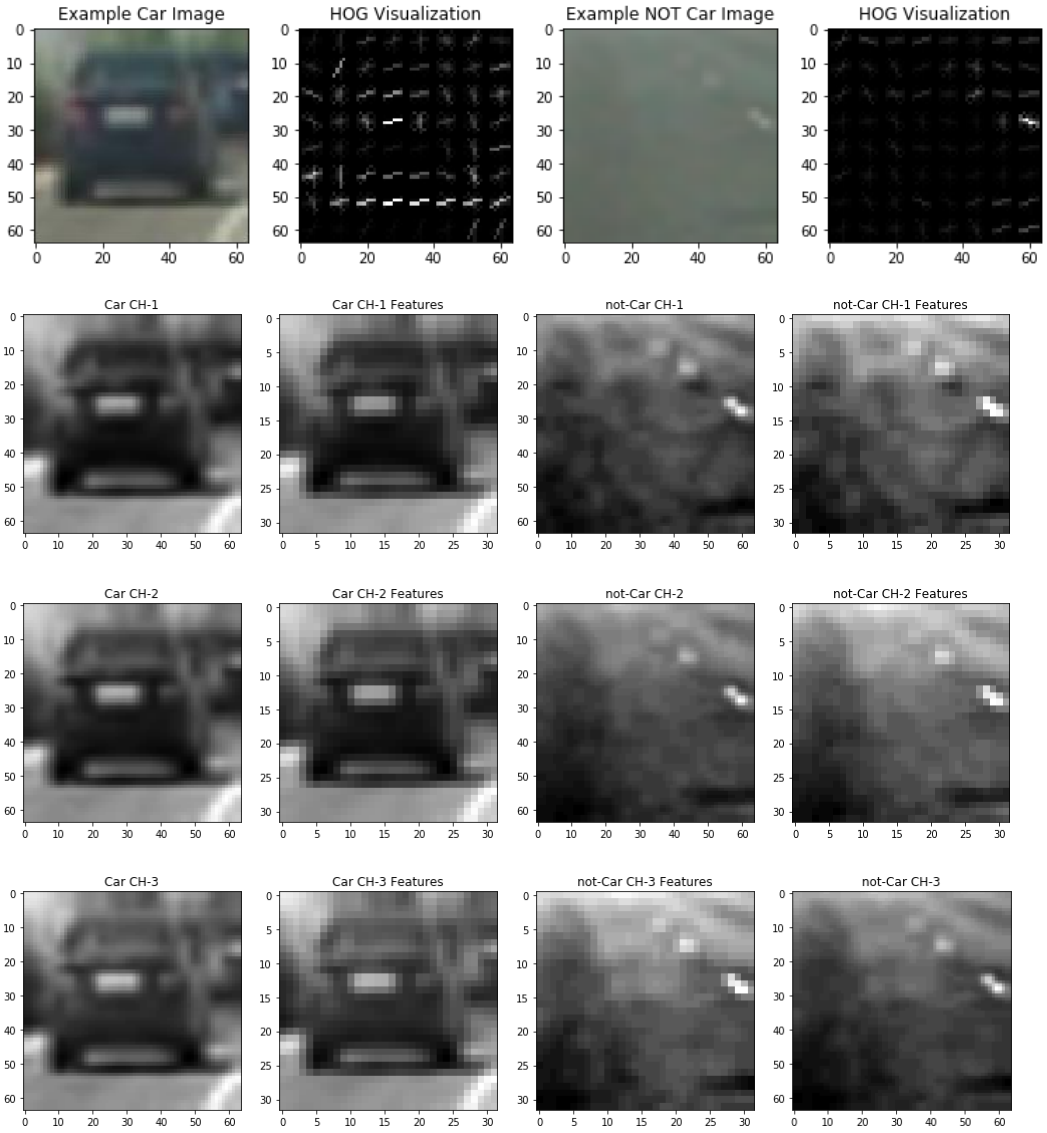
Histogram of Oriented Gradients (HOG)

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

The code for this step is contained in lines #84 through #204 of the file called P5.py.
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 RGB 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 used the RGB color space and HOG parameters of orientations=9, cells_per_block=(2, 2) as default of the lesson 20. And I decided pixels_per_cell=(8, 8) because it took longer time when I changed pixel_per_cell = (2, 2).

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 spatial binning features, color histogram features, and HOG features. The test accuracy after training is 0.9786.

Spatial binning features

I transformed car and not car images to 32x32 pixel.

Color histogram features

I extracted color histogram features from each of RGB color channel.

HOG features

I extracted HOG features from each of RGB color channel.

I used approximately 9,000 car images and almost similar number of not car images to train a linear SVM and test the accuracy of that training.

I separated these images into training ones and test ones.

I used 80% of the image data set for training.

And then I used rest of them to ensure the training accuracy.

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 used 2 size windows to search vehicles in the images.

The size I chose are (96, 96) and (160, 160). I think they detect vehicle well when I changed window size.

The window overlap is 50%.(Actually I didn't try to change it.)

Window search ranges are from 360 to 620 and from 460 to 720 respect to (96, 96) and (160, 160) of the window size at y-axis.

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 RGB 3-channel HOG features plus spatially binned color and histograms of color in the feature vector, which provided a nice result. Here are some example images:

Video Implementation

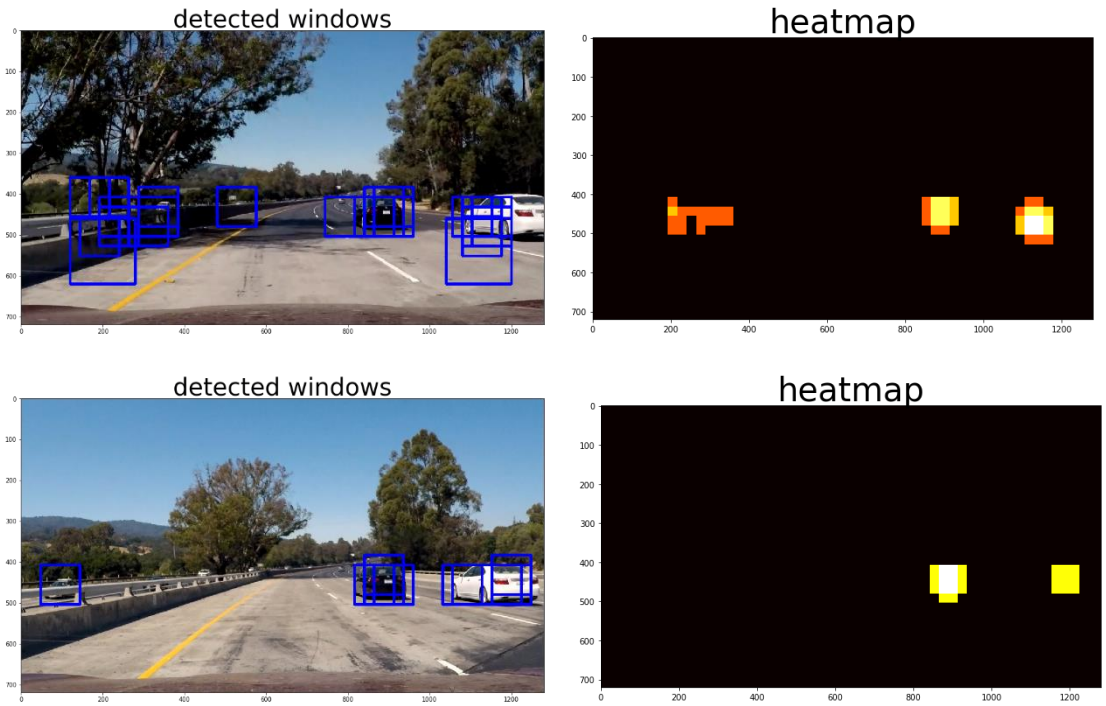
1. Provide a link to your final video output. Your pipeline should perform reasonably well on the entire project video (somewhat wobbly or unstable bounding boxes are ok as long as you are identifying the vehicles most of the time with minimal false positives.)

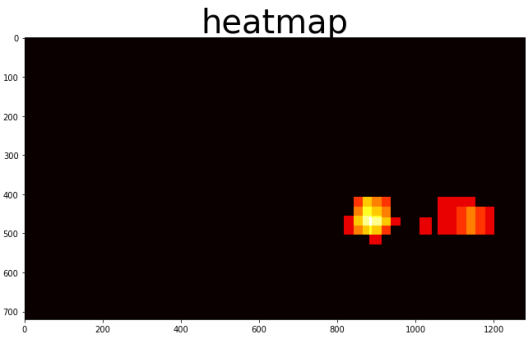
Here's a link to my video result

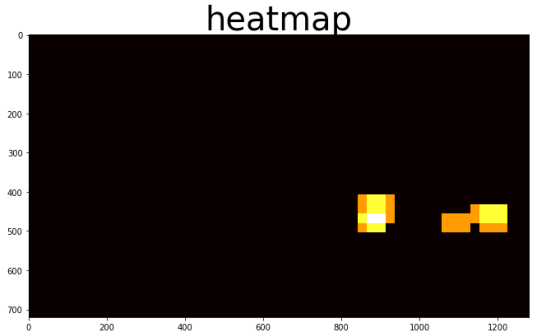
2. Describe how (and identify where in your code) you implemented some kind of filter for false positives and some method for combining overlapping bounding boxes.

I 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. 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 some test images, the result of scipy.ndimage.measurements.label() from these images:

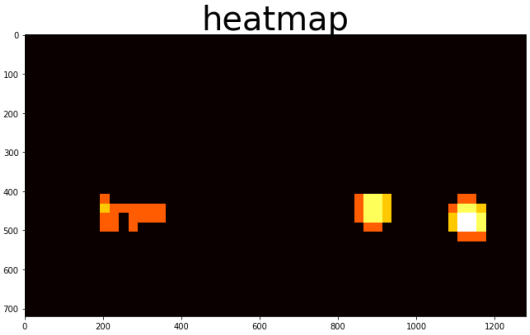
Here are six frames and their corresponding heatmaps:

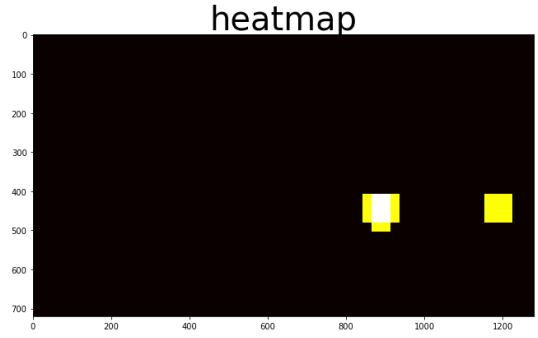


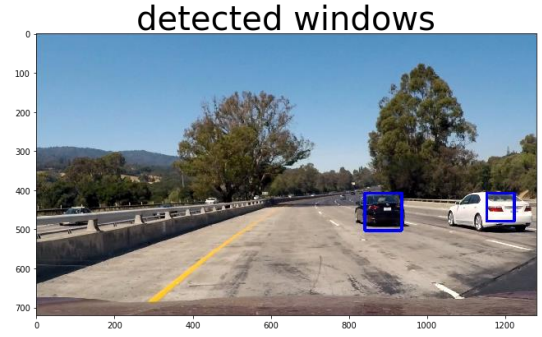


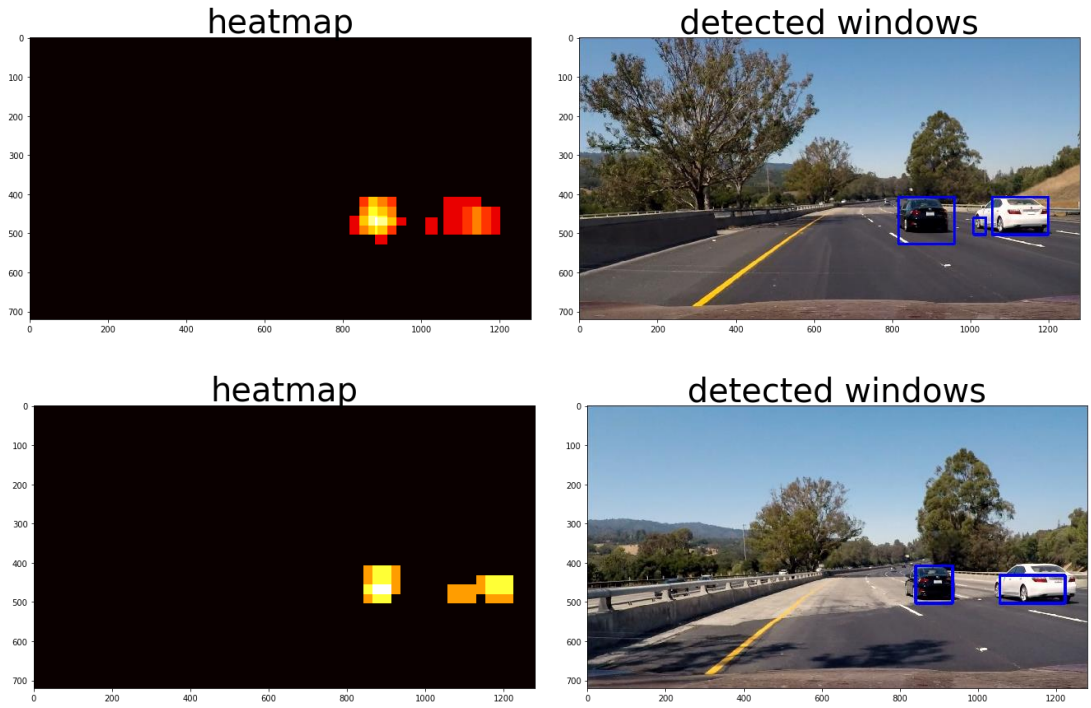


Here the resulting bounding boxes are drawn onto the above four heatmap with threshold filter treatment:









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?

Here I'll talk about the approach I took, what techniques I used, what worked and why, where the pipeline might fail and how I might improve it if I were going to pursue this project further.
I use filter treatment to avoid detecting false positive. However if the car comes in front of my vehicle suddenly from out of camera area, my program has time lag to detect it. To improve this I think I should save video frames to use filters.