Vehicle Detection and Tracking

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.

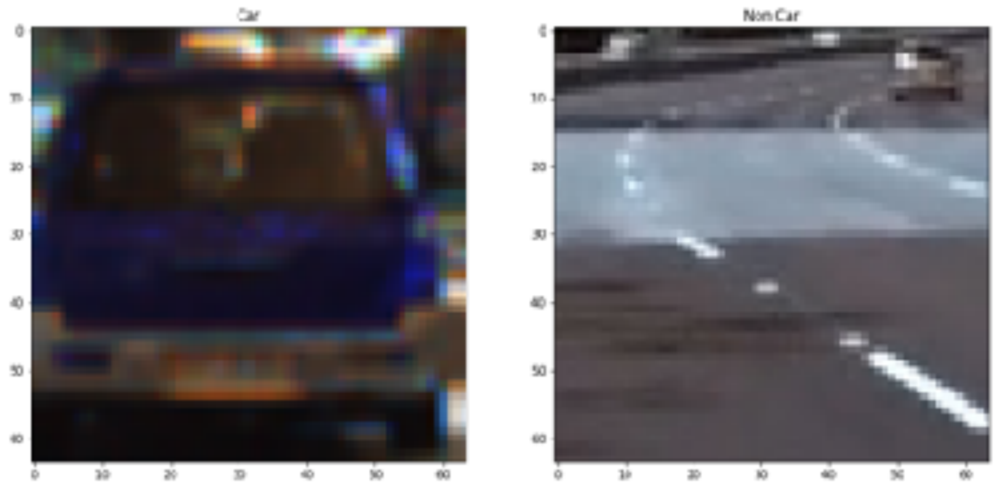
Files for this project:

- saved_images
- T1P5.ipynb
- T1P5_writeup.pdf
- project_video_output.mp4

Histogram of Oriented Gradients (HOG)

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

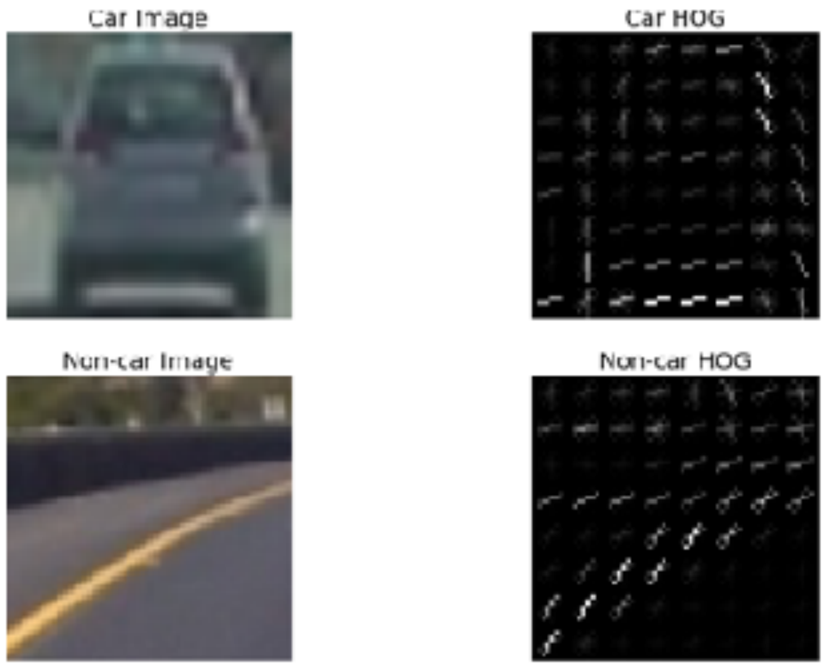
The code for this step is contained in the fifth code cell of the notebook.



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

There were a total of 8793 cars and 8968 non-cars.

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 of a car and a non-car image using the get_hog_features() function.



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

Through extensive testing I was able to settle on the following parameters:

color_space = “LUV”
I experimented with the other colors and found this to work best with my SVC accuracy.

orient = 11
I found this to provide enough increase in accuracy beneficial enough to sacrifice a little more time in training time but going higher than this was to much of a negative impact.

pix_per_cell = 8
A higher pixel per cell speeds up calculations but loses accuracy and a pix_per_cell of 8 makes for a good of both.

cell_per_block = 2
I found that a higher number will make a longer feature vector and gave more false positives.

hog_channel = “ALL”
Including the hog features of all channels gives a better result compared to using a single channel.

spatial_size = (16,16)
The higher the number the higher the accuracy but the longer it takes to train. 16 seemed to give me the best results.

hist_bins = 16
Again with testing I found this to be ideal.

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. I combined the get_hog_features(), bin_spatial(), and color_hist() in the extract_features function. I split 20% of the training data as test data and after extensive testing as noted above I obtained the following results:

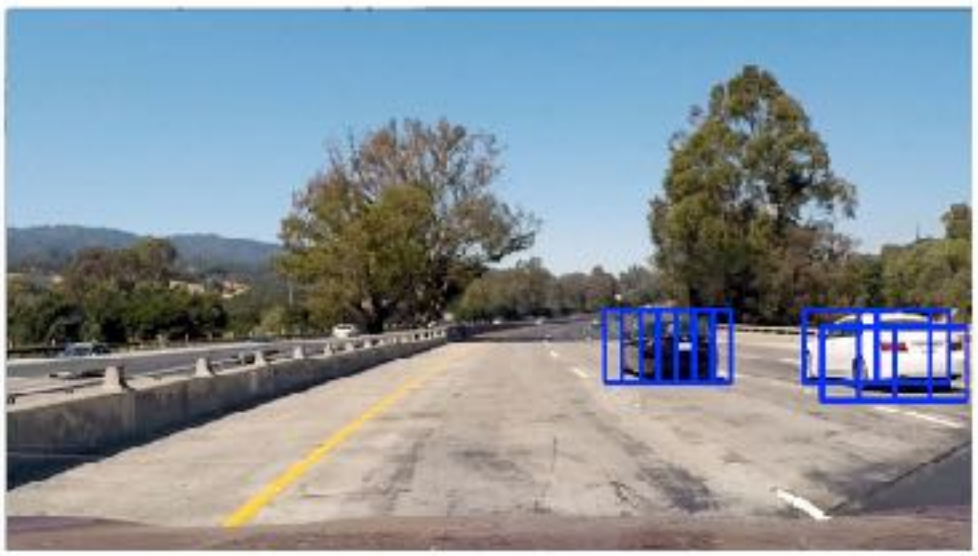
Using: 11 orientations 8 pixels per cell and 2 cells per block
Feature vector length: 7284

```
Time to train SVC: 34.88 sec
Accuracy of SVC: 1.0000
```

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 initially tried HOG sub-sampling method but changed to the method from the Udacity lesson with a small change to search for multiple scale sizes and introducing a threshold to help filter. I found this combination to be very helpful.



Video Implementation

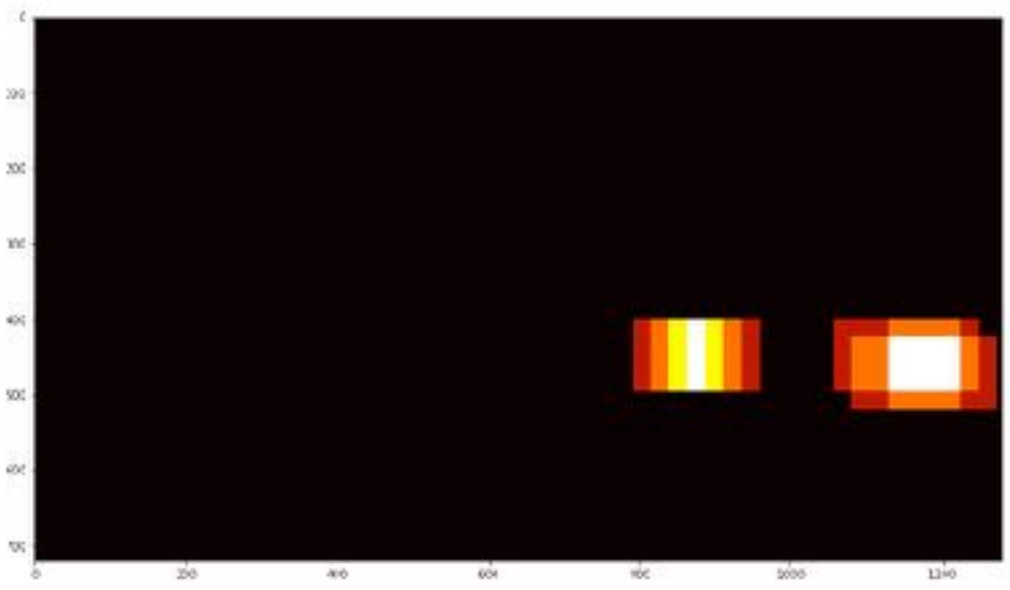
1. **Provide a link to your final video output. Your pipeline should perform reasonably well on the entire project video (somewhat wobbly or unstable bounding boxes are ok as long as you are identifying the vehicles most of the time with minimal false positives.)**

The video is located in the project file and is labeled project_output_video.mp4.

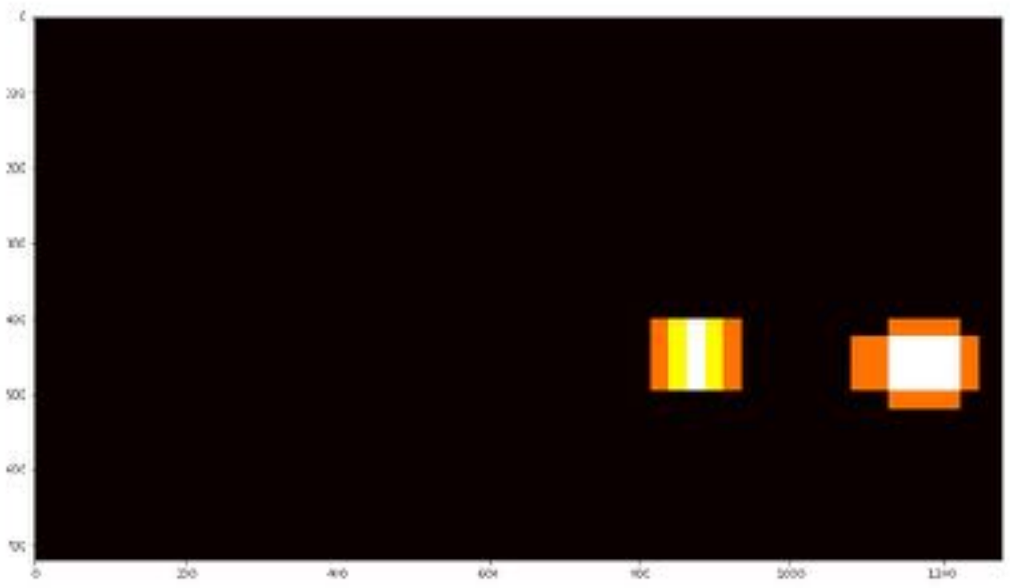
2. Describe how (and identify where in your code) you implemented some kind of filter for false positives and some method for combining overlapping bounding boxes.

The code for this begins in cell sixteen and ends with cell twenty-one. This will use the the find_cars() function to create boxes using the most recent frames and boxes. I created a heatmap and then thresholded that map to identify vehicle positions.

Here's an example image from the add_heat() function.

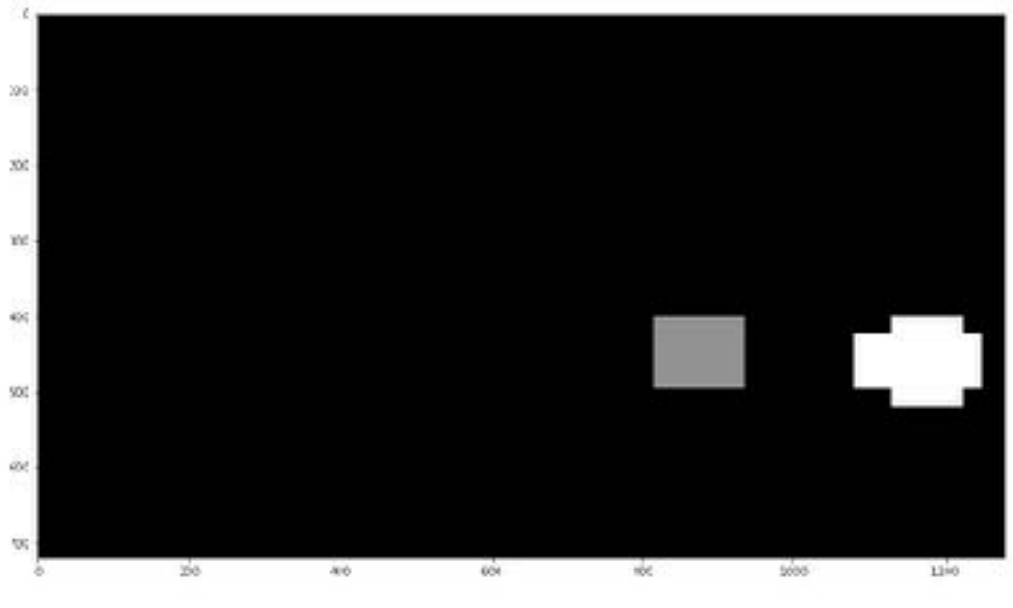


Here's an example image from the apply_threshold() function where you definitely notice the change in the colors.

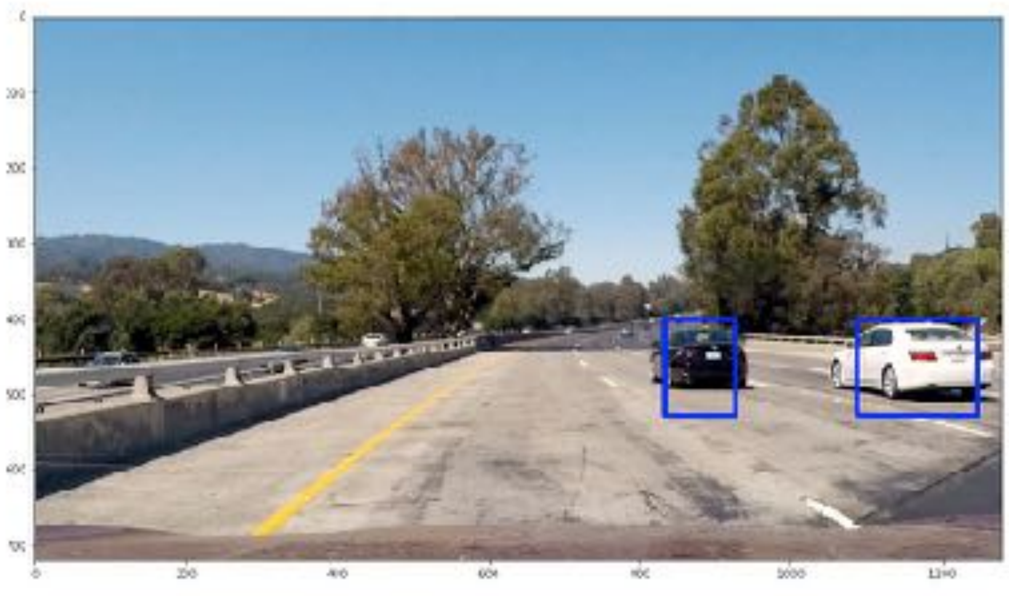


Once you have a thresholded heat-map, there are many ways you could go about trying to figure out how many cars you have in each frame and which pixels belong to which cars, but one of the most straightforward solutions is to use the `label()` function from `scipy.ndimage.measurements`.

Here is an image after we apply the label() function.



And now here is an example image from the draw_labeled_boxes() which draws the boxes on an actual image detecting the cars location.



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?

- During my learning experience with this project I’ve learned that using a network such as YOLO, SSD, or even FCN would make for more robust system.
- The processing time for this project took a long time on my Macbook but fortunately for the provided test video I was able to feel confident in running the full length video without problems.
- It took me a lot of experimenting with different settings to get a good result in my training and seemed to take me the most time.
- I question using a camera for using vehicle detection due to confusion with lack of visibility at night or in severe weather such as identifying a white car in the snow.