Vehicle Detection and Tracking

In this project, the goal is to write a software pipeline to identify vehicles in a video from a front-facing camera on a car.

Vehicle Detection and Tracking Project

The steps which were taken to accomplish this project are the following:

1. Perform Histogram of Oriented Gradients (HOG) feature extraction, binned color feature extraction and color histogram feature extraction on a labeled training set of images
2. Setup and train a Linear SVM classifier
3. Implement sliding-window technique with multi-scale windows and use the trained classifier to search for vehicles in images.
4. Implement hog sub-sampling window search technique to speed up the vehicle detection process; Apply a heat-map from vehicle detections to combine overlapping detections and remove false positives.
5. Modify the parameters and run the pipeline on the project_video stream to 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're reading it!

Histogram of Oriented Gradients (HOG)

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

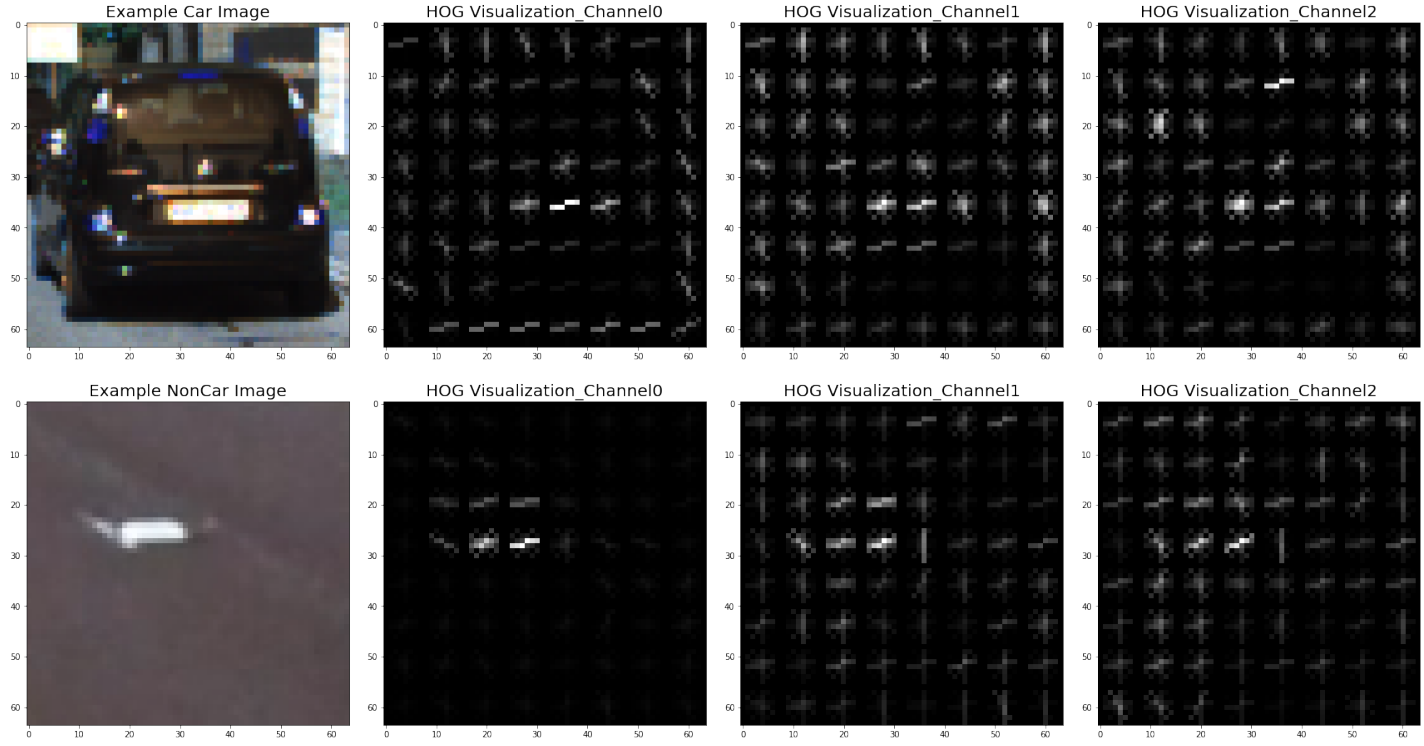
In this project, binned color,color histogram and HOG features were extracted. The code for this step is contained in the second code cell of the `vehicle_detection_tracking.ipynb`.
All of the features were extracted by a function `extract_features` in code cell 3 and 4.

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. I grabbed one image from each of the two classes and displayed them to get a feel for what the feature output looks like.

Here is an example using the following HOG parameters:

```
color_space = 'YCrCb'
orient = 9
pix_per_cell = 8
cell_per_block = 2
hog_channel = 'ALL'
```



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

The values of orient, pix*per*cell and cell*per*block were set same as the ones in lesson. I tried various combinations of parameters (color space and hog*channel) and test vehicle detection of images in test*images folder using same svm classifier. Most combinations can provide good classification results

during training.
The above combination provide best detection results with test images. So I chose these parameters.

The hog_channel set to ALL improved the accuracy but slow down the classification process.

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 linear SVC. Hog, binned color and color histogram features were extracted from car and noncar images. The test data were split from labeled train data (20%).

After training, 0.9916 accuracy was achieved.

The code can be found in part 2 Classifier Trainning.

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?

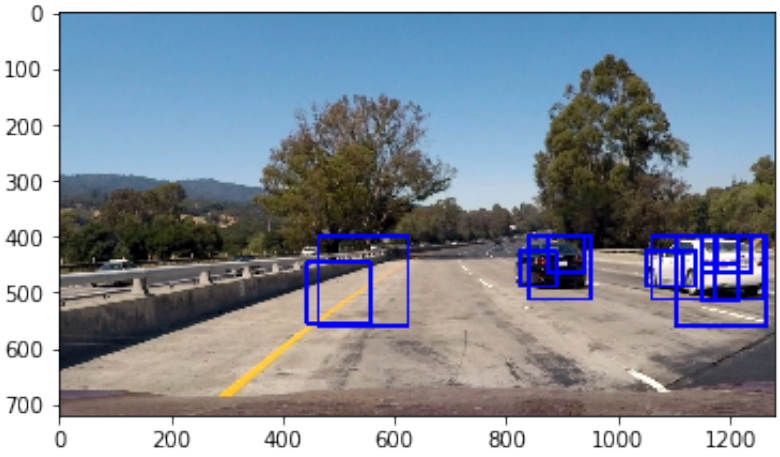
I search several sizes of windows in selected areas of images. The size of windows were set according to the approx size of vehicles in test images.
The size of windows were set to (64,112,160). The overlap is set to 0.6.

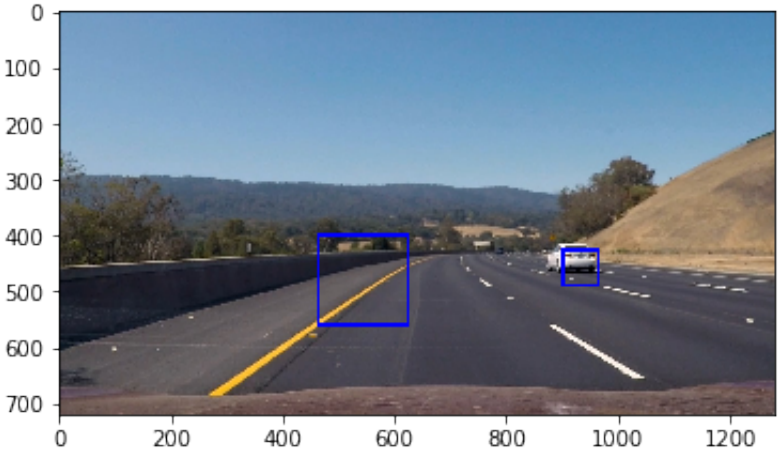
The value of overlap came from several test on test_images.

The intrest area was selected as following:
x*start*stop=[400, 1280]
y*start*stop=[400, 600]
This area is where the cars could appear.

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?

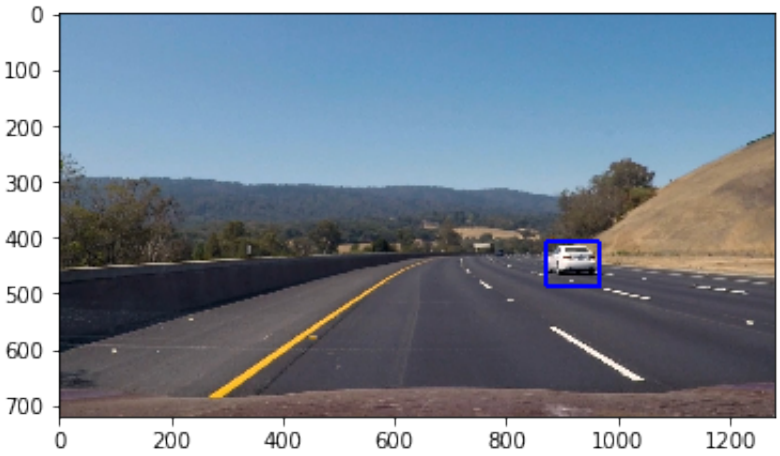
Ultimately I searched on 3 scales (64,112,16) using YCrCb 3-channel HOG features plus spatially binned color and histograms of color in the feature vector. Here are some example images:





There are some false detections which can be handled by the filter and heatmap technichs which will be mentioned in video implementation part.

The final outputs are:



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 my result video

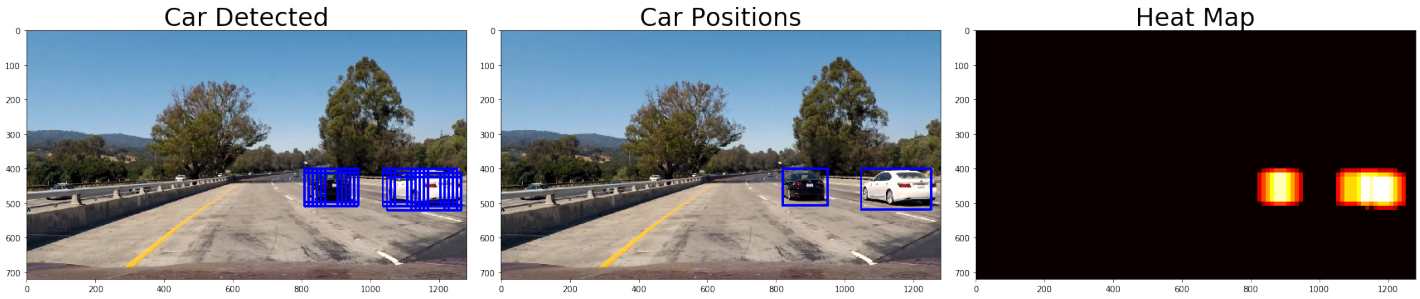
2. Describe how (and identify where in your code) you implemented some kind of filter for false positives and some method for combining overlapping bounding boxes.

The code can be found in code cell 18 where the function "vehicle_detection" was defined. The helper functions can be found from code cell 14-16.

The steps inculde:

- Record the positions of positive detections in each frame of the video.
- Create a heatmap and then thresholded (the threshold approach same as the suggested one in the lessons) that map to identify vehicle positions
- Use `scipy.ndimage.measurements.label()` to identify individual blobs in the heatmap. I then assumed each blob corresponded to a vehicle.
- Constructed bounding boxes to cover the area of each blob detected.

Here's one example result showing the heatmap from an image in test_images folder.



In the video implementation part, I used 'find_cars' functions four times. This help me to use different window scale and overlapping parameter settings to detect vehicles.
The parameter sets are:

```
ystart_1=400
ystop_1=500
xstart_1=600
xstop_1=1000
scale_1=1
cells_per_step_1=1

ystart_2=500
ystop_2=650
xstart_2=600
xstop_2=800
scale_2=2
cells_per_step_2=3

ystart_3=400
ystop_3=650
xstart_3=400
xstop_3=600
scale_3=2
cells_per_step_3=3

ystart_4=400
ystop_4=650
xstart_4=800
xstop_4=1280
scale_4=1.5
cells_per_step_4=1
```

To smooth the output in video, I implement a simple low-pass filter on the new and the previous cars boxes coordinates and sizes of the previous data. The filter coefficient is set to 0.4.

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 pipeline is able to correctly lable cars in the project video frames. The steps and methods I used are:

1. Perform Histogram of Oriented Gradients (HOG) feature extraction, binned color feature extraction and color histogram feature extraction on a labeled training set of images
2. Setup and train a Linear SVM classifier
3. Implement sliding-window technique with multi-scale windows and use the trained classifier to search for vehicles in images.
4. Implement hog sub-sampling window search technique to speed up the vehicle detection process;

Apply a heat-map from vehicle detections to combine overlapping detections and remove false positives.

5. Use a low-pass filter to smooth the video output.

There are some drawbacks in my pipeline. It may not process very fast in real vehicles.

It also may not work properly in conditions like:

- Too many cars situation like urban traffic
- Bad light or weather situation (rain, night)

More datas and detail parameter tuning may help the pipeline to work more robust and efficient.