Vehicle Detection and Tracking

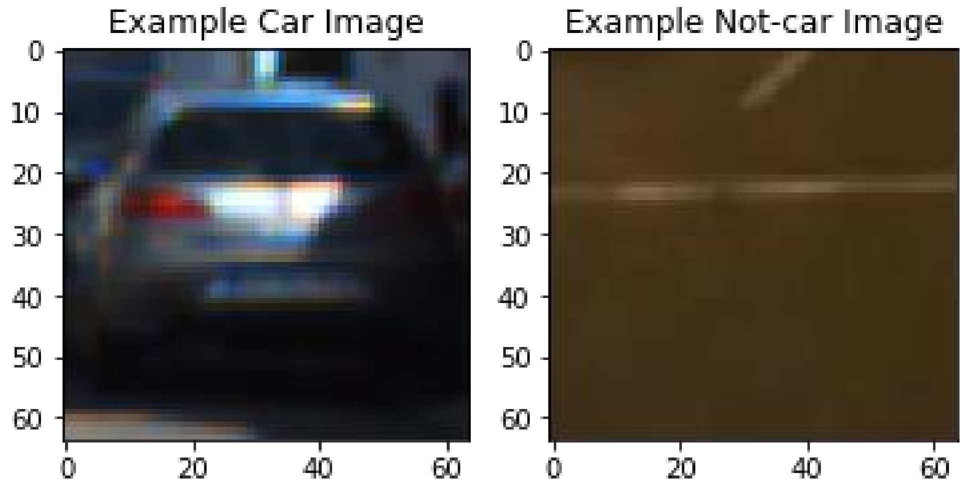
Udacity self-driving car Nanodegree project 5

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.
- 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.

1.Histogram of Orient Gradient(HOG)

I started by reading each vehicle and non-vehicle images,Here are two example images.



In order to build a classifier to classify vehicle and non-vehicle images, I first need to extract useful color and shape features of an image. According to the instruction of the lesson videos,I performed histogram of orient gradient to each images.HOG features can reflect the shape the object and are more robust to variation in shape.I also added spatial features and color histogram features to the feature vectors because they give me better result. After several experiments, I ended up to the parameters as below

Color space：YCrCb=9

pix_per_cell=8

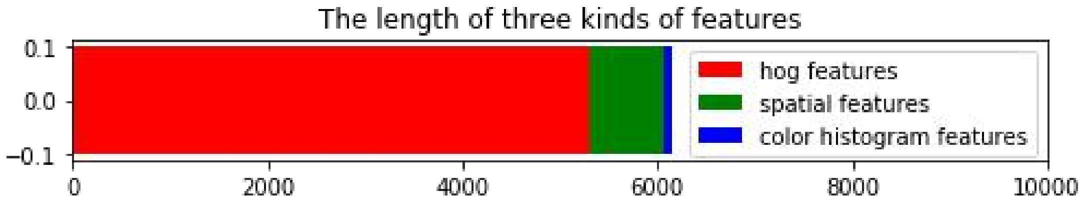
cell_per_block=2

hog_channel='ALL'

spatial_size=(16, 16)

hist_bins=32

hist_range=(0, 256)



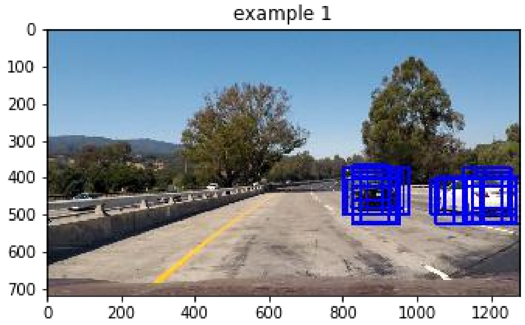
The length of the final feature vector is 6156.

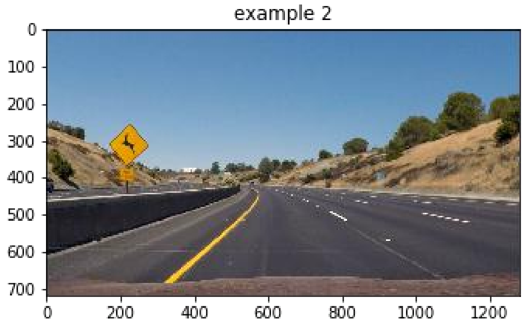
2.Build a Classifier

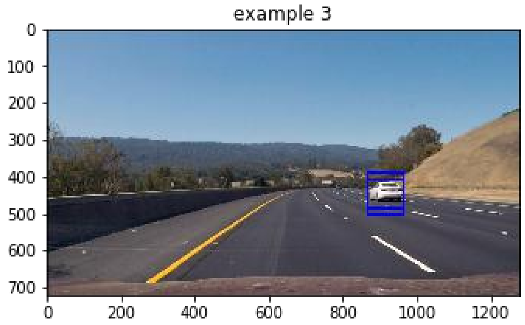
After reading all the training images and get a feature array of(17760, 6156), I built a Linear support vector machine to classify them.I first shuffle the training data randomly to avoid the influence of order and split 20% samples for testing the performance of my model. Every features had been scaled to zero-mean and unit variance distribution before feeding into the model. Because the number of features is very high relative to the number of samples. I set the C parameter of SVM to a very low value 0.001 to avoid overfitting. After all this done, I got a very nice test accuracy at 0.9916.

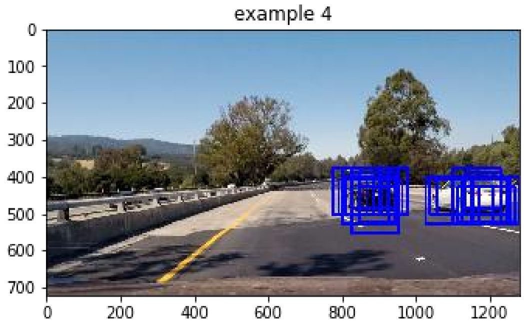
3.Sliding windows

I applied a slide windows approach to each test frames in which the classifier build above decided whether there is a car in each window images. This step is very time consuming so I only focus on the interest regions where a cars my appear. The size of a car maybe different in each frame so I have to use different size of windows to search the image for several times.After some experiments, I choose only two window size (96,96) and (128,128), applying smaller window size such as (64,64) do not seem help much and may bring too much false positive. Here are some example output images of my approach.

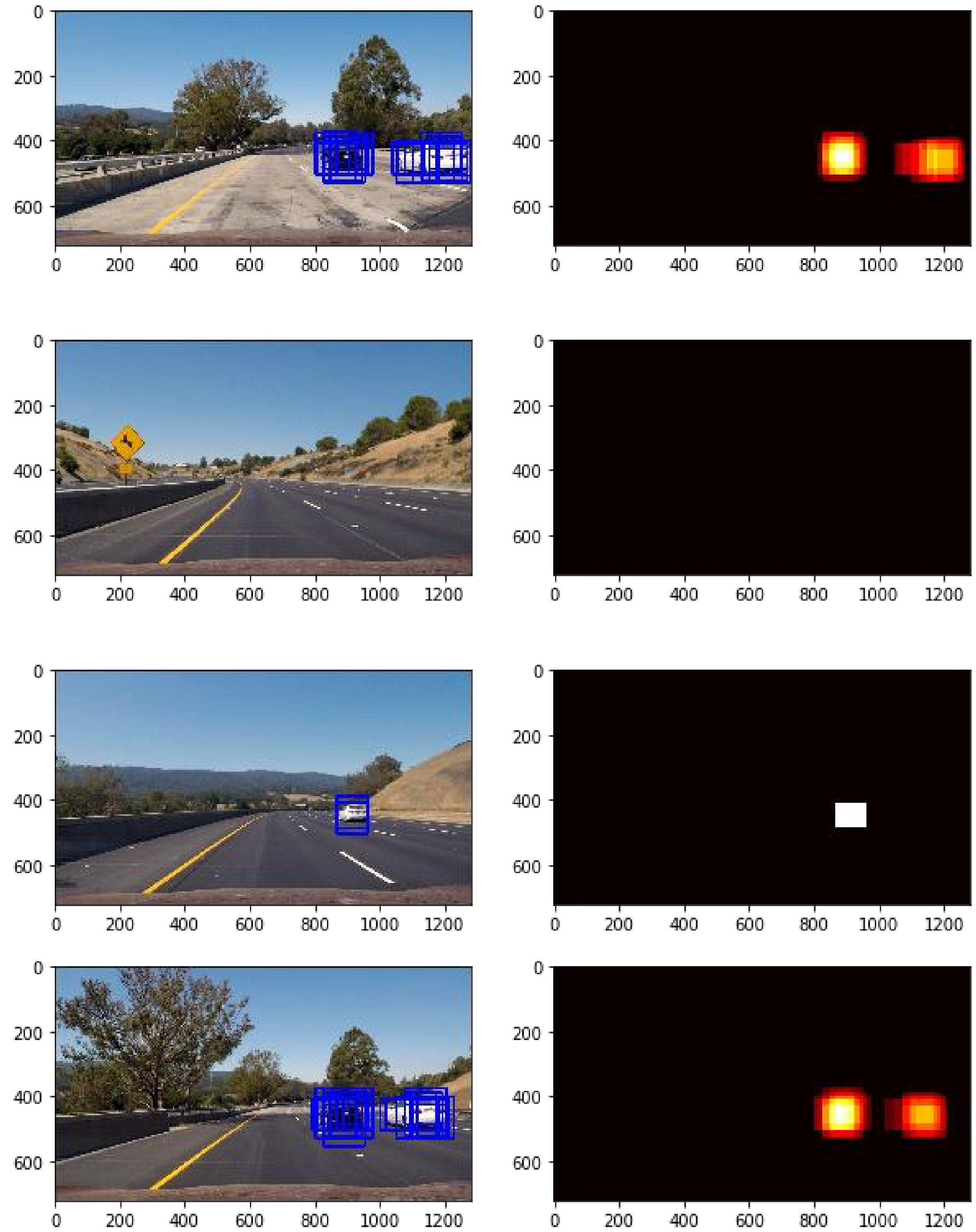

example 1

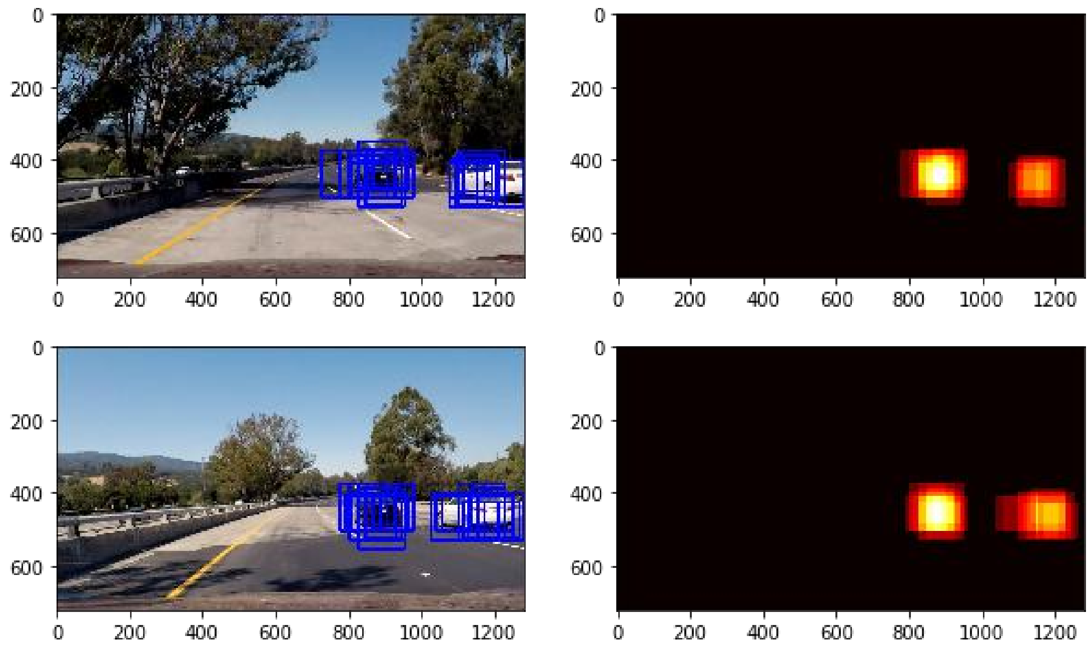

example 2


example 3


example 4

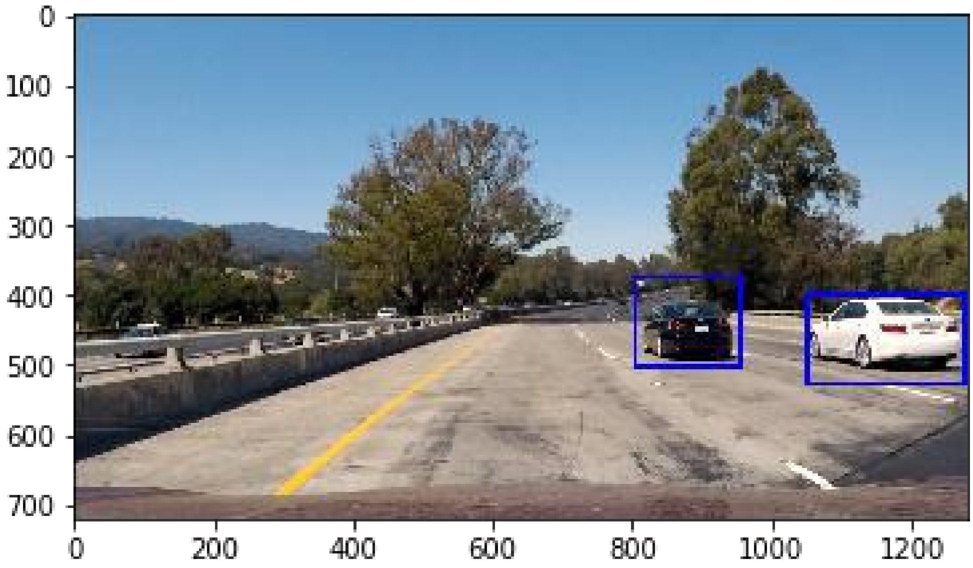
As shown above ,the slide window approach and my classifier can accurately detect the position of vehicles but will tick a single vehicle for many times. Therefore, I draw the heat map of each processed image and only remain the region that has been boxed at least two times.This approach can more robustly detect the region of a vehicle and filter most false positives.Here are the heat map outputs of test images provided





4.Video processing

Once I got the heat map of an image,I can identity the exact bounding boxes region and draw rectangle boxes along the detected vehicles.In this project I processed each frame independently. Here is an example output. Here is the link to my video result



5.Discussion

In this project, I applied an explainable feature extraction method and build a high accuracy classifier to classify vehicle and non-vehicle images.This pipeline is robust with nice parameter tuning.However,I think this slide window approach is too time-consuming for real application.It cost more than 3 seconds per frame in my machine.I think we can accelerate this pipeline by consider the position of a vehicle in last frames.Deep learning approach can also be considered in which the image maybe treated as a whole.