Vehicle Detection Project

The steps of this project are the following:

- Normalize features and randomize a selection for training and testing dataset.
- Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images and train a classifier SVM classifier
- 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.

Data Exploration

I started by exploring the data set. Images were provided in png format and other details of the dataset used are given below

```
count of vehicle images :  8792
count of non vehicle images :  8968
Image Size :  (64, 64)
Image Channel :  3
```

From the count of images in both the class i.e. vehicle and non-vehicle seems nearly balanced both are having around 8k images

Size of the images provided is 64x64 with 3 color channels.

Data Split

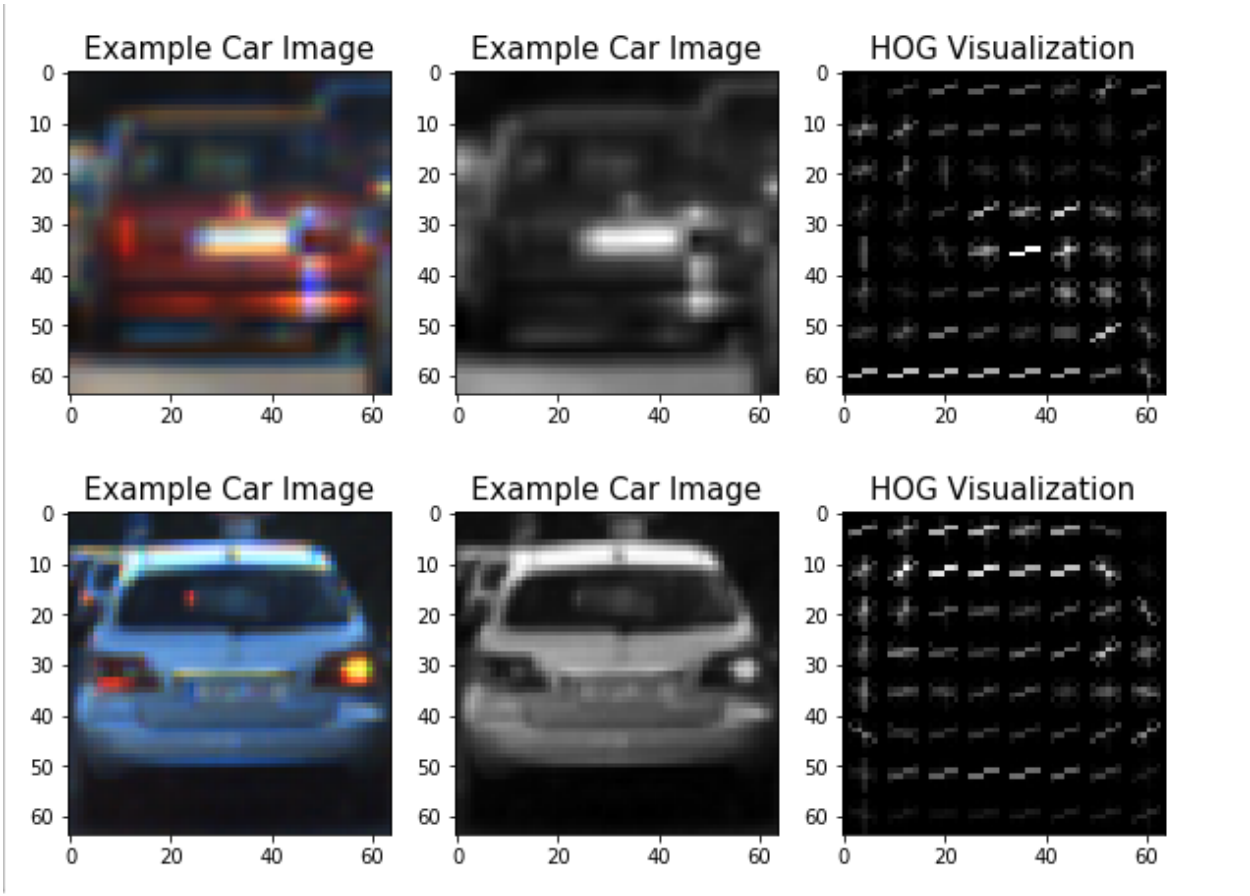
Normalization was required to be performed on the training and testing dataset. For which I have used StandardScaler function of sklearn library, code for the same is given in the cell Data Split.

For splitting the data, I am using train_test_split from sklearn library, my training and test dataset split is training data: 80% and test data is 20%.

Histogram of Oriented Gradients (HOG)

Code for this section is under HOG cell of Jupyter notebook

I have made a function to extract HOG features from the supplied image i.e. extract_HOG_features(img,vis=False,feature_vector=True). To test it I have used 2 images from vehicle dataset and applied HOG on it , results of which are shown below:



Total size of the hog features is 4400, I am converting RGB image to YUV color space and using Y channel for extracting HOG features.

I tried various combinations of parameters, but parameters which provided the best result in my case are mentioned below:

pix_per_cell = 8, cell_per_block = 4, orient = 11

Training SVM Classifier

I have used SVM classifier with rbf kernel which is a non-linear kernel, Earlier I tried using linear kernel but I got best result with rbf kernel with the value of C as 86.

I have also saved the model after training to utilize it later.

Code for the same is available under classifier training cell in jupyter notebook.

Trained classifier is giving 97% accuracy on test dataset

Sliding Window Search

After successful training of classifier, the next step is to search for vehicle in video frame and pass that portion of the image to classifier. As vehicle might appear in many different sizes based on the distance from our car, I have used 5 different scale of window search. as the image from our dash camera is a perspective view, vehicles which are near to our car will appear bigger and vehicle which are far from our car will appear small.

I have use 5 different scale with 5 different value of y_start to search vehicle in the frame, which help me to optimize the performance of the pipeline as we are only searching in the specific region of the frame

Code for the same is available in process_img(img) function.

Heat Map

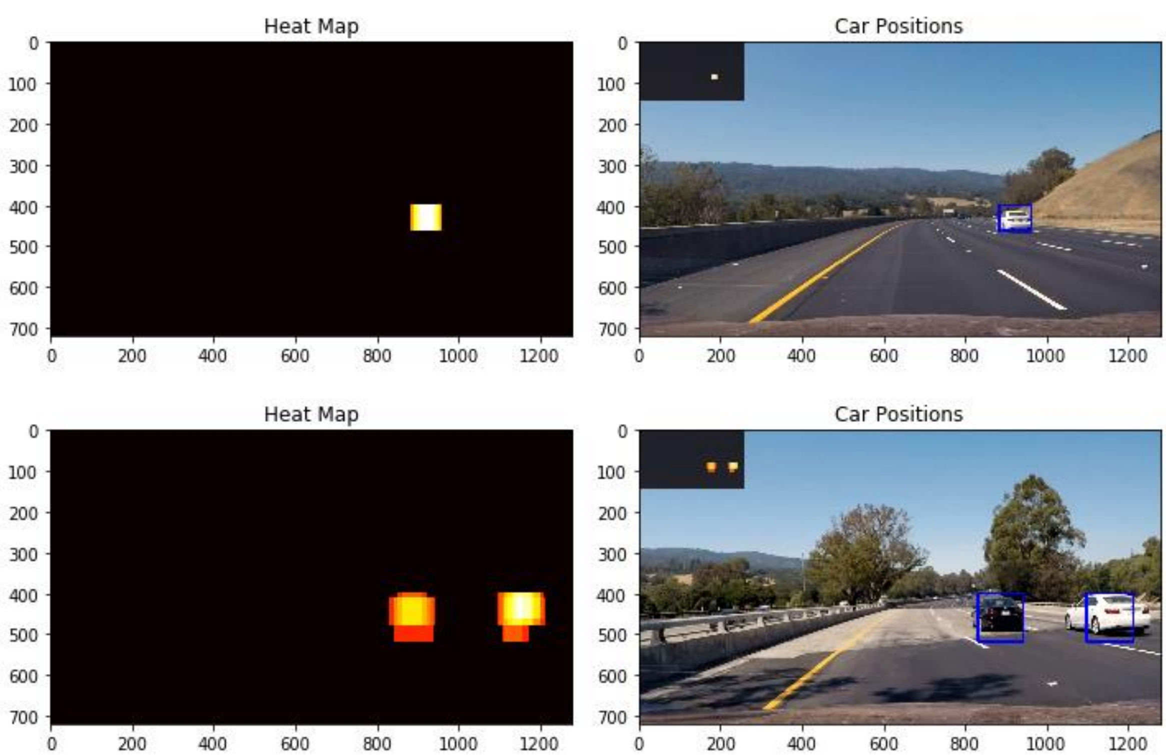
Result after the sliding window search has too many windows detected for one vehicle to detect only one window around the vehicle, I have utilized heatmap.

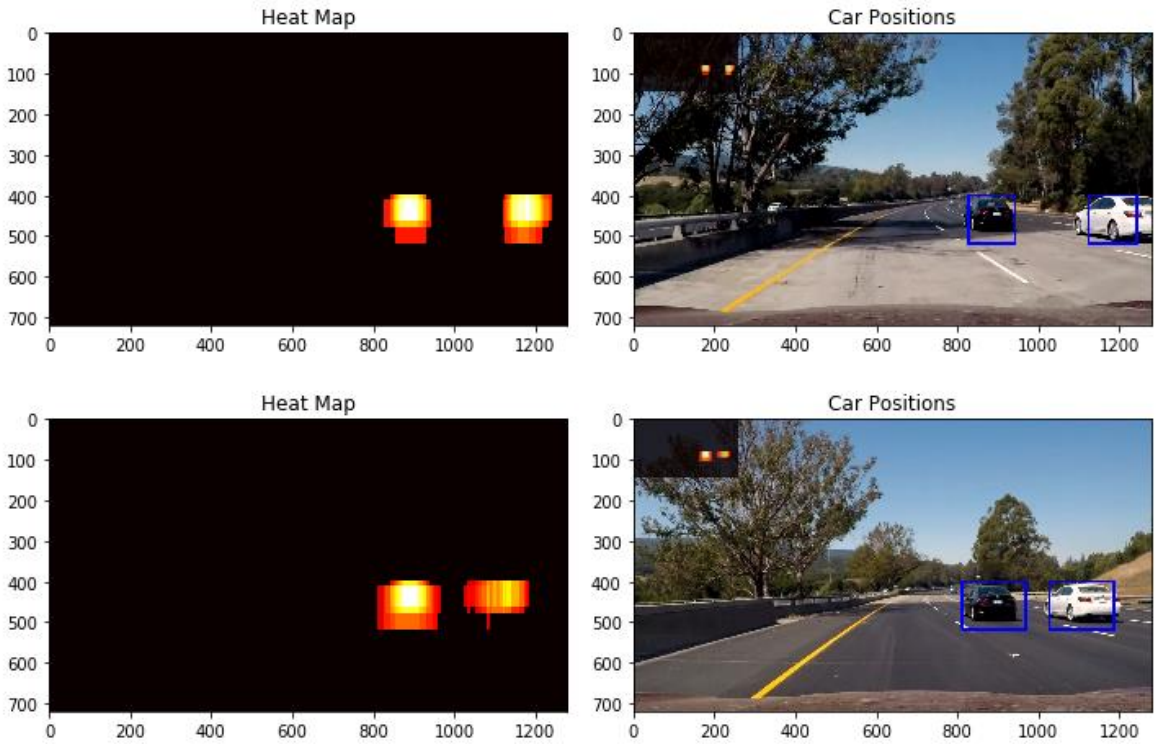
I am adding the heat using add heat function which adds the heat inside all the windows detected finally I am using apply_threshold method to threshold heat value and coming up with only one window around the vehicle .

Heat map thresholding helps in removing false positives from the final results

Result on few examples/test images are shown below


Heat Map
Car Positions
Heat Map
Car Positions
0 100 200 300 400 500 600 700
0 200 400 600 800 1000 1200


Heat Map
Car Positions
Heat Map
Car Positions
0 100 200 300 400 500 600 700
0 200 400 600 800 1000 1200



Video Implementation

To further smoothing out the result I have used heat map history of last 7 frames, I am also using thresholding on heat map history.

test_videos_output/output.mp4 is the final output result video of given project video.

Discussion

There are few problems with the current pipeline, the most important one is that it cannot be used in real-time situation as the current pipeline is very slow to be used in real-time.

Pipeline is also not able to classify other vehicle like truck, cars where positon of the car is other than provided in the training dataset. For which we need to use larger dataset.

Pipeline is also not able to detect vehicle position accurately , it is just a approximate location , in real time self-driving car we need to rely on other sensors to get more accurate location of surrounding vehicles