ND013 - Self-Driving Car

Vehicle Detection

The goals / steps of this project are the following:

- Train a SVM classifier based on the sample vehicle and non-vehicle images provided
- Create a sliding window of various sizes on the image to extract image patches to determine whether it is a vehicle or not from the SVM classifier
- Extract the features from the windows
- Form a heatmap from the sliding windows and the result of the SVM classifier to determine the exact location of the car
- Filter out the false positive from the heatmap
- Draw the bounding boxes of the detected vehicle to a video

Source Code

The source code of the main executable is 'vehicle_detection.py'. It uses function from the 'util' folder. Inside the folder, there are 2 more source code. 'draw.py' contains the function to generate sliding window and also draw bounding boxes. 'classifier.py ' mainly does the classifying work and also features extraction.

Classification

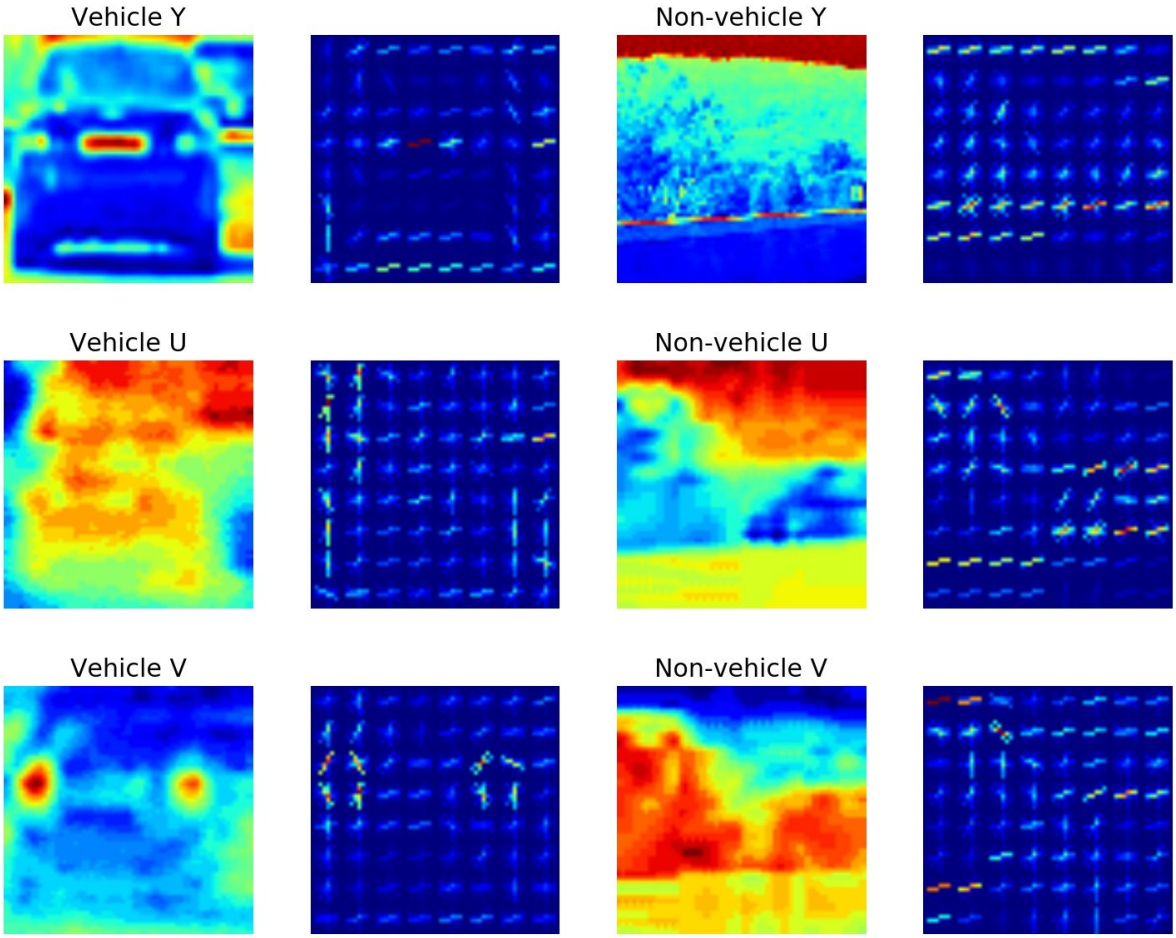
In order to do classification using SVM, the prior step is the features extraction.

Features Extraction

For features extraction, I only uses HoG and spatial binning. Although I included color histogram function, but I did not use it mainly because it doesn't improve the detection much and also slow down overall detection. I tried all the colorspaces including BGR, HLS, YUV, and YCrCb. I found that BGR and HLS perform poorly during detection whereby many false positive arises especially on the yellow lane and tree. I settled with YUV

because it yielded lesser false positives for my model. Both HoG and spatial binning are performed on the YUV colorspace. For HoG, I started with 18 orientations and 3 cells per block, however it uses too much memory due during the training phase. In the end, I reduced the HoG setting to 9 orientations and 3 cells per block. The image below are the HoG samples performed on 2 images (vehicle and non-vehicle). The HoG is performed for all channels (Y,U,V).

Original Sample Images



HoG Samples

Beside HoG, I uses spatial binning too. It started with spatial size of (32,32). In the end, it is reduced to (16,16) to increase the speed of detection. The combinations of features extraction is done at 'extract_features' at 'draw.py'

Classifier

I chose SVM as the classifier for vehicle detection because it works well with HoG features. At first, SVC (RBF) is used. However, the detection took too long (>2mins per image). I changed to LinearSVC to find out that the accuracy is similar (0.9935 on LinearSVC vs 0.991 on RBF). Using LinearSVC, the detection is much faster (~20 secs per

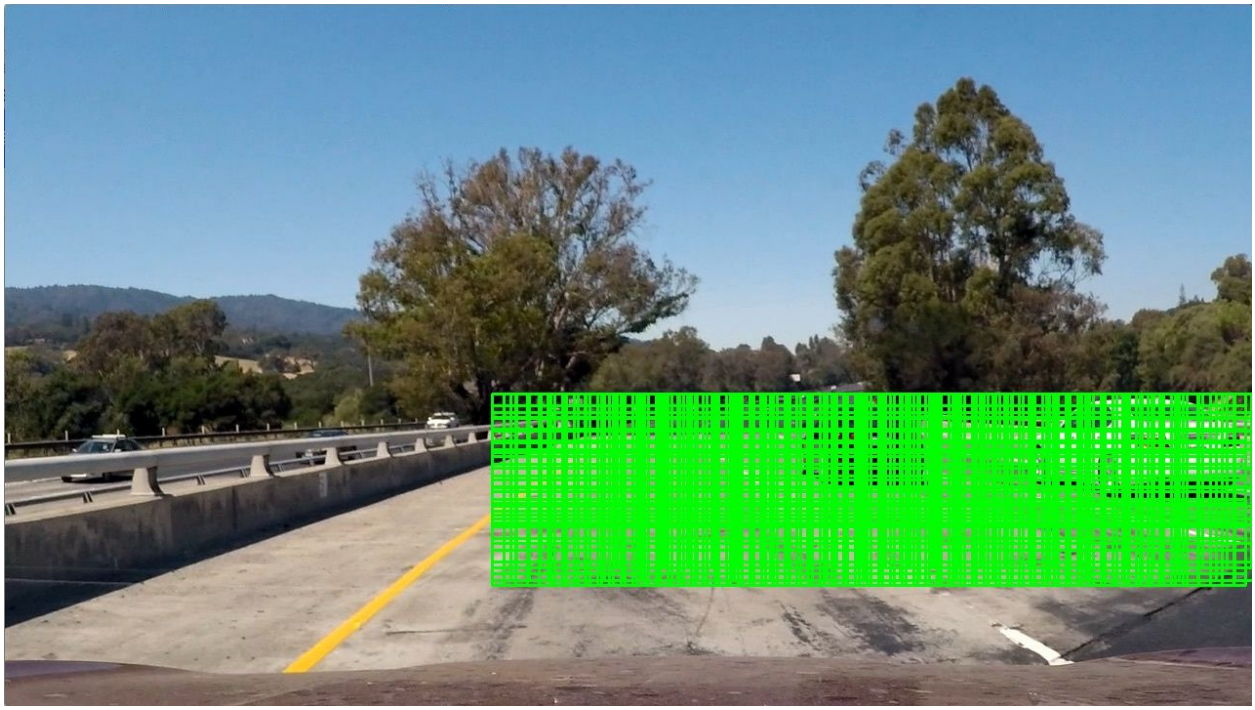
image). The classifier is trained by executing ‘classifier.py’ which will output a scaler and also classifier file which will then be loaded into the main executable.

For the data, I used the data prepared by udacity. However, I included some other false positives that arises from previous detection as non-vehicle. This further improve the accuracy during the detection.

Some examples of false positive

Sliding Windows

A sliding windows is generated on top of the image to determine the ROI to be extracted.



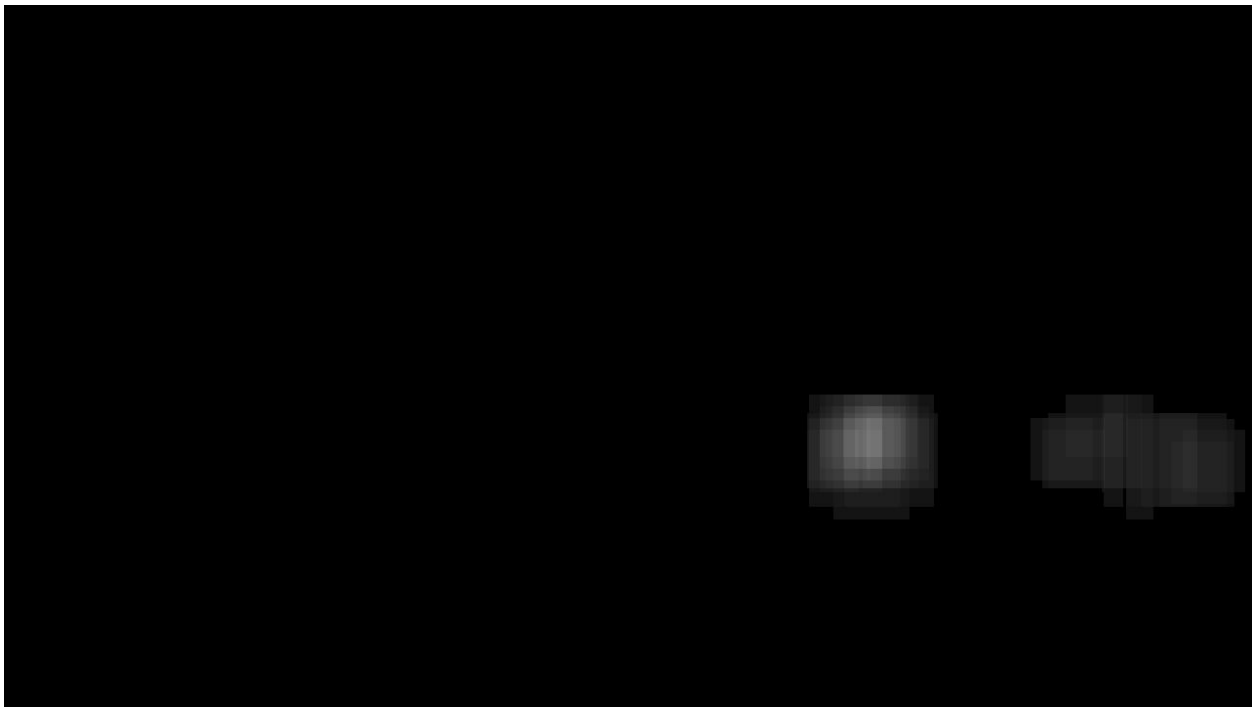
I only uses a small portion of the image for the sliding window. It serves 2 purpose which are to reduce the false positive (that occurs a lot in the trees region) and also to quicken the process of detection. 3 sizes of window are used (64, 64), (96, 96), (128, 128). The windows starts from 500 (x-axis) until 600 (y-axis). The source code for sliding windows is at ‘generate_sliding_windows’ at ‘draw.py’.

Vehicle Detection

In order to do vehicle detection, ‘generate_sliding_windows’ is called to get the bounding boxes of the ROI. Each roi is then pass to the SVM for detection to determine whether it is vehicle or non-vehicle. The following image is the result of the detection:-

Step 1: Detection

From the image, we can see the boxes in green determine the ROI that is classified as vehicle. Heatmap is the generated from the bounding boxes.



Step 2: Heatmap

The brightness of the pixels determine the confidences of the location of the vehicle. The region with less than 3 bounding boxes are discarded.

Step 3: Labeling

Using the function from skimage called ‘label’, the location of the vehicle can be determine from the heatmap.

Video

To reduce the false positive in the video, I included an additional 10 heatmap buffer for averaging purpose. (Line 32 - Line 37 at vehicle_detection.py). After doing the 10 frames averaging, other false positive diminished and the bounding boxes are more stable.

The video is processed and uploaded at:

https://youtu.be/ZlLao3OxD6o

Discussion

There are many ways to improve the pipeline including using deep learning method like SSD and YOLO to speed up the detection and also localization of the vehicle. Using SVM + features is often slow because of the amount of sliding windows.

My method will fail when there are too many car together (e.g. traffic jam) whereby the `label` function will determine all of them as a single vehicle.