# Vehicle Detection Project

## Software Modules

The major implementation of this project is `vd module` which is locale under `vd folder` :

### 1. vd

vd is `vehicle detection` implementation for this project.

#### 1.1 vd.feat_ext.FeatureExtracter

Histogram of Oriented Gradients (HOG), color special, bin histogram feature extraction implementation

#### 1.2 vd.agent.VehicleDetectAgent

The learning agent for detecting vehicle implementation

#### 1.3 vd.env.VehicleDetectionEnv

The Environment is providing image and label for agent

### 2. sdcp5

Except vd, there is a file sdcp5.py implements the pipeline to generate final output image / video

### 3. vehicle_detection.ipynb

The code to generate images for the document can be found in this ipython notebook file.

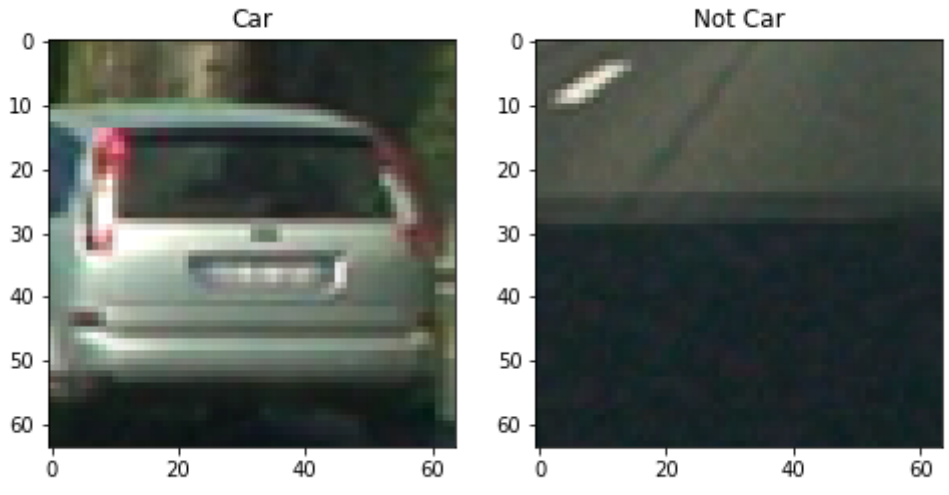
```
usage: python sdcp5.py [-f filename]

optional arguments:

-f  input video file name, defult is project_video.mp4
```

## Implementation

## Histogram of Oriented Gradients (HOG)

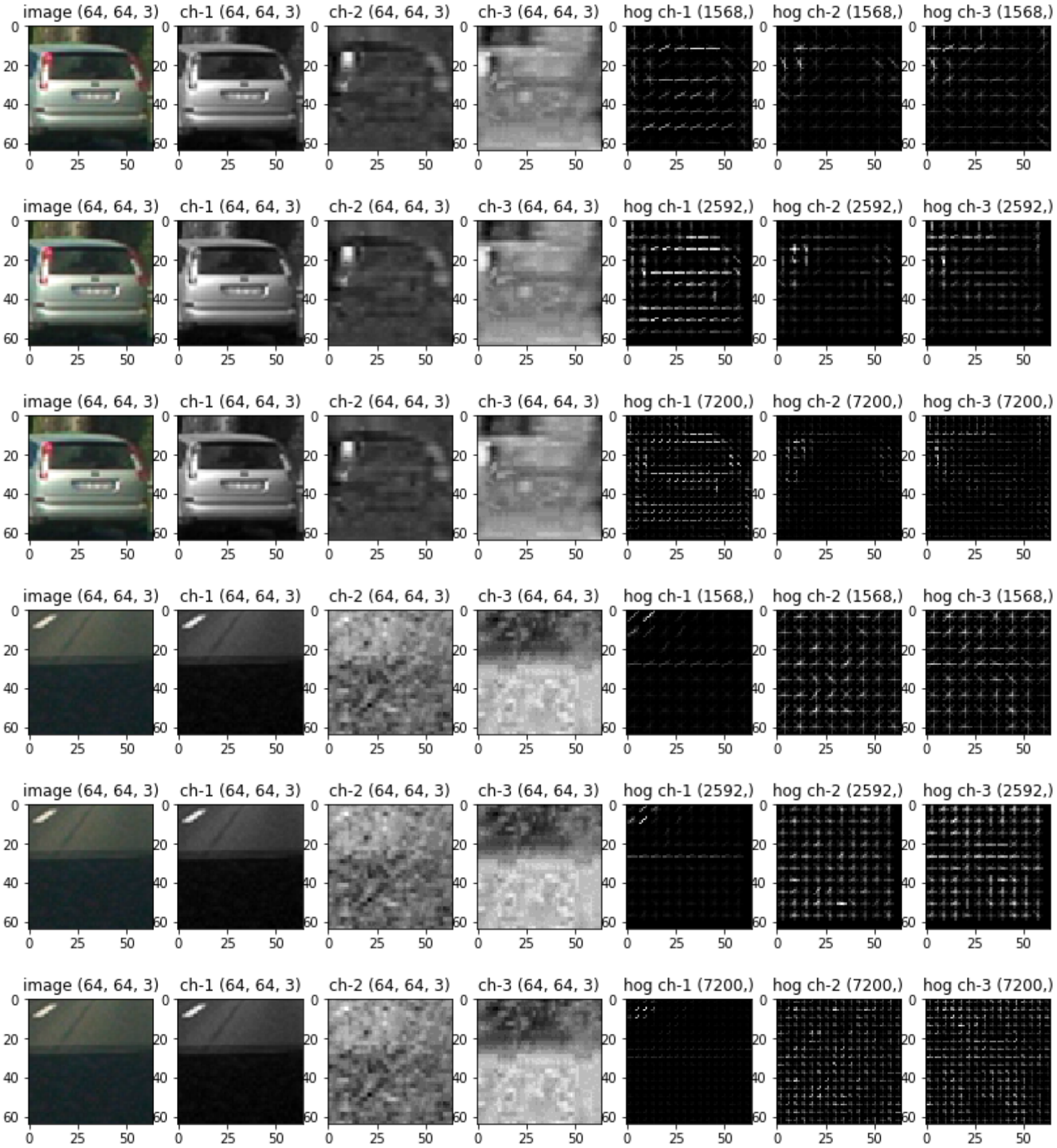
### 1. HOG feature extraction

HOG feature extraction core function is implemented at `vd/FeatureExtracter.py` . I used all the `vehicle` and `non-vehicle` images for building model, the following is an example of each of the `vehicle` and `non-vehicle` classes:




There are 3 kind parameters of hog function `skimage.hog()` should be decided: ( `orientations` , `pixels_per_cell` , and `cells_per_block` ). The other implementation decision is choosing color space to represent features, in my expriments, `YCrCb` and `HSV` are good, `YCrCb` is slightly better than `HSV` on detecting white color car. The following pictures using the

`YCrCb` color space and HOG parameters of `orientations=8 , pixels_per_cell=(8, 8);(6, 6);(4, 4)` and `cells_per_block=(2, 2)` The following pictures are car/not-car channels and hog features with different `pixels_per_cell` :




## 2. HOG parameter selection

After some experiments, I found there is no significant improvement of accuracy when orientations more than 8, but it will require more data to represent the features. The second parameter is `pixels_per_cell` , I have tried 3 values: ( `8` , `6` , `4` ), `4` is good but requires 7200 much more then the others, I decide to take `8` finally. The last parameter is `cells_per_block` , the value larger than 2 will need much more features but no significant improvement in accuracy on validation dataset. I decide `orientations=8 , pixels_per_cell=(8, 8)` and `cells_per_block=(2, 2)` in final implementation.

## 3. Classifier implement

I trained a linear SVM using `svm.SVC` with probability. That will help to set proper threshold for filtering false positive detection. The parameter `C` is 0.1 after parameter selection from `(0.1, 0.5, 1.0, 5.0, 10.0, 50.0, 100.0)` .

# Sliding Window Search

## 1. Implementation

The implementation can be found at `vd.agent.Partitioner` in `agent.py` . Sliding window design should take well considering on trade-off between accuracy and performance. High density will get better result but it requires more computing time. I create 3 types sliding window: finner: `(72, 56)` , middle: `(132, 96)` , coarser: `(192, 176)` . The program will only scan the bottom half in a image. The sliding windows looks like the figure:

## 2. Apply sliding window on classifier

Once the sliding windows are generated, we could apply it on classifier for detecting vehicle. According to previous design considering, I am using YCrCb 3-channel HOG features with `orientations=8 , pixels_per_cell=(8, 8)` and `cells_per_block=(2, 2)` plus spatially binned color and histograms of color in the feature vector. The result on test images are listed as the following:

# Video Implementation

## 1. Video output

The video could be generated through the following command:

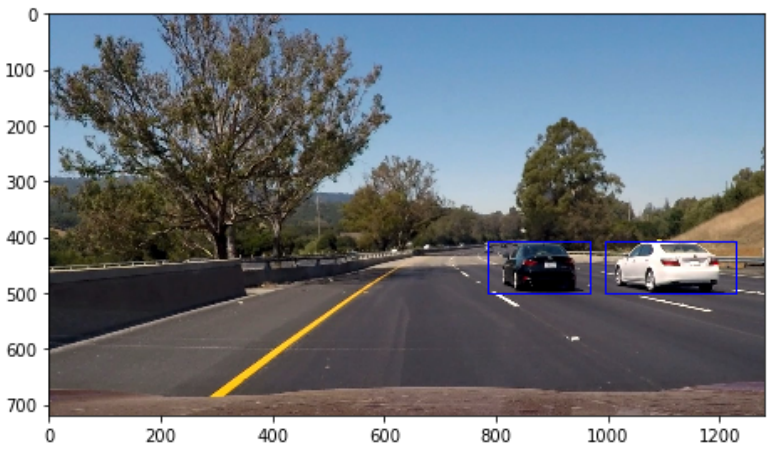
```
python sdcp5.py
```

It may be required a while to process. Here's a link to my video result

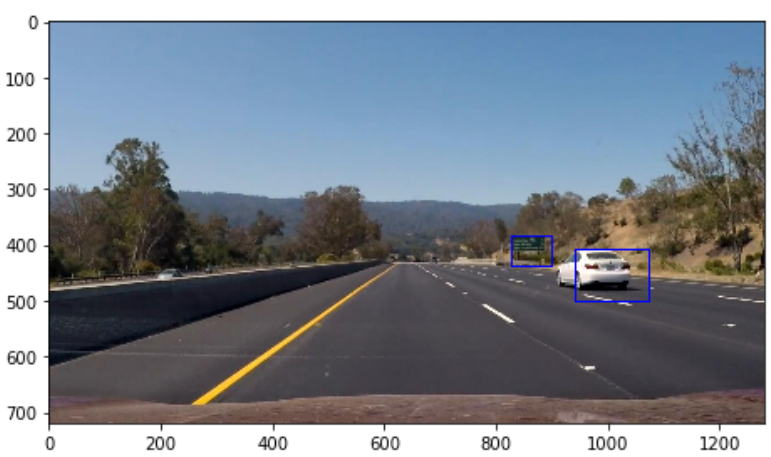
## 2. Pipeline fine tuning

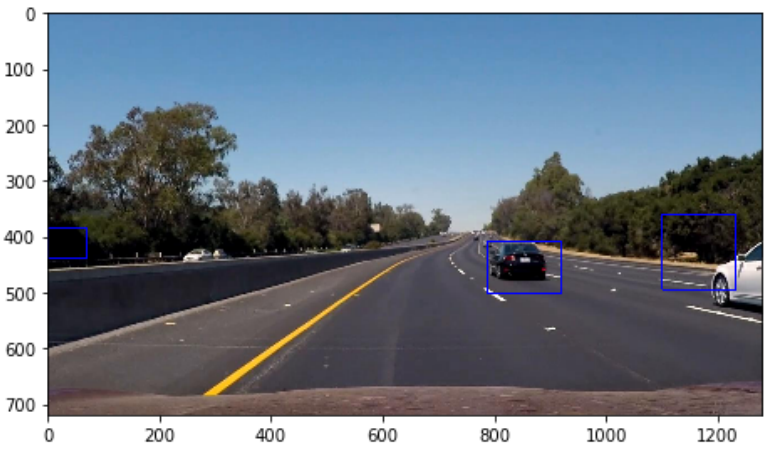
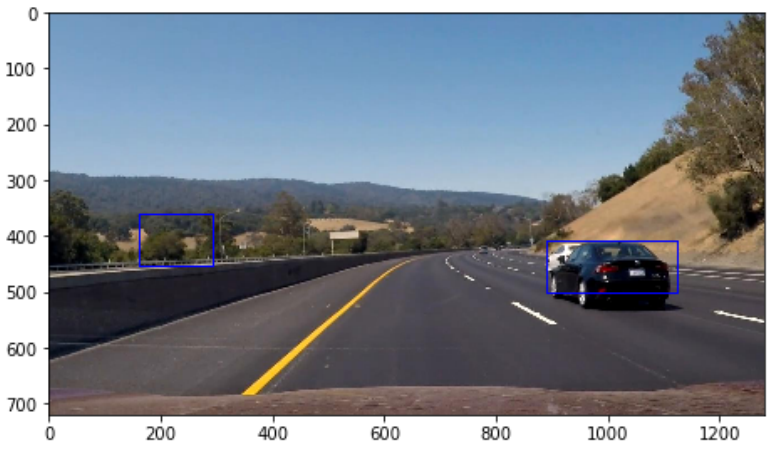
The validation error is quite good (acc:99.7%), we have confidence on the feature extraction and the classifier. In first pipeline implement (sdcp5.py.pipeline_0()), if the probability of vehicle > 0.5 occurs on any sliding window, the program will report the vehicle is detected. It looks good at test images as the following figure:






When applying the setting on project video, it occurs false positive on some frames, like the following figure:

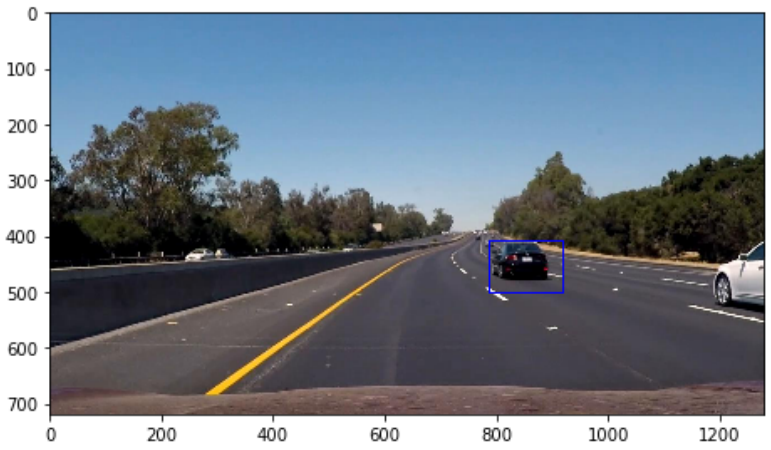
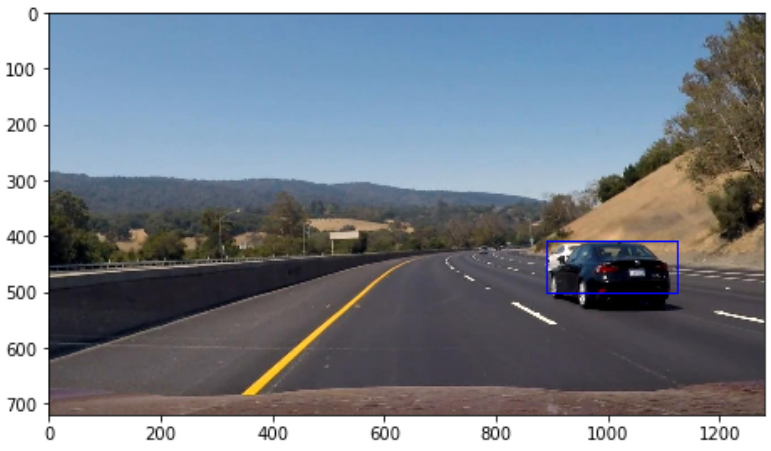



Increase sliding window will impact performance heavily, we try to increase the threshold of classifier confidence level from 0.5 to 0.95. The following figure is the result of applying it on test images:

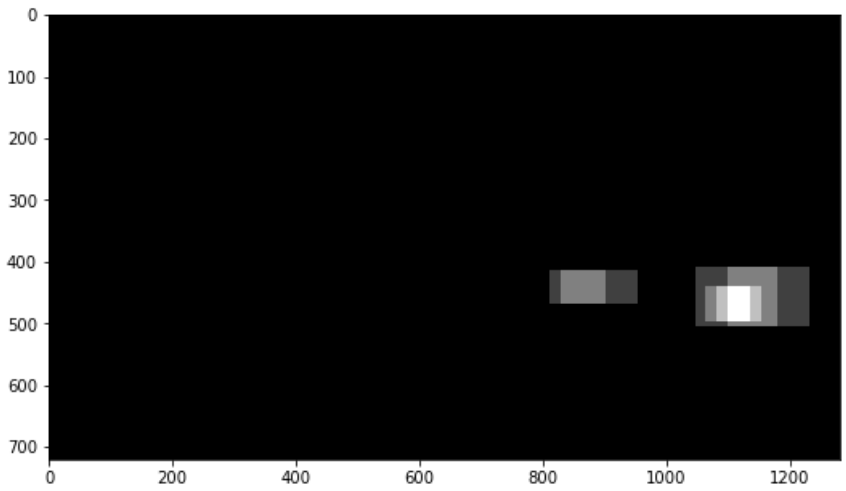
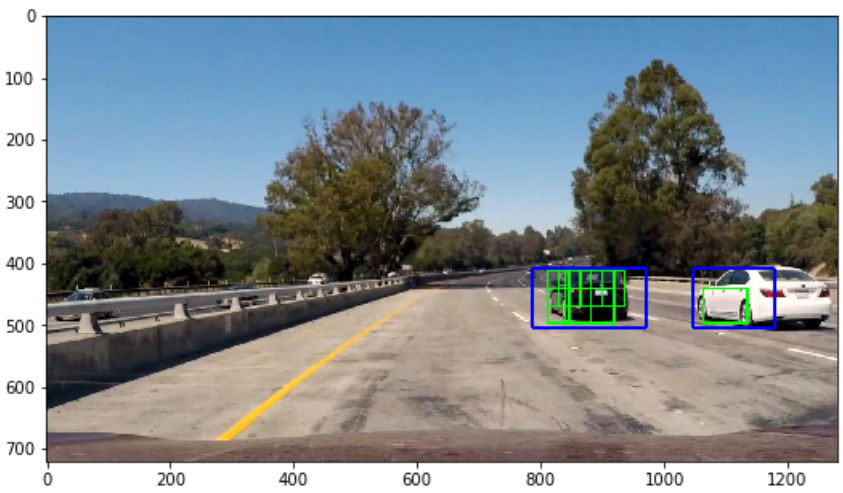
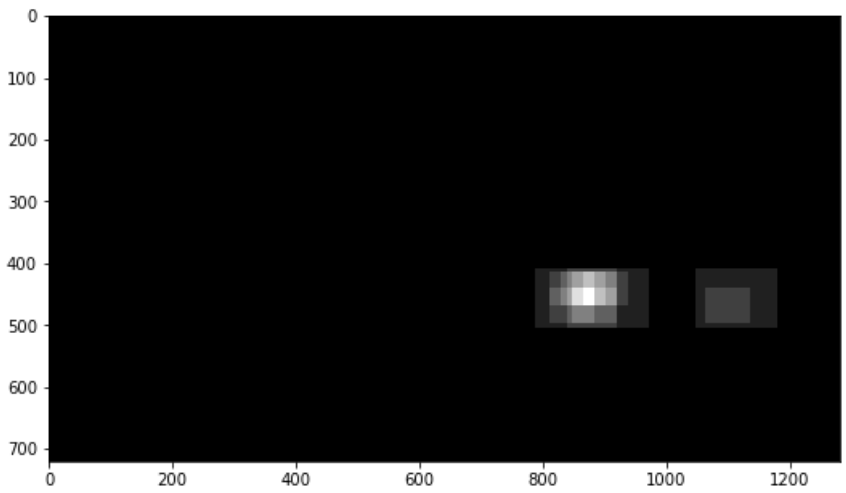
Then we test it on false positive images in original setting, the result is the following figure:

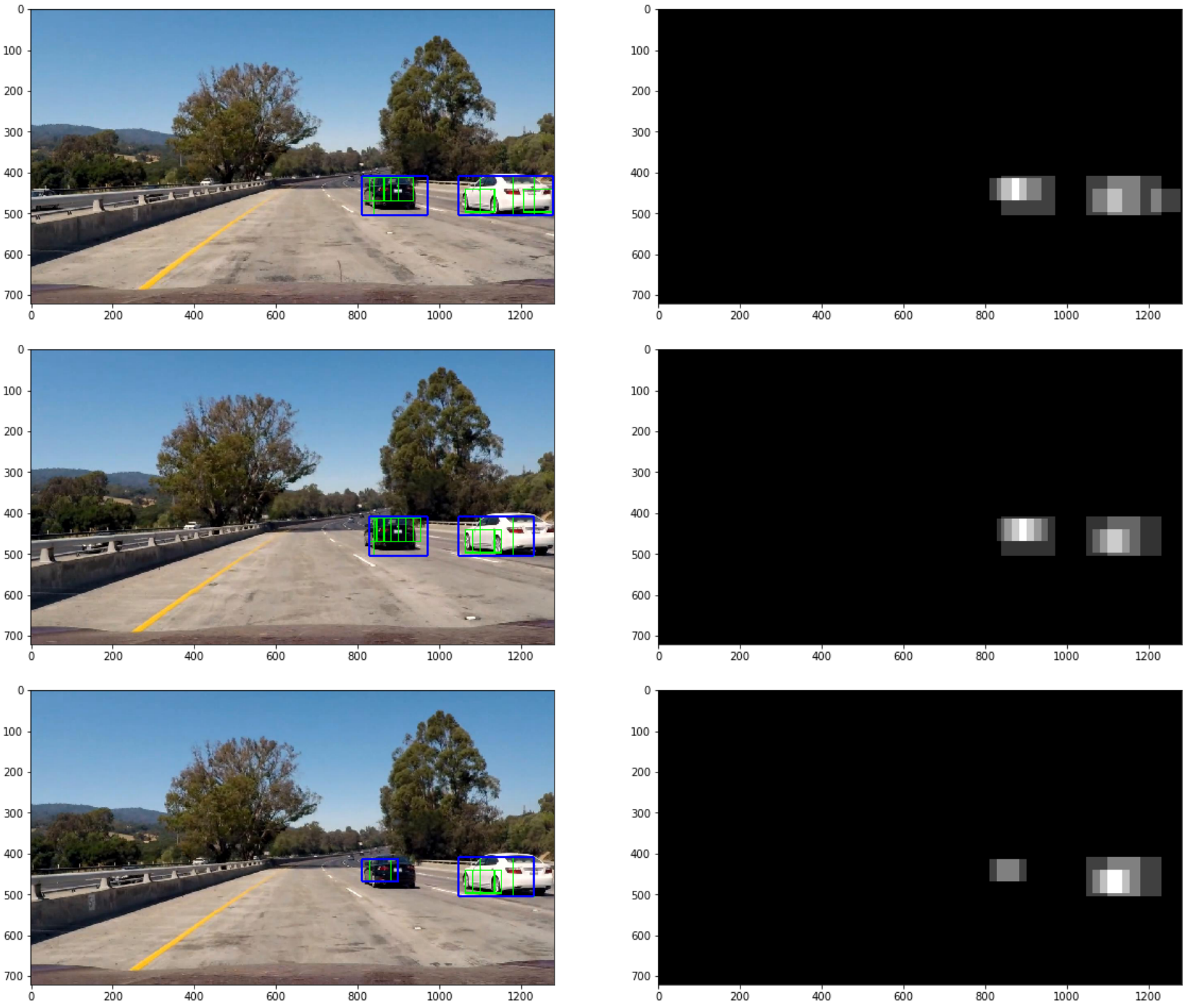



The `scipy.ndimage.measurements.label()` is used to identify individual blobs in the heatmap in `vd.VehicleDetectAgent.action()`. Here's an example result showing the heatmap from a series of frames of video, the result of `scipy.ndimage.measurements.label()` and the bounding boxes then overlaid on the last frame of video:

## Here are six frames and their corresponding heatmaps:



**Here the resulting bounding boxes are drawn onto the last frame in the series:**

## Discussion

### 1. Stability

The implementation is frame by frame, the bounding box may not smooth in some period within frames. To apply smoother within frames may be better.

## 2. Performance and accuracy

If we put more sliding window and set higher threshold for heapmap, the result may be better. But it needs more computing time when applying pipeline. The trade off within performance and accuracy for a handy feature engineering process is hard to be judged. I will prefer to applying end-to-end deep learning object detection model. Some end-to-end Deep Learning object detection algorithm may achieve near real time speed on GPU (ex. SSD, YOLO) and they have better capability on generalization than and handy feature engineering process, I will try to apply these models to detect vehicle and combine with the advance lane detection in the future.