Vehicle Detection Project

The goals / steps of this project are the following:

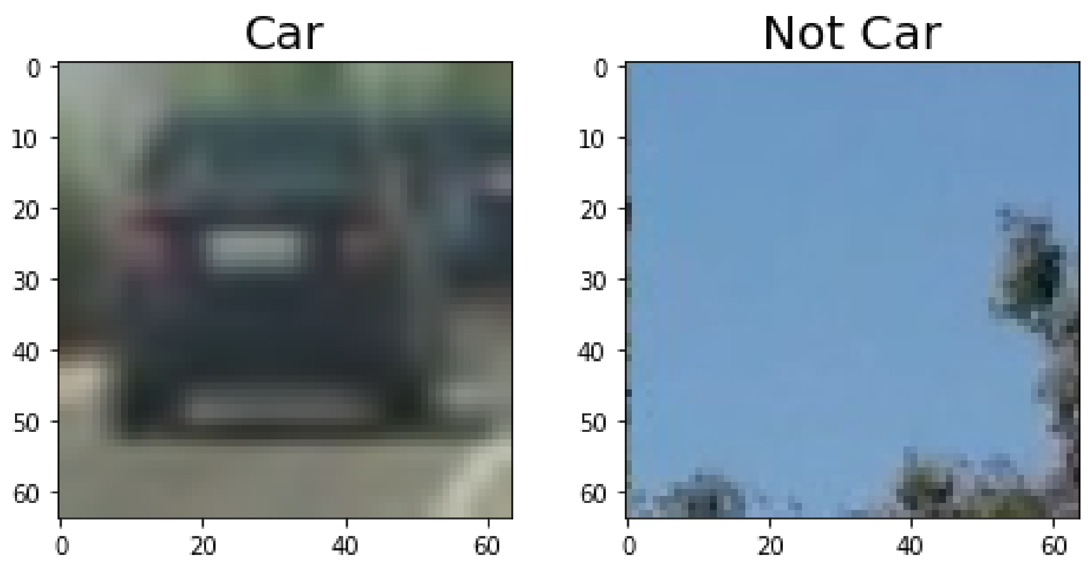
1. Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images and train a classifier Linear SVM classifier
2. Optionally, you can also apply a color transform and append binned color features, as well as histograms of color, to your HOG feature vector.
3. Note: for those first two steps don't forget to normalize your features and randomize a selection for training and testing.
4. Implement a sliding-window technique and use your trained classifier to search for vehicles in images.
5. 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.
6. Estimate a bounding box for vehicles detected.

### Histogram of Oriented Gradients (HOG)

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

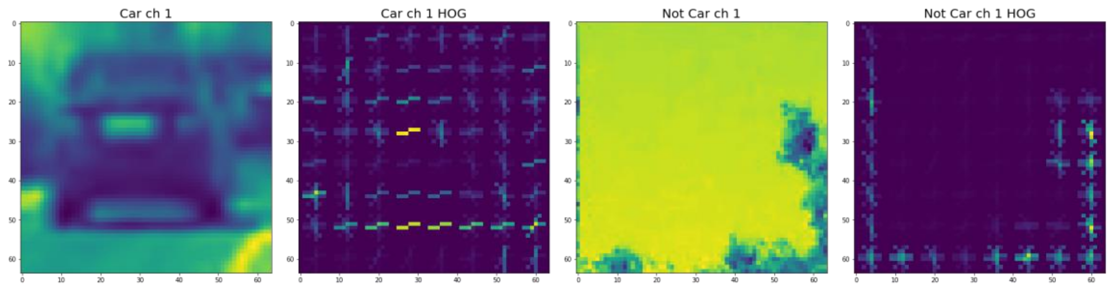
The code for this step is contained in the 2th code cell of the vehicle_detection_code.ipynb.

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:



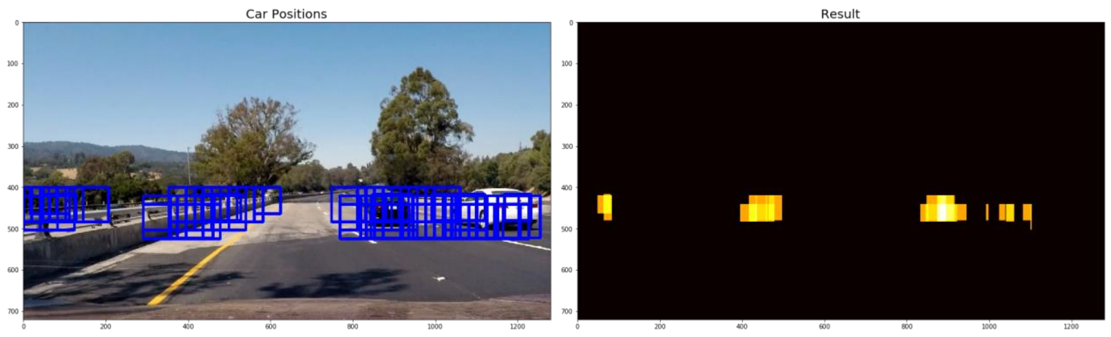
I then explored different color spaces and different `skimage.hog()` parameters (`orientations`, `pixels_per_cell`, and `cells_per_block`). I grabbed random images from each of the two classes and displayed them to get a feel for what the `skimage.hog()` output looks like.

Here is an example(HOG_example) using the `YCrCb` color space and HOG parameters of `orientations=9`, `pixels_per_cell=(8, 8)` and `cells_per_block=(2, 2) '(The code for this step is contained in the 4th code cell of the vehicle_detection_code.ipynb.(:



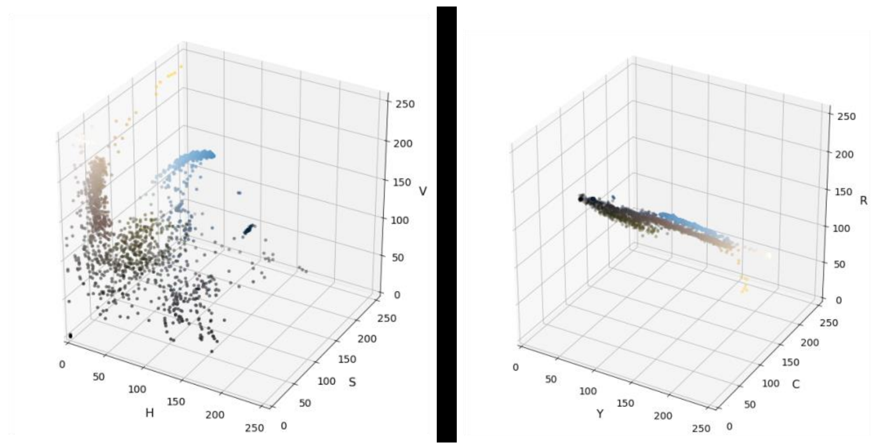
#### 2. Explain how you settled on your final choice of HOG parameters.

Firstly, I used different color space: RGB,HSV,LUV,HLS.YUV,YCrcb. But the results of various color spaces are different. For example, when I using HSV color space, the result of vehicle detection is following:



From picture, we can see that there are too many wrong slide windows that judge it has vehicle in the image. However， using color space:RGB, YUV, LUV,YCrCb have less wrong slide windows in the image.

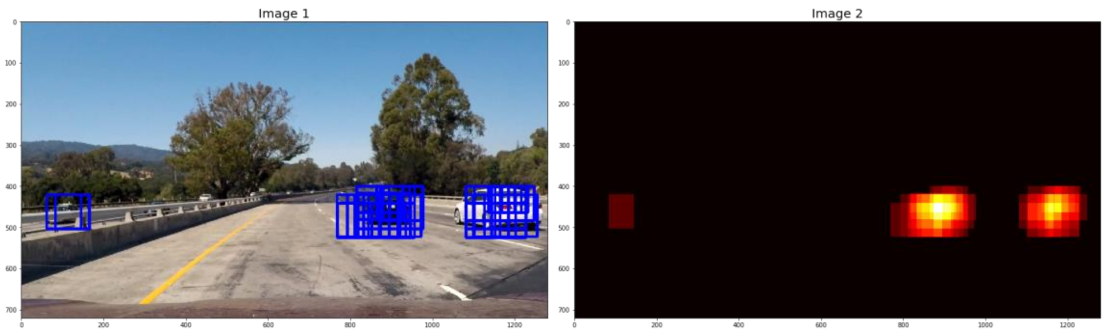
In the 3th code cell of the vehicle_detection_code.ipynb, I explored color space with Plot pixels in 3D. Color spaces like HSV and HLS have obviously separate for test image in 3D but it causes more wrong slide windows. However, Color spaces like RGB, YUV, LUV, YCrCb seems not obviously detached (See color space Plot pixels in 3D(left is HSV and right is YCrCb)) and it has less wrong slide windows.



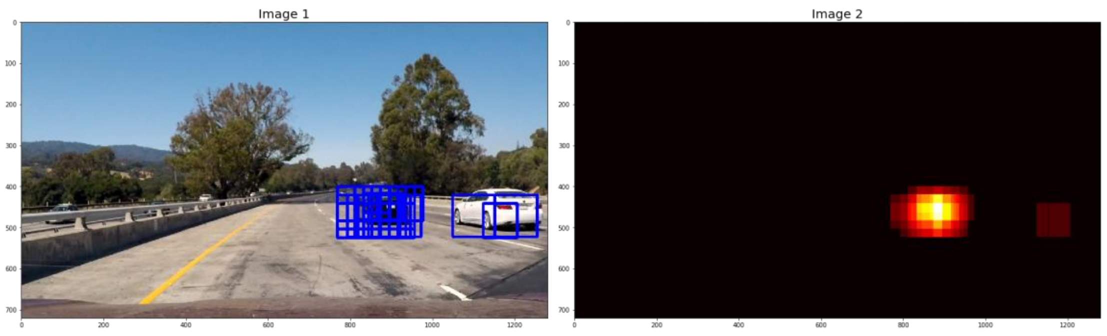
Finally, I chose color space YCrCb because it has better than others with combinations of parameters.

Then the orientation value was selected by many testing. The result of vehicle detection of orientation = 5 and orientation =15 was following:

Orientation = 5



Orientation = 15



From the picture, we can see that the bigger orientation value causes that it is hard to find some not obvious vehicles. However, the bigger orientation value also has less wrong slide windows. So for this project, I chose the optimize orientation value is 9 with many tests.

For `pixels_per_cell=(8, 8)` and `cells_per_block=(2, 2)`. These will affect overlapping descriptor so that it affects the result of vehicle detection. According to Dalal and Triggs(paper references: Histograms of Oriented Gradients for Human Detection), using suitable overlapping descriptor blocks can decrease miss rate of image detection and `pixels_per_cell=(8, 8)` and `cells_per_block=(2, 2)` are best selection after experiment. At the meaning time, these two parameters are more suitable than others by many tests, So I chose these two parameters.

#### 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).

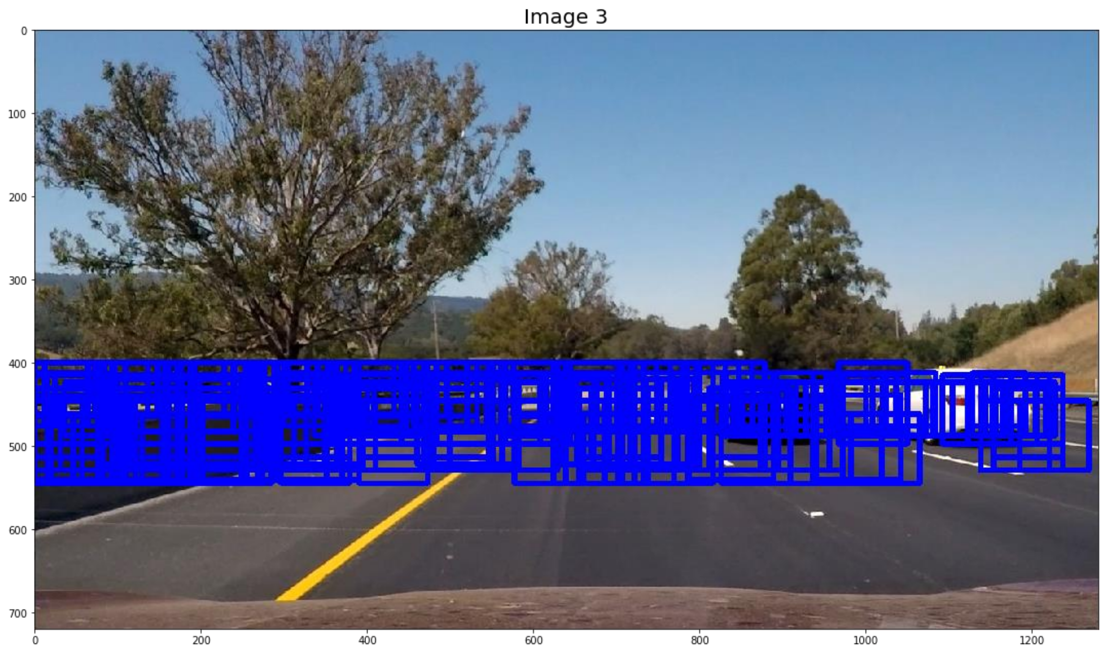
I trained a linear SVM using 9 orientations 8 pixels per cell and 2 cells per block(The code for this step is contained in the 5th code cell of the vehicle_detection_code.ipynb.)

Before using linear SVM, using StandardScaler()to normalize different features as a single feature vector and then split up data into randomized training and test sets.

### 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?

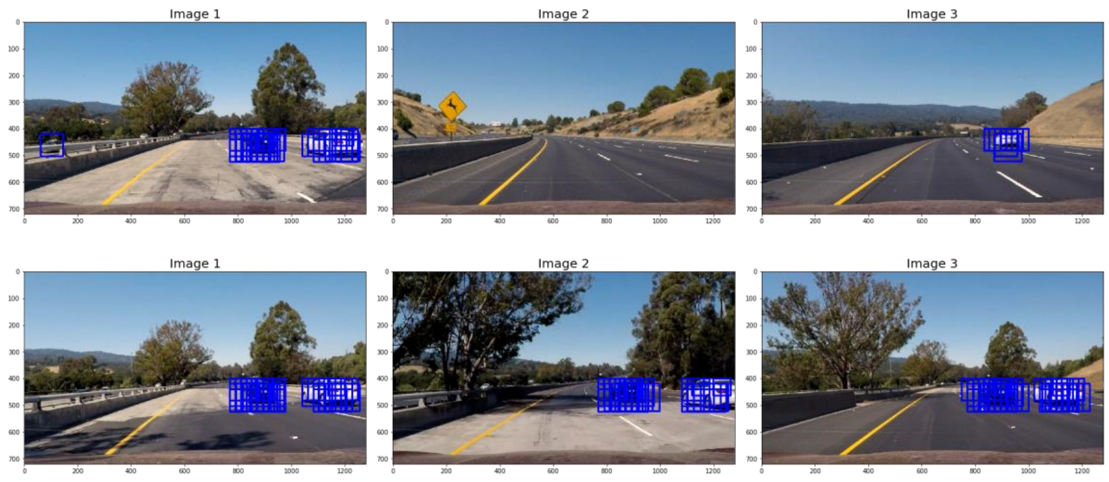
I decided to search random window positions from at random scales from 400 to 650 width of images(The code for this step is contained in the 6th code cell of the vehicle_detection_code.ipynb.) and came up with one example:



It is because, when the width of image < 400, the content of image is sky and others, not including the road. So it is not necessary to slide windows to search vehicles. As the farther the car is, the size of car is smaller in the image, it use various size of rectangle to search (max size is same as in the HOG features and min size is reduced by 3 times) and the overlap is same as in the HOG features due to the same pixels_per_cell and cells_per_block.

#### 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 two scales using YCrCb 3-channel HOG features plus spatially binned color and histograms of color in the feature vector, which provided a nice result(The code for this step is contained in the 6th code cell of the vehicle_detection_code.ipynb.).   Here are some 6 example images:



### 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 a [link to my video result](result_ycrcb.mp4)

#### 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 for this step is contained in the 10th code cell of the vehicle_detection_code.ipynb.

I recorded the positions of positive detections in each frame of the video. From the positive detections I created a heatmap and then thresholded(threshold =1) that map to identify vehicle positions. I then used `scipy.ndimage.measurements.label()` to identify individual blobs in the heatmap. I then assumed each blob corresponded to a vehicle. I constructed bounding boxes to cover the area of each blob detected.

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:

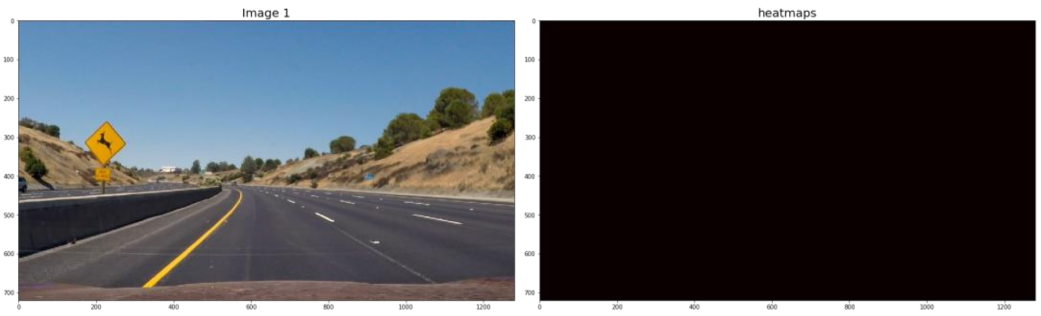


Image 1
heatmaps
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

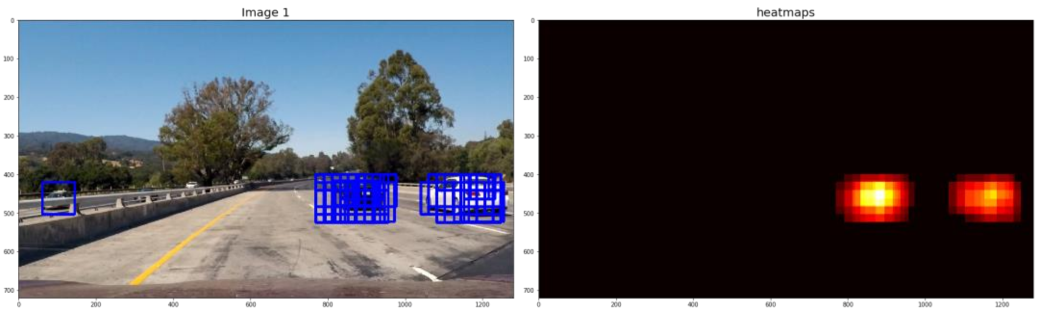


Image 1
heatmaps
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

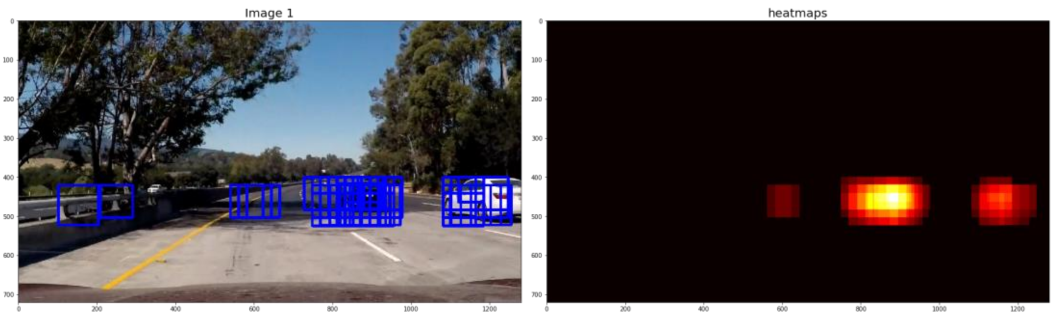


Image 1
heatmaps
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

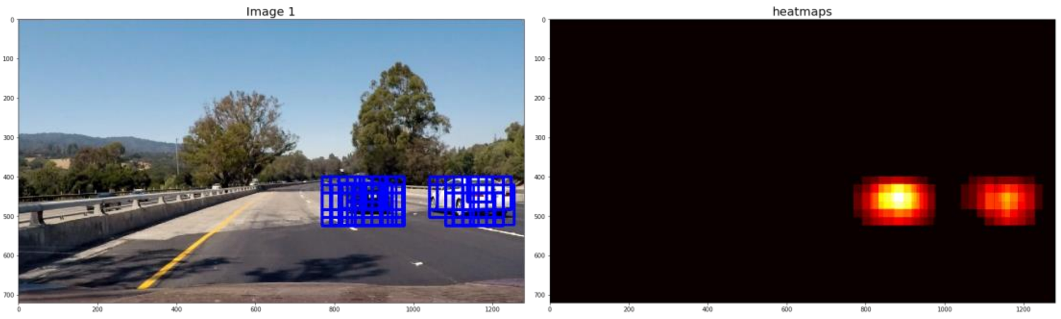


Image 1
heatmaps
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

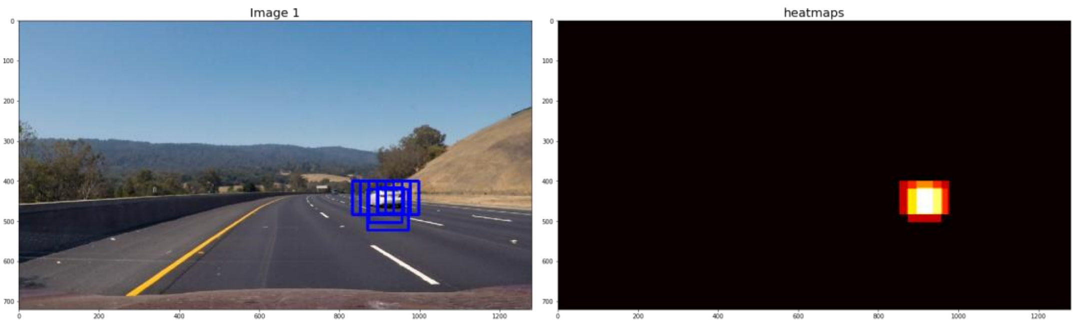
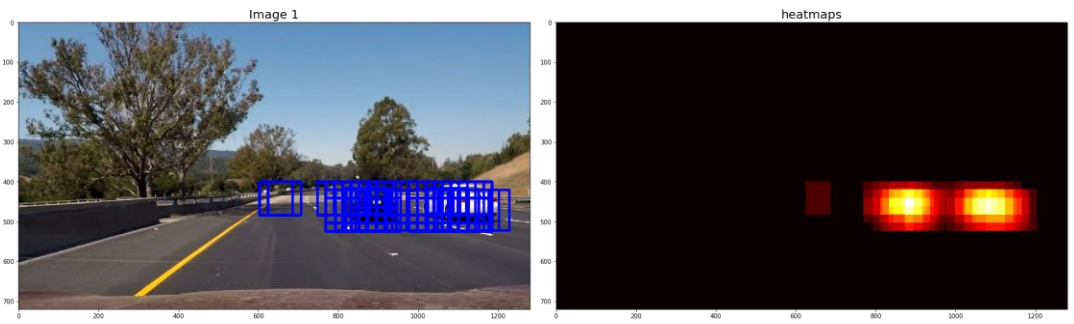
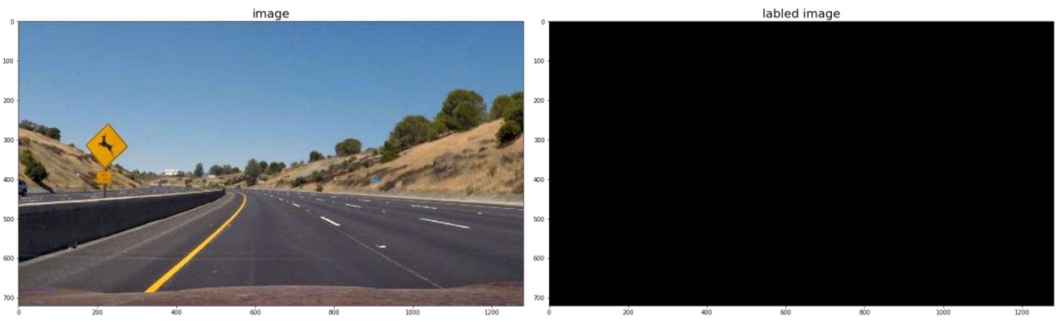
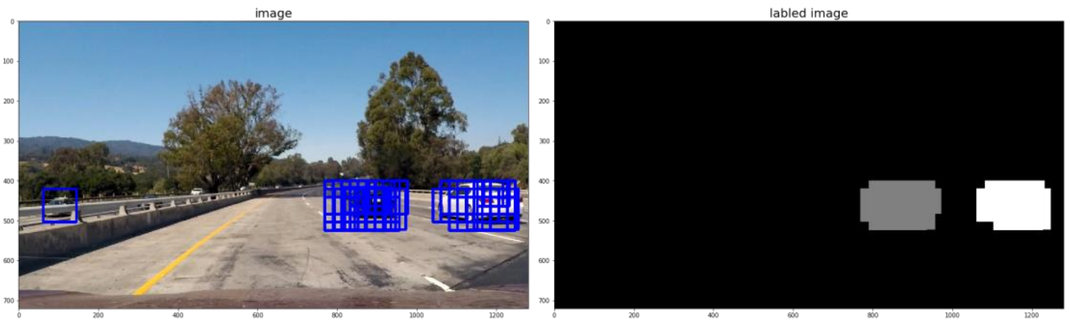


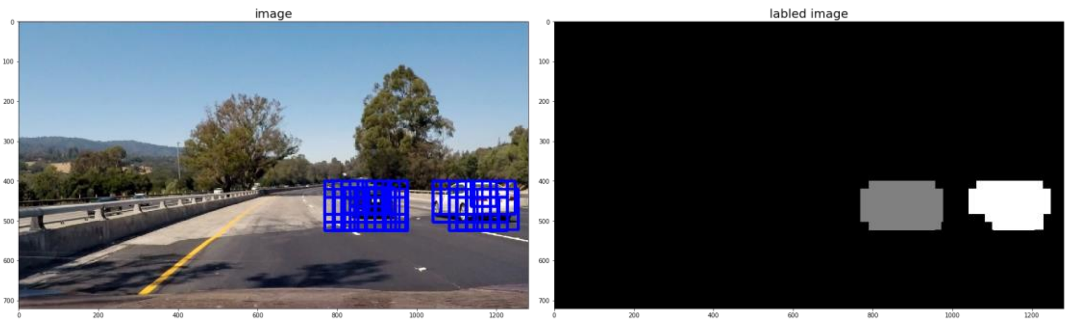
Image 1
heatmaps
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

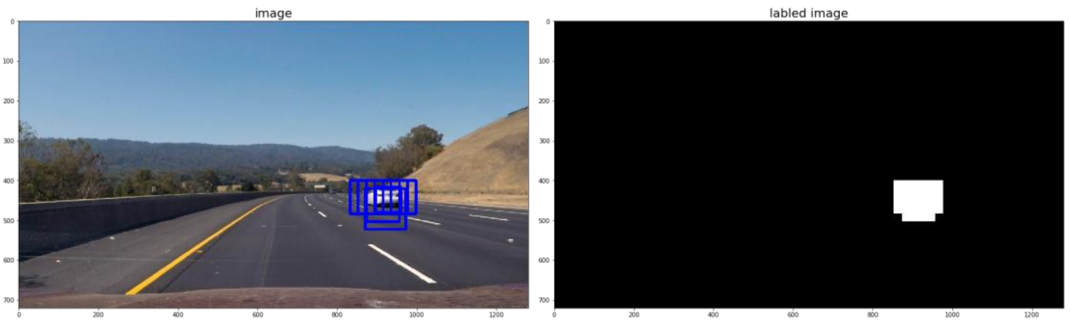


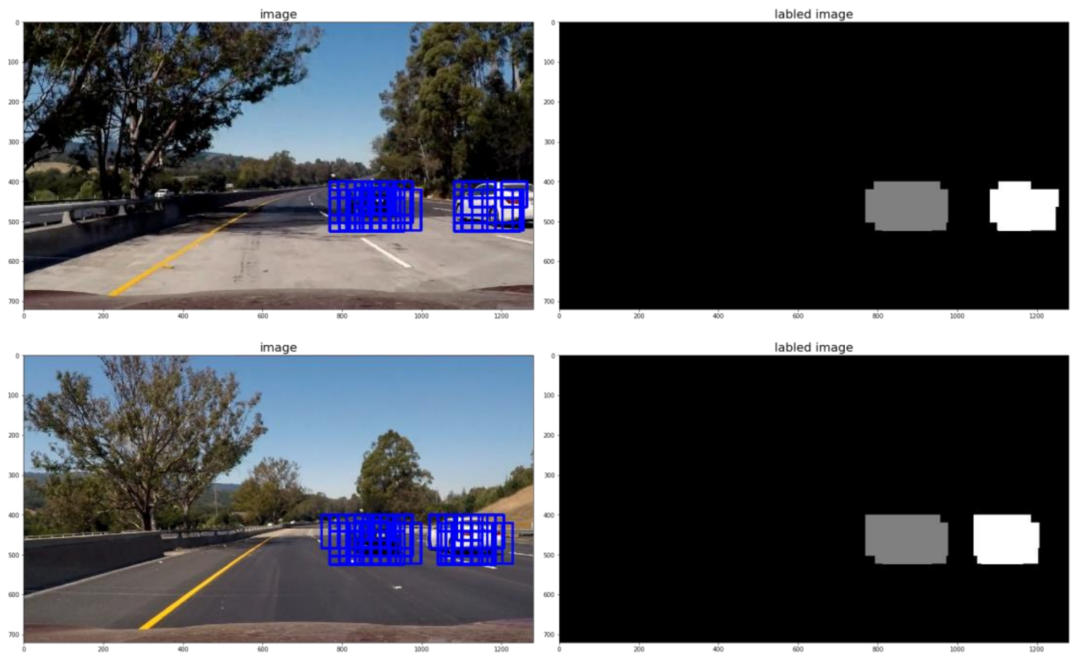
### Here is the output of `scipy.ndimage.measurements.label()` on the integrated heatmap from all six frames:



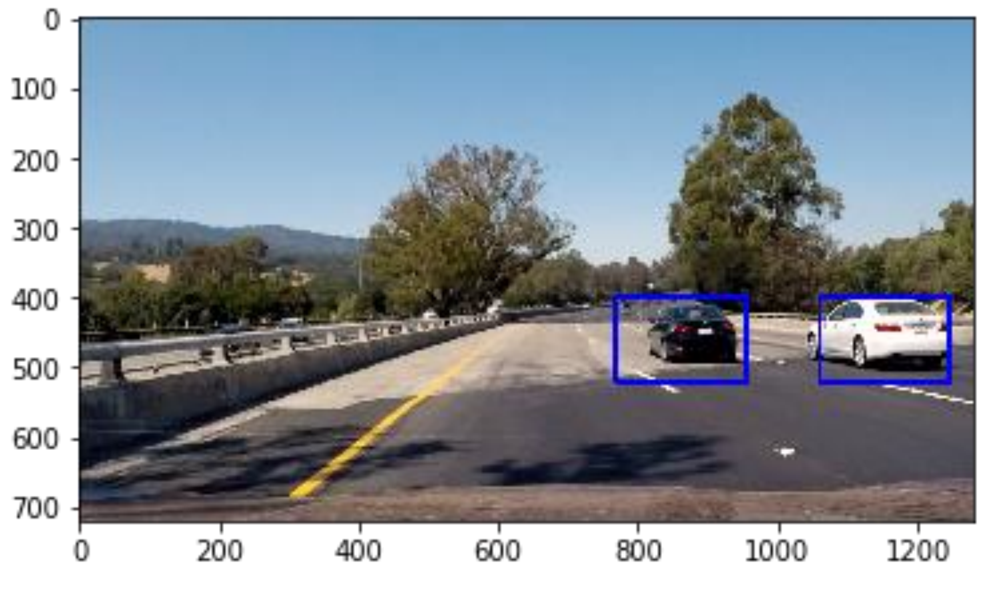








### Here the resulting bounding boxes are drawn onto the last frame in the series:



### 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?

1. When using the small set of data from amazon as‘vehicles’and ‘nonb-vehicles’data, the effect of vehicles feature is not good. After using larger project dataset (from udacity link), the effect on predict result is much better. So choosing a training data has a big effect on result of detection.

2. The effect of vehicle is always not good, especially, for small vehicles or some not obvious vehicle. Moreover, when 2 cars are together, it is not separate to draw boxes respectively.

In the future, I need to learn to more knowledge on detections like RCNN, Fast RCNN, YOLO and SSD.