# Vehicle Detection Project

***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.
- Note: for those first two steps don't forget to normalize your features and randomize a selection for training and testing.
- 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.

## Rubric (https://review.udacity.com/#!/rubrics/513/view) Points

### Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

### 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 code cell #12 and #15 of the IPython notebook Project_5_submission_v01.ipynb.**

**Before extracting HOG features, I read the vehicle and non-vehicle data. Here are examples of images of vehicle and non-vehicle images**

**cars dataset has 8792 records and notcars dataset has 8968 records.**

**As these numbers are close, we have nearly balanced dataset of car and non car images**

**After reading cars and notcars datasets, I wrote a function ( taking cue from Udacity lecture notes). I explored various color spaces and after lot of testing I narrowed down to 2 color spaces - YCrCb and YUV. These 2 color spaces give almost similar results. Finally I have kept YUV in the code.**

**Finding features function ( extract_features) is defined in code cell# of the ipython notebook. In this these types of features are extracted - Hog features, spatial features and color histogram features.**

***I tried various combinations of parameters for skimage Hog function.***

- I tried "orientations" from 5 to 9 and found out that 7 gives good results.
- I tried cell_per_block 2 and 4 and finally got good results with 4
- I tried spatial_size parameter (16,16) and (32,32) - both give almost similar results and used the lower value (16,16)
- I took pix_per_cell = 8 and cell_per_block = 4 for training dataset.

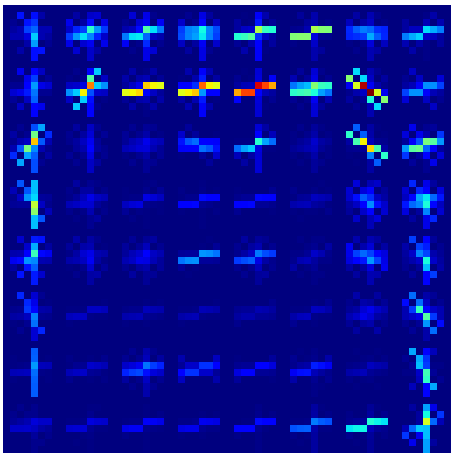
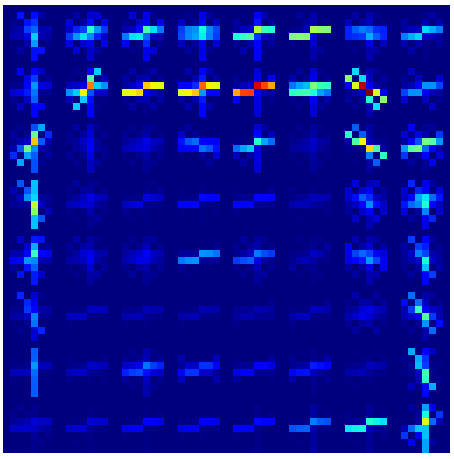
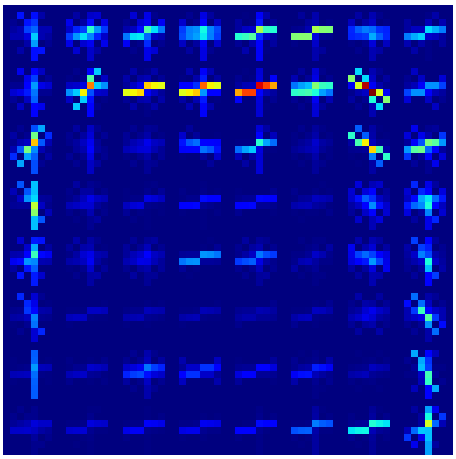
***I extracted features separately for cars and notcars dataset, then combined these features to make one single array of features.***

***Then these features are normalized using StandardScaler function of sklearn. This is done so that all features have equal weightage.***

***Then features dataset is split using sklearn train_test_split (80:20) to train and then test the classifier.***

***I then used sklearn LinearSVC ( support vector machine ) classifier. I tried using base SVC classifier also but it was taking too much time. I used C=0.1 for LinearSVC.***

***Here are examples of HOG features of 3 channels of a single image***

| Hog feature Channel 0 | Hog feature Channel 1 | Hog feature Channel 2 |
|---|---|---|
|  |  |  |

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

The code for this step is contained in the code cell #10 and #22 of the IPython notebook Project_5_submission_v01.ipynb.

I tried various combinations of parameters for skimage Hog function. I tried "orientations" from 5 to 9 and found out that 7 gives good results. I tried cell_per_block 2 and 4 and finally got good results with 4 I tried spatial_size parameter (16,16) and (32,32) - both give almost similar results and used the lower value (16,16) I took pix_per_cell = 8 and cell_per_block = 4 for training dataset

Test Accuracy values for various combinations are given below Although the best accuracy was with 9 orientations 8 pixels per cell and 2 cells per block but I found out that there were many false positives with this combination and I was getting good results with 7 orientations 8 pixels per cell and 4 cells per block. So I finally chose this combination

Using: 7 orientations 8 pixels per cell and 4 cells per block Feature vector length: 9264 21.11 Seconds to train SVC... Test Accuracy of SVC = 0.9854

Using: 8 orientations 8 pixels per cell and 4 cells per block Feature vector length: 10464 23.8 Seconds to train SVC... Test Accuracy of SVC = 0.9856

Using: 7 orientations 8 pixels per cell and 2 cells per block Feature vector length: 4980 12.41 Seconds to train SVC... Test Accuracy of SVC = 0.9882

Using: 8 orientations 8 pixels per cell and 2 cells per block Feature vector length: 5568 12.63 Seconds to train SVC... Test Accuracy of SVC = 0.9882

Using: 9 orientations 8 pixels per cell and 2 cells per block Feature vector length: 6156 14.92 Seconds to train SVC... Test Accuracy of SVC = 0.9893

**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 then used sklearn LinearSVC ( support vector machine ) classifier. I tried using base SVC classifier also but it was taking too much time. I used C=0.1 for LinearSVC.***

***The code for this step is contained in the code cell #22 of the IPython notebook Project_5_v01.ipynb.***

## 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?**

***The code for this step is contained in the code cell #27 of the IPython notebook Project_5_submission_v01.ipynb.***

***Find_cars function contains this code.***

***I decided to use lower half of the image (from 384 to 656 pixels) as search window to stay in view of camera and avoiding tree tops and and far off places.***

***I used scale of 1.5 and used windows of sizes (16,24,32,48,56,64) to match images. I used 50%( 8 pixels per cells, 4 cells per step) and 75% overlap(8 pixels per cells, 2 cells per step) and got better results with 75% overlap. As the scale factor( 1.5) is greater than 1, the search window is reduced by this factor and therefor have used lower size windows for feature searching.***

***I further divided the window search as per window sizes. For the upper half part of the search window ( 384 + ( 656-384)/2 ) I used small window sizes ( 16,24,32) and for the lower part of the search window I used windows of sizes 48, 56, 64.***

***I tried various scale factors from 0.75 to 3 and found that we need to change window sizes for search according to the scale factor. FInally I have been able to find decent match with scale factor of 1.5 and window sizes of (16,24,32,48,56,64)***

***I think we can achieve similar results with other scales, only thing required will be to spend more time on tuning parameters, finding more window sizes and using different threshold values.***

*I used method of first finding HOG features for whole image ( based on window sizes) and then extracting features for sliding windows.*

**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?**

*I tried various combination of parameters for Hog feature generation, used different scale factors, window sizes for search, and different threshold values. Ultimately I used*

*7 orientations 8 pixels per cell and 4 cells per block Feature vector length: 9264*

*Scale factor 1.5*

*sliding windows of sizes - (16,24,32,48,56,64)*

*Threshold value - 4*

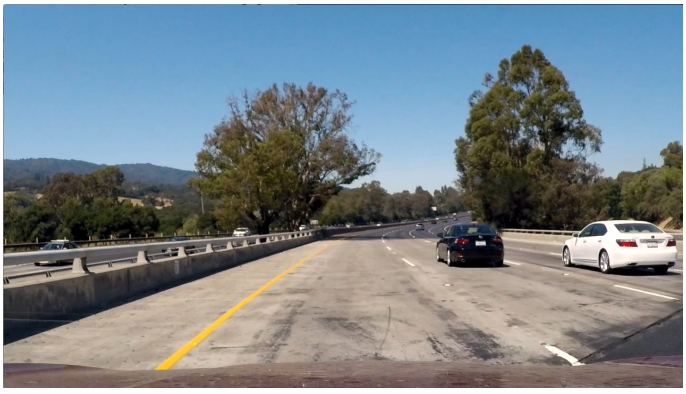
*I tried various combination of parameters for SVC and found out the best results were obtained with C = 0.1 for LinearSVC*

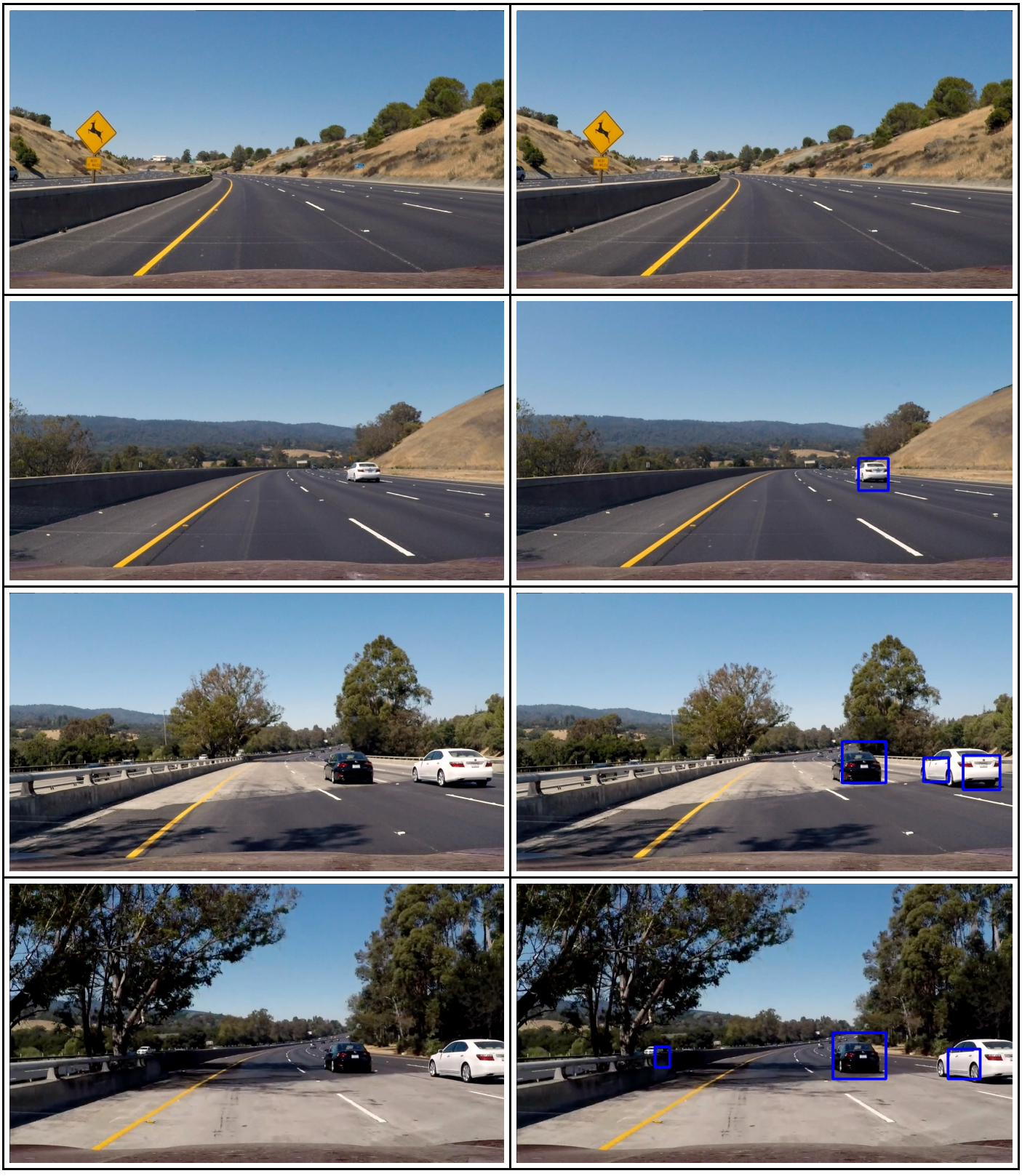
*and 7 orientations 8 pixels per cell and 4 cells per block Feature vector length: 9264*

*All three Hog Channels*

*spatial_size = (16, 16) - I chose 16, 16 instead of 32, 32 as there was not much improvement with 32,32 over 16,16*

*hist_bins = 32*

| Orig Image | Image with boxes |
| --- | --- |
|  |  |




## 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.)**

The output video is added in the zip file and its link is

https://youtu.be/S1psj_0gE64 (https://youtu.be/S1psj_0gE64)

**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 code cells #24, #27, #29 and #30 of the IPython notebook Project_5_submission_v01.ipynb.***

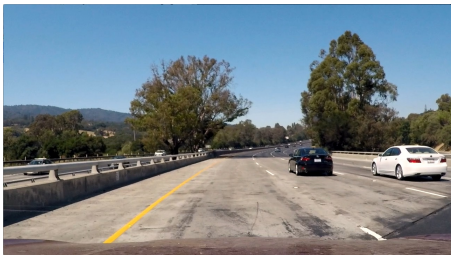
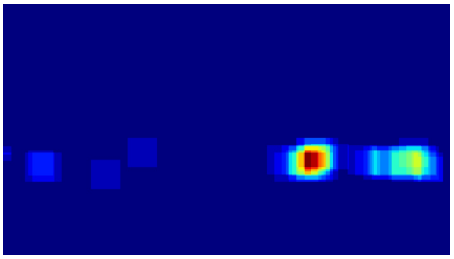
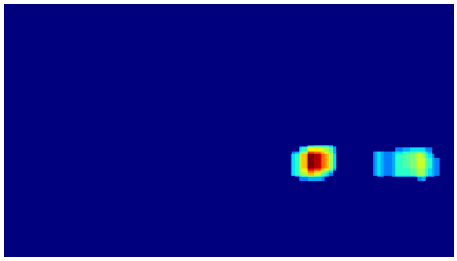
**For handling false positives, I used a combination of svc decision function and heatmaps. I used probability confidence value of 0.9 for making predictions and then used heat maps to keep most frequent boxes.**

**For removing false positives, I generated heatmaps of the images and used threshold value of 4 to remove less frequent boxes.**

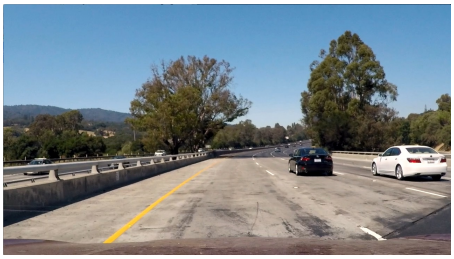
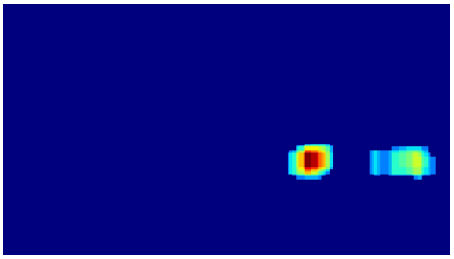
**After thresholding heatmaps, I used scipy label function to identify the individual blobs to draw them around objects.**

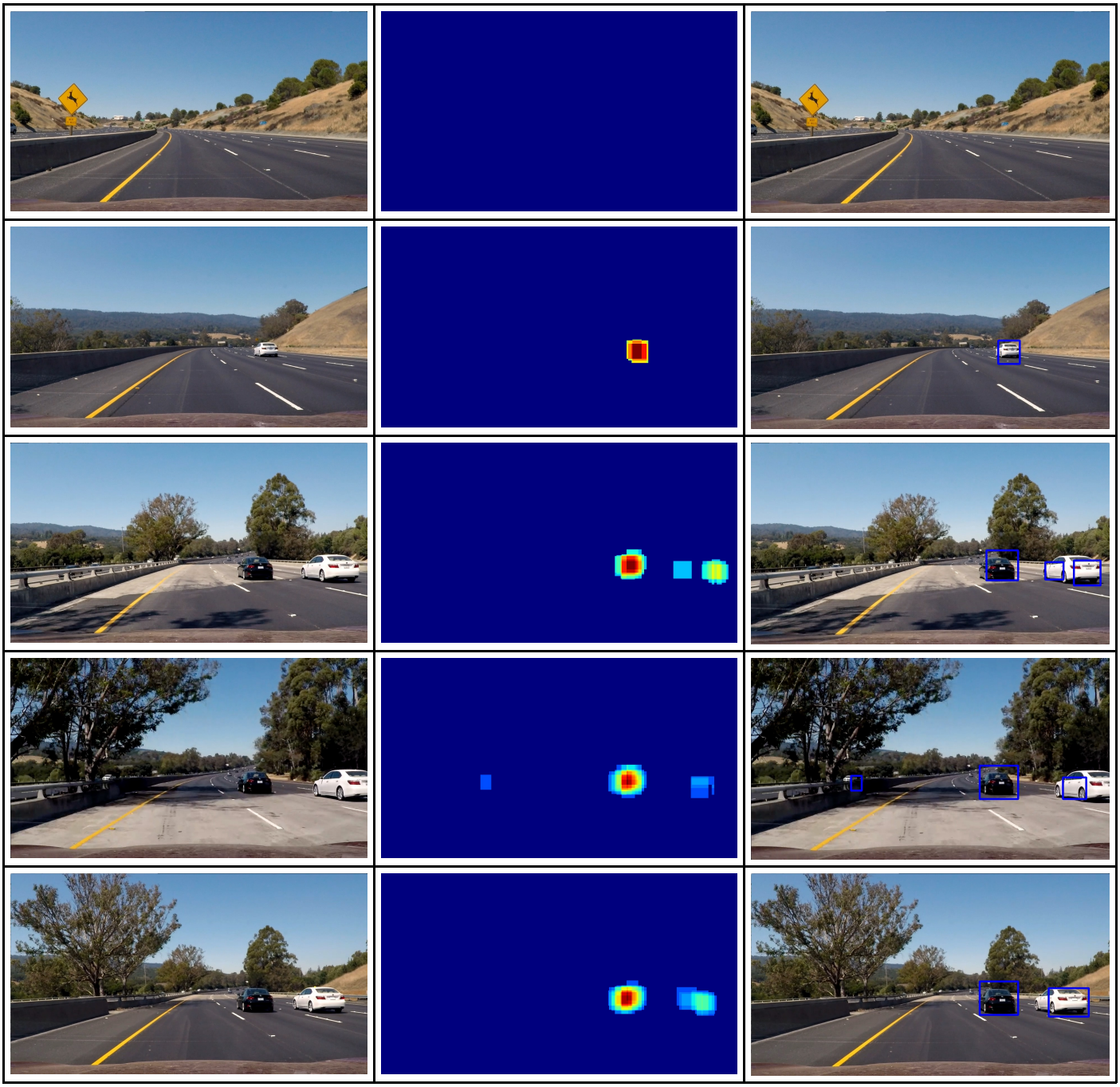
**Blobs were constructed using max, min left and right corners of the blobs.**

## Here are six frames and their corresponding heatmaps:

| Orig Image | HeatMap before threshold | HeatMap after threshold |
|---|---|---|
|  |  |  |

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

| Orig Image | HeatMap after threshold | Image with labels |
|---|---|---|
|  |  |  |



## 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?**

***Most of the time was spent on identifying scale factor and the window sizes and then removing false positives.***

***I tried a number of scale factors, window sizes and threshold values.***

***I tried more time on sclae factors, window sizes and thresholds and kept the trained SVC same. I think I can explore more on the other combination of parameter values.***

***I still see that the there are chances of improvement in building boxes around cars.***

***Above solution does identify cars on the left side of the road. I need to explore more about keeping\rejecting those objects.a***