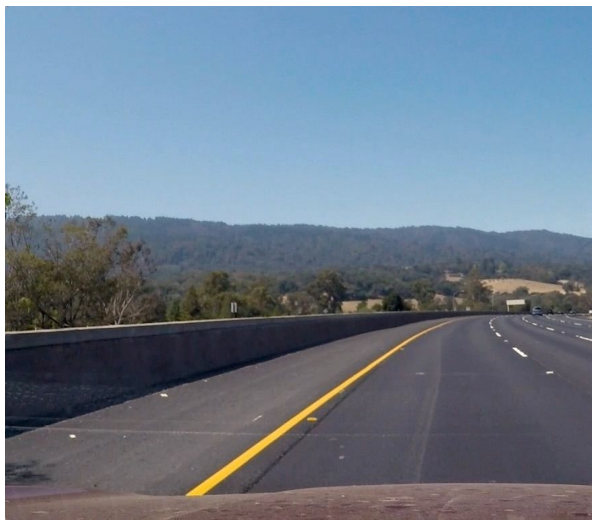
Data Augmentation

We were provided with ~8000 car and non-car images to be used for training and testing. I further used Autti dataset to collect ~10000 car images. Furthermore, I manually collected few **road-signs** and **road images** to be used as non-car. Random crops were generated from these manually collected images. These random crops were also rotated by a randomly chosen angle (-10,10 dregree) to create variations. In total 10000 such random crops were generated to be used as non-car images.

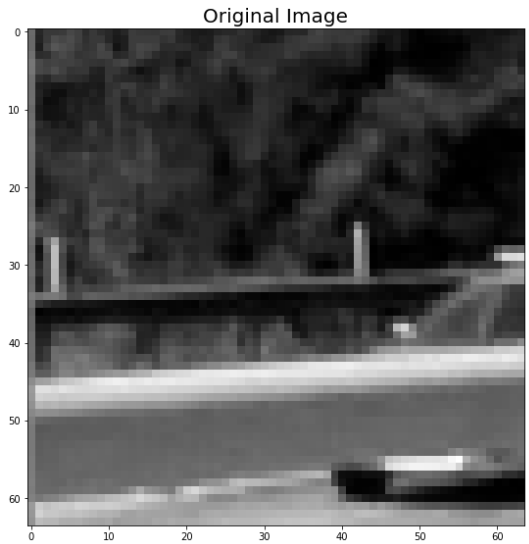


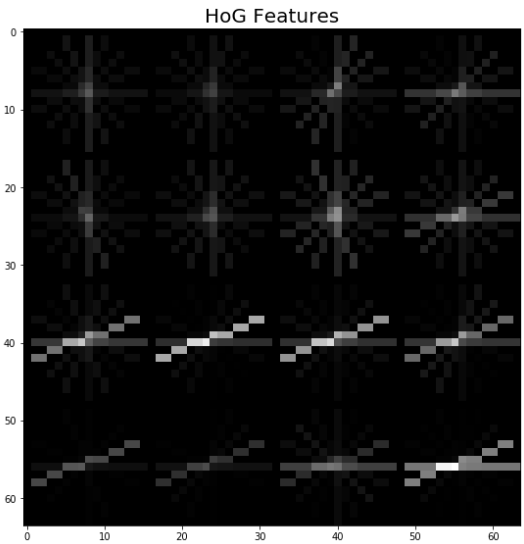

HoG Features

After lots of experimentation, I decided to go with the following HoG parameters:

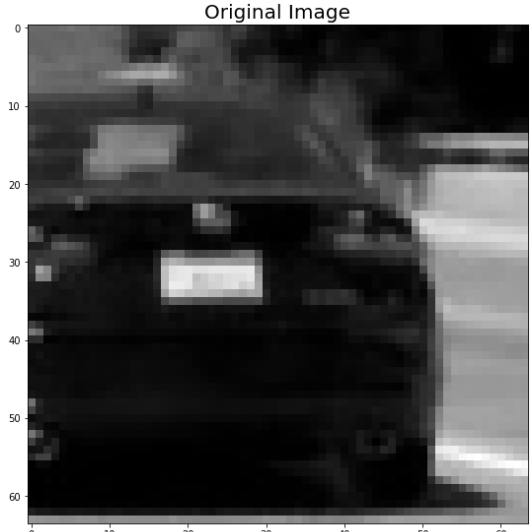
color_space = 'GRAY' # Can be RGB, HSV, LUV, HLS, YUV, YCrCb
orient = 8 # HOG orientations
pix_per_cell = 16 # HOG pixels per cell
cell_per_block = 4 # HOG cells per block
hog_channel = 'ALL' # Can be 0, 1, 2, or "ALL"
spatial_size = (16, 16) # Spatial binning dimensions
hist_bins = 16   # Number of histogram bins
spatial_feat = False # Spatial features on or off
hist_feat = False # Histogram features on or off
hog_feat = True # HOG features on or off

I chose colorspace as Gray, since any 3-channel scheme would try to retain color information of cars. As cars may come in many colors, i don't find it a good idea to use color as basis for identifying cars. Using gray, drastically reduced false positives for me. Hist_Features does not seem to be discriminatory enough to be able to help in car localization. Rest of the HoG parameters were decided experimentally.

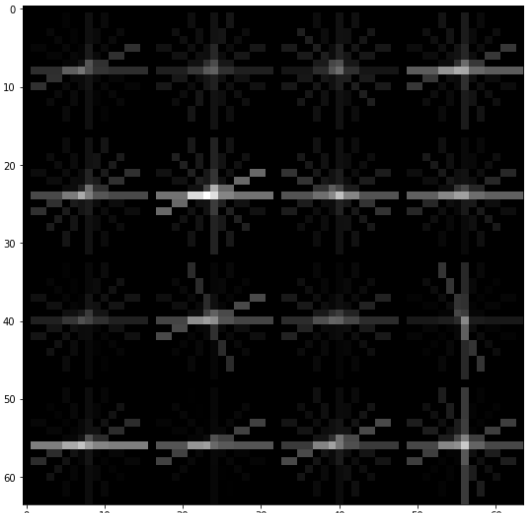




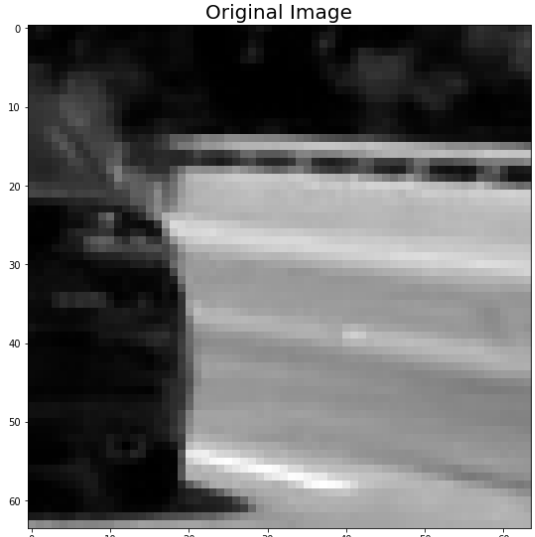
Original Image



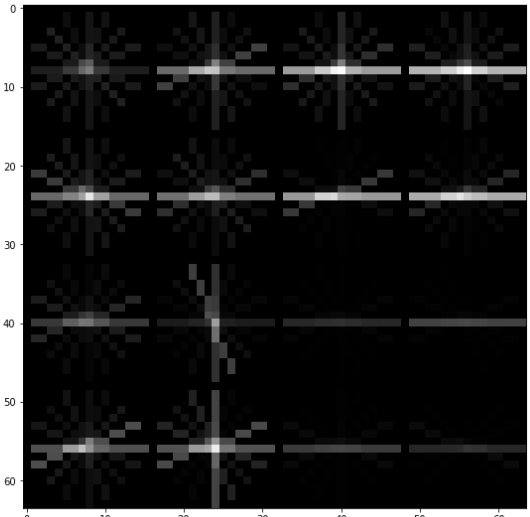
HoG Features



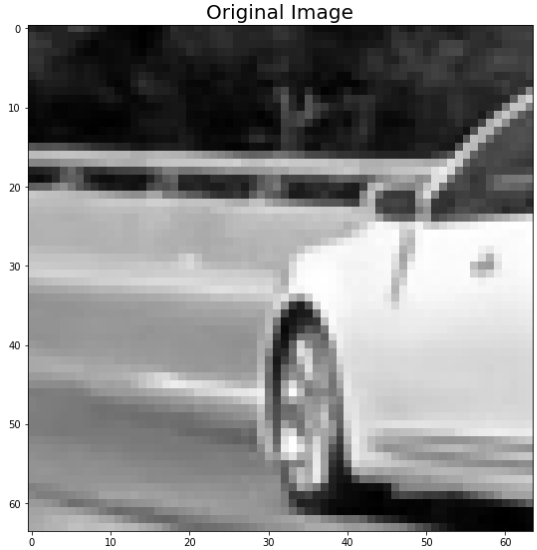
Original Image



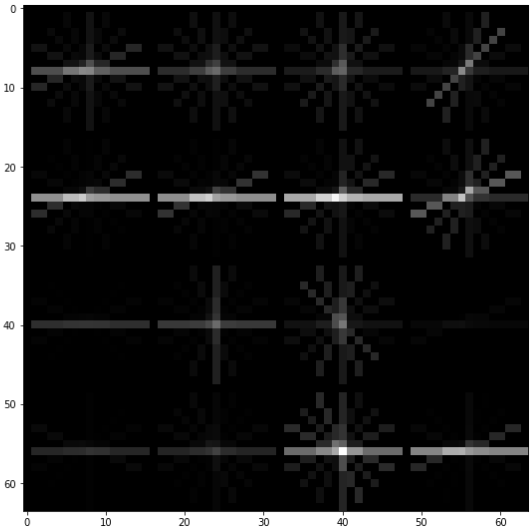
HoG Features

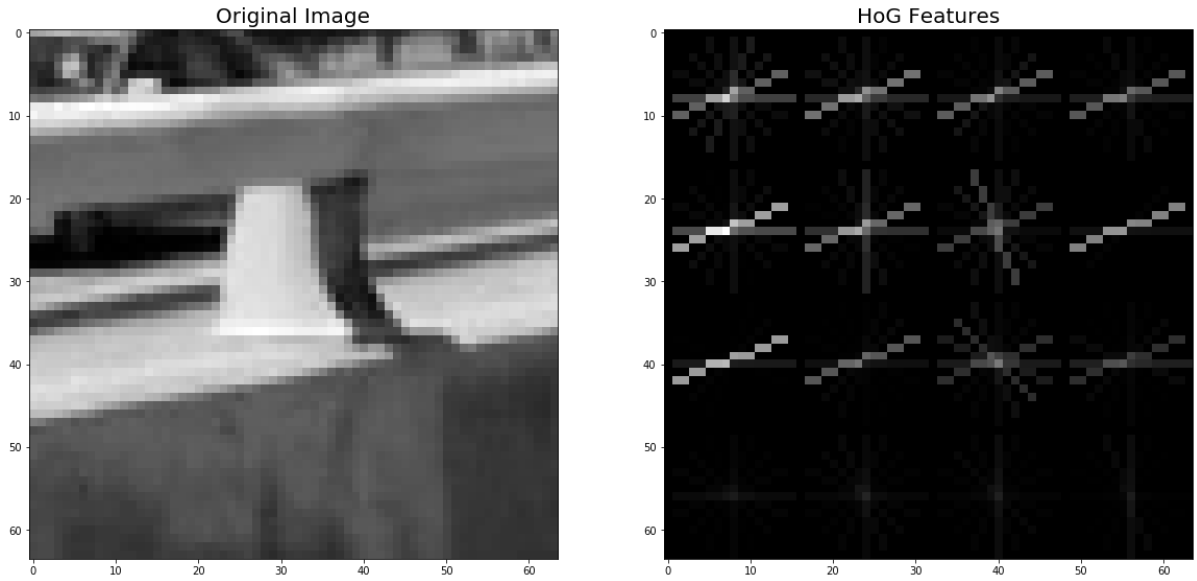


Original Image



HoG Features





Training SVM

The HoG features were scaled using the StandardScalar in sklearn before training SVM. I used *GridSearch* to find the best SVM kernel and its parameters for the task. The best parameters are: RBF kernel with C=10.0. Training with these gave me ~98.3% accuracy over the selected 20% testing dataset. I computed probabilistic output from SVM which helped in further thresholding low confidence values in order to reduce false positive detections.

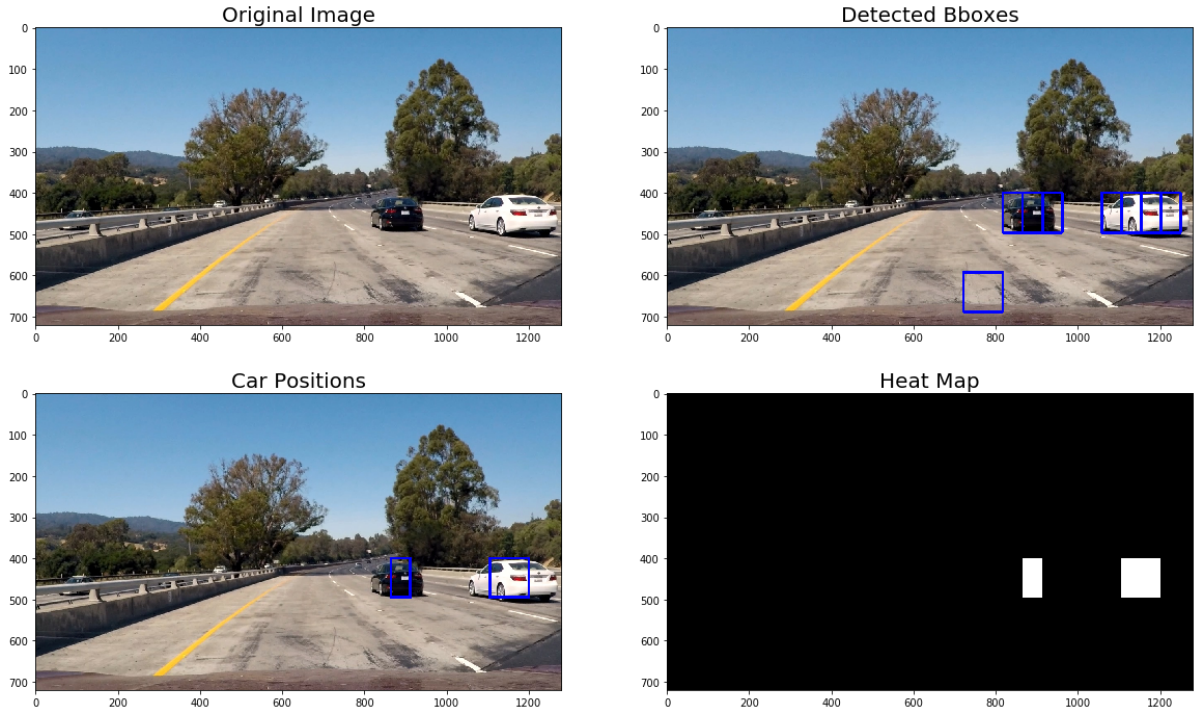
Sliding Window Approach (slide_windows() & search_windows())

A function slide_windows() computes the 50% overlapping windows of 64x64 dimension over the entire image. I searched for windows only in the lower half of the image. I decided not to use multiscale windows just to avoid complexity. The current approach is already quite time intensive and adding multiscale would really slow down the entire pipeline. Also, I was able to get satisfactory result using this above single scale, therefore I decided to skip multiscale approach.

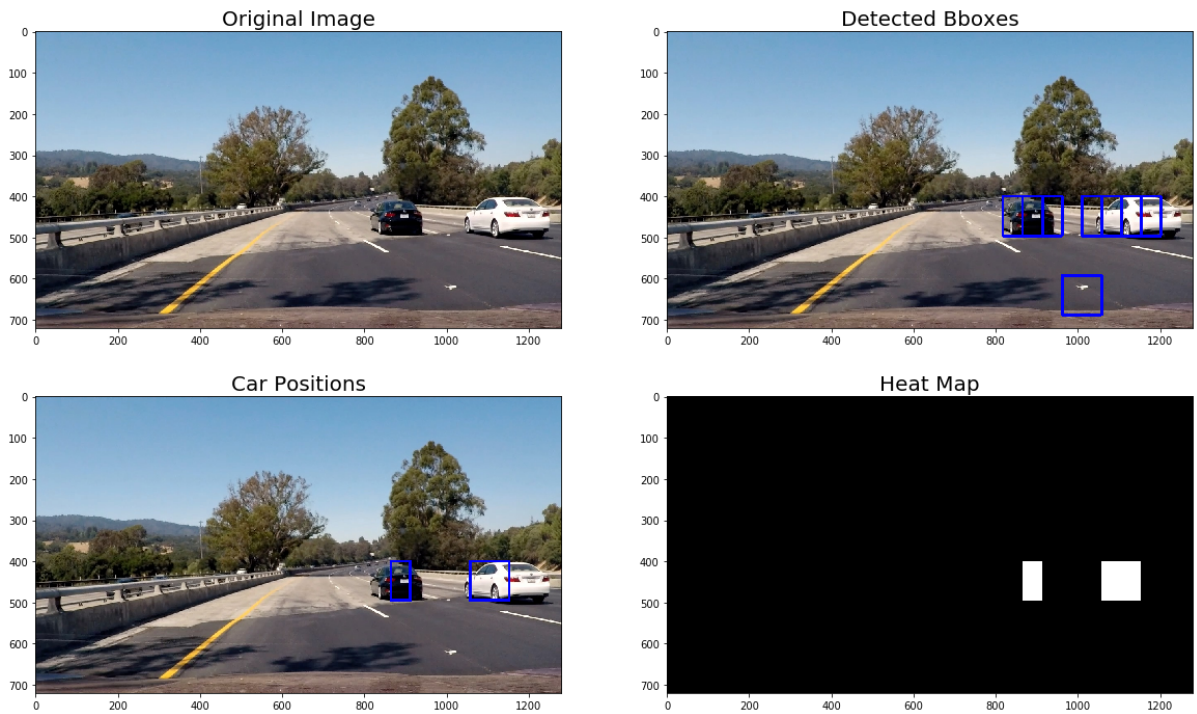
Furthermore, a function search_windows() is applied over each sliding window just decided where a particular window belongs to car category or non-car. The HoG features computed from the window are passed through the trained SVM for a binary decision. If the SVM returns “car” with high enough confidence, we accept it. I used a *temporal-probability-map* in order to further reduce false positives. This is explained in detail later on.

Detected Bounding Boxes

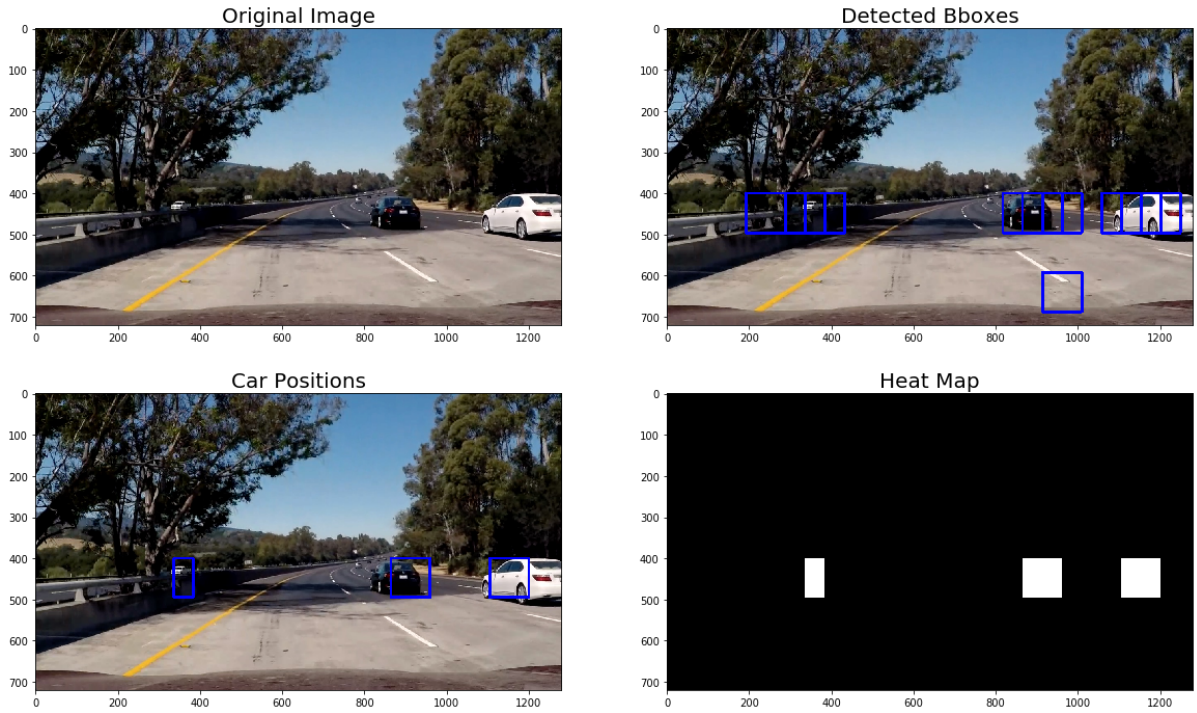
In the images below, we have shown the detected bounding boxes from search_windows(). Afterwards, we generate heatmap by giving 1 vote to each pixel classifier as “car” once. Finally, we accept only the pixels with vote greater than 1 as car. This results in heavy reduction of false positives, at the cost of very few true positives.



Another example:



An example of false detection:



Reducing False Positives

As one can imagine, a high number of false positive was observed initially. Therefore, we employed several strategies to reduce it.

1. Using Heatmap

Each pixel classifier as car by any window was given vote of 1. Finally, we retain only pixels with overall vote greater than 1.

2. Using temporal information to increase or decrease probability of detection of a certain roi, using ProbabilityMap

Each pixel with:
heatmap value>1 was given a confidence of 0.1
0< heatmap value < 1 was given a confidence of 0
heatmap value ==0 was given a confidence of -0.1
This repeated for 3 time steps (i.e. 3 successive frames). Afterwards, the SVM output confidence over the next frame is added to the corresponding confidence obtained from the above ProbabilityMap.

This approach gives higher confidence to areas where a car is detected and lower to ones without a car. There are other ways as well to use the temporal information for even better results.

3. Thresholding the confidence value output by svm in addition to probability of detection as given by ProbabilityMap from last step

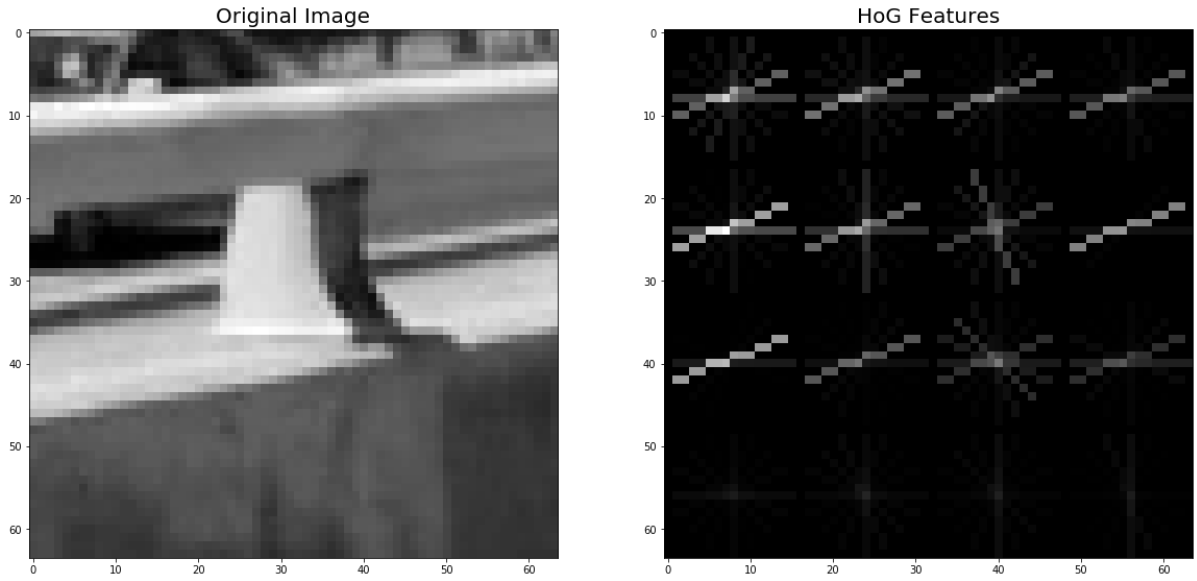
We used a threshold of 0.5 applied over SVM's output + Probability map

4. Hard negative mining

As show in the Data Augmentation section, I manually picked some of the images from wrong results as well in order to further reduce false positives.

Discussion and Possible Improvement

Initially I used only ~8000 samples per class for training SVM and it resulted into too many false positives. Though, it might look like falsely detected rectangles are purely irrelevant to a "car" structure. But,if we zoom in enough we can see that car like edges are present in those false detections. E.g.



The HoG feature of the image above look like side view of the front side of a car. So, I think there is not much more we can get out of HoG in the current set up. We can surely kill bunch of false positives using hard negative mining, which I did. But, this approach might not generalize well to other unseen data samples. Surely, using more data for training helps, and we did that too.

Another approach might be to use different kind of features as well in addition to HoG. One of doing that would be combine other features with HoG and feed to the classifier. Another approach might be build a different classifier using those other features and combine decision of the two classifiers using any technique, e.g. majority voting

If we wish to improvise further, then we should look for features which have better generalization capability that HoG, e.g. convolutional features. In recent research, Faster-RCNN based approaches have worked very well in object localization. We can first try to detect car in a frame and then further use any object-tracking algorithm e.g. Optical

Flow to track that object in future frame. This kind of approach will give more real time detection than the current pipeline.