Vehicle Tracking

Files

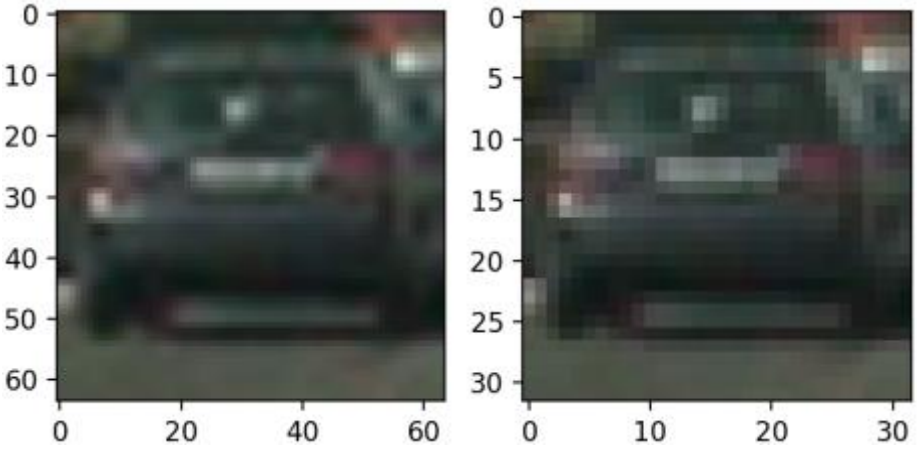
- Process_images.py – functions used to process project videos and single test images
- Feature_utlis.py – functions to extract hog and color features from images
- Window_utils.py – functions to apply sliding window technique, generate heatmap, and draw bounding boxes
- Utils.py – miscellaneous functions to visualize test images and make color conversions
- Train_svc.py – functions to load training images and train linear svc classifier
- Adv_lane_output.mp4 – this pipeline applied to the ouput of the advanced lane lines project
- Project_ouput.mp4 – this pipeline applied to the regular project video

Pipeline

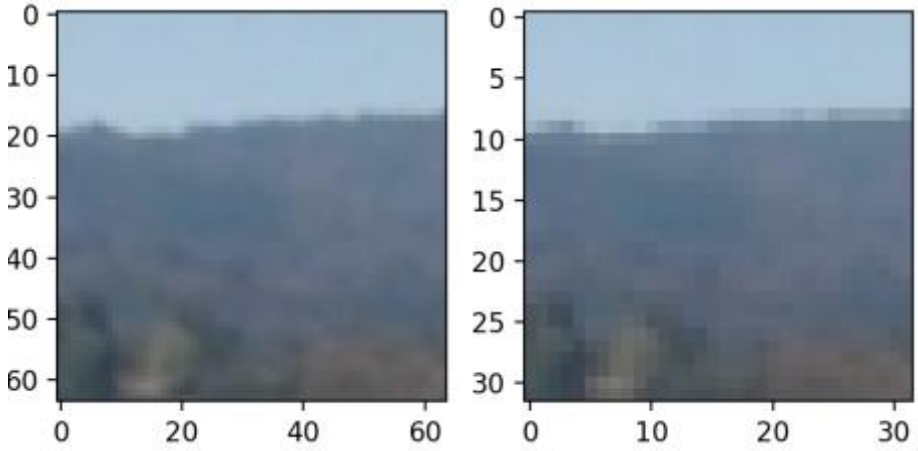
Spatial Binning – feature_utils.py line 27

(32,32) – scales down the image with cv2.resize. I found 32x32 to perform best, any smaller and the project video began to show false positives due to the loss of important image features

Car (original vs spatially binned):



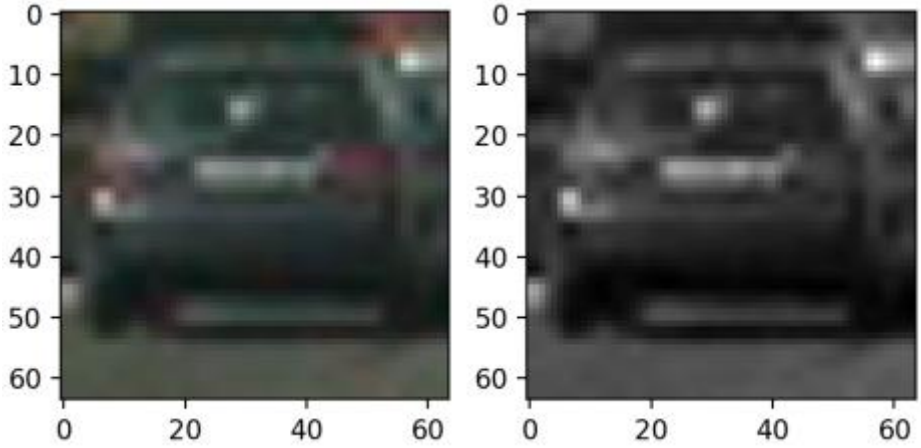
Not Car:



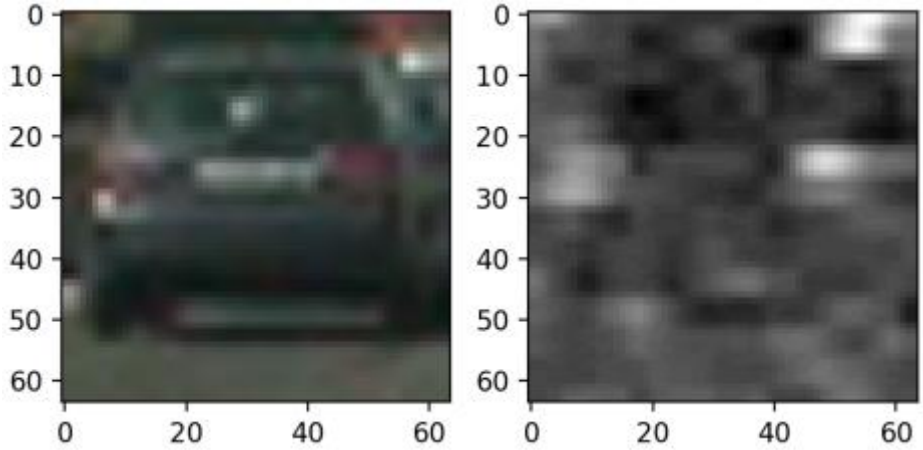
Color histogram - feature_utils.py line 37

YCrCb – yielded the best SVC training accuracy (99.27%) compared to RGB (99.1%) and HSV (99.14%).

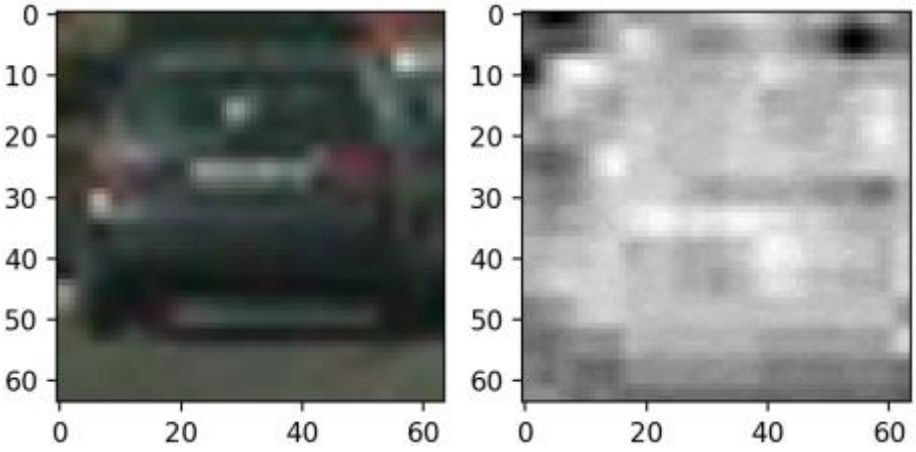
Y Channel



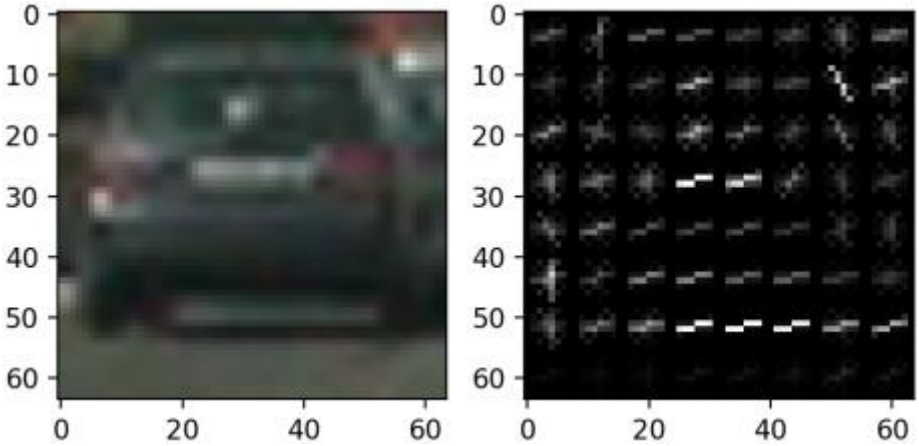
Cr channel



Cb Channel



Hog – feature_utils.py line 13
parameters: All HOG channels, orientations = 9, pixels_per_cell=(8,8), cells_per_block=(2,2). 8x8 pixels per cell was chosen because it is in good proportion to the size of images features of interest (taillights, wheels, rear-windshield)
Car:



Linear SVC Classifier - train_svc.py line 36
Feature vector length 8,460 Test Accuracy of SVC = 99.27%

Processing images

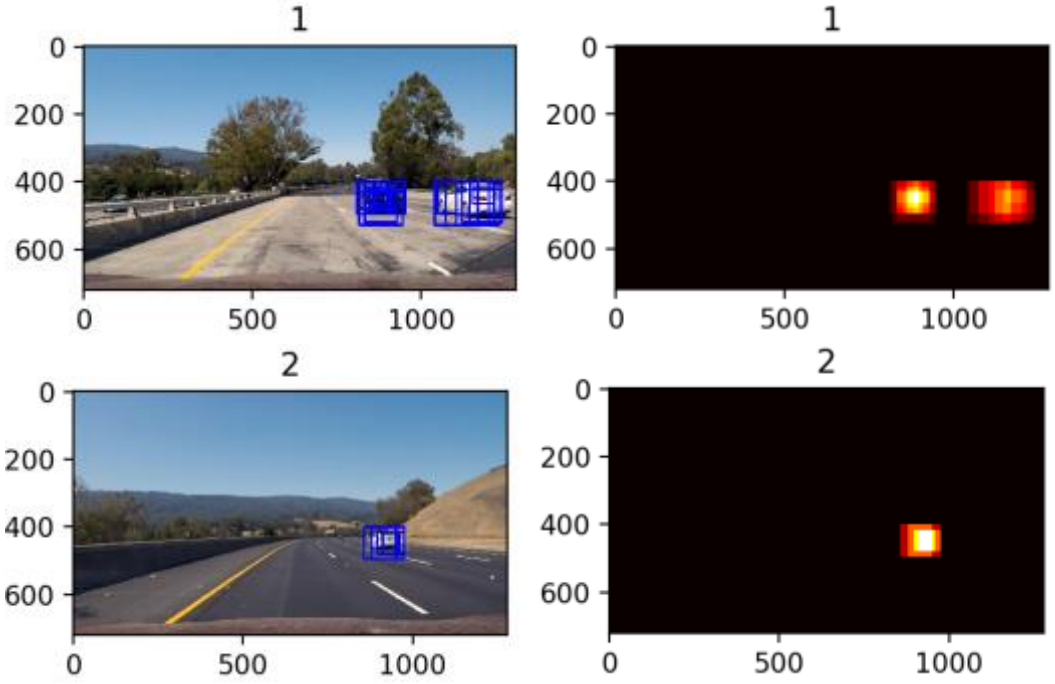
For each image:

- Search region of interest (400-656px y-axis) at 3 different scales, effectively using sliding windows of size 64px, 96px, and 128px. This allows the pipeline to identify cars at different distances from the camera. Because

64px windows are the smallest, and will be looking for smaller cars (i.e. cars towards the horizon) I have limited the 64px window search to between 400-528px on the y-axis

- For each scaled image, take HOG of region of interest one time instead of taking the HOG for each window in our sliding search (Window_utils.py lines 151-153)
- For each window in our search, subsample our HOG array for the current window (Window_utils.py lines 151-153).
- Then apply spatial binning and color histogram features. (Window_utils.py lines 172-173)
- Stack all the features in the same order as we did when training our svc and use the trained svc to make a prediction. If the prediction is positive, draw a rectangle of the current window on the original image, and add "heat" to heatmap. The same heatmap is used for all 3 scaled searches (Window_utils.py lines 180-186)

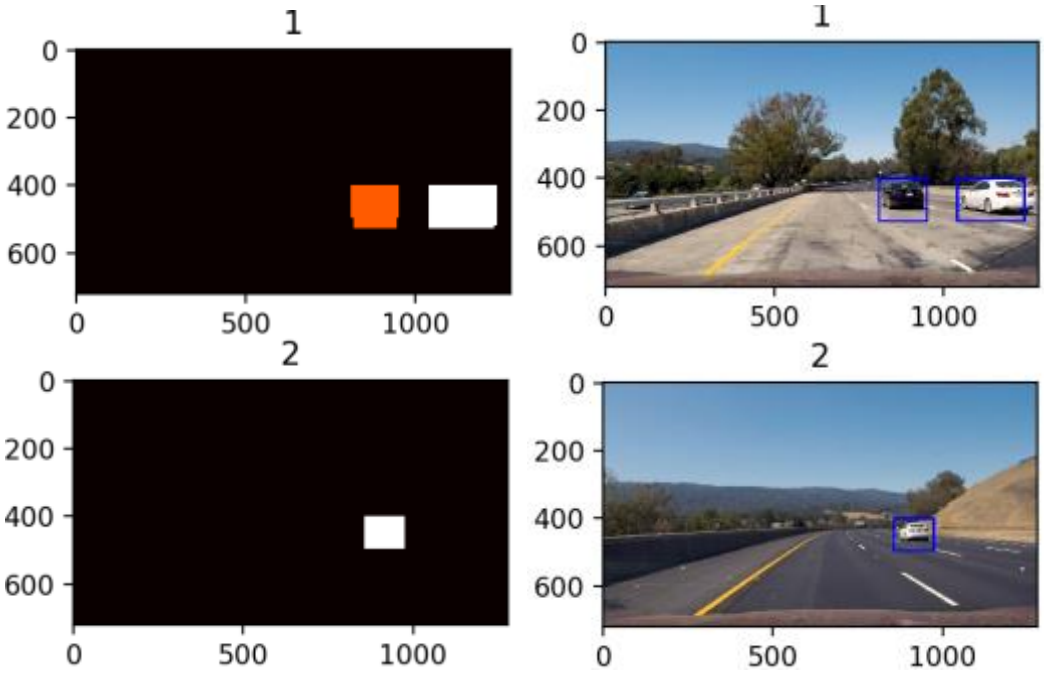
These examples are the results of the sliding window search on all 3 scales (window sizes).



- After applying our sliding window search at all 3 scales, combine the heatmap for the current image with the heatmaps from the previous 8

images. Then apply a heatmap threshold of 6 to the combined heatmap. We then label the heatmap using the label function from scipy.ndimage.measurements and draw bounding boxes for each labeled area (process_images.py lines 145-148)

Result of labeling heatmap using label function from scipy.ndimage.measurements and final result of drawing bounding boxes on labels:



Discussion

A central difficulty of this project was filtering out false positives and smoothing the bounding boxes drawn frame to frame. Combining previous the previous 9 heatmaps allowed me to set a high threshold on my heatmap and effectively took out all of the false positives. It also had the effect of smoothing out the bounding boxes so they weren't as jittery.

This pipeline could be improved by searching more window sizes. This would likely allow the pipeline to get a tighter bounding box drawn around the car, and identify cars further down the road.