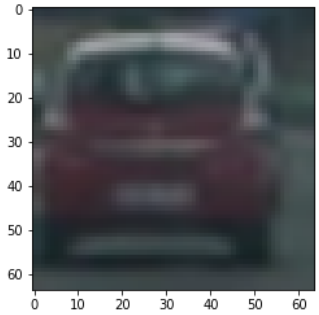
Vehicle Detection Project

Notes: The name of the file is “VehicleDetection2”. I used the procedure recommended by my mentor in order to reduce false positives in the pipeline. https://medium.com/@andreasdaiminger/self-driving-cars-vehicle-detection-e4b95fc82baf

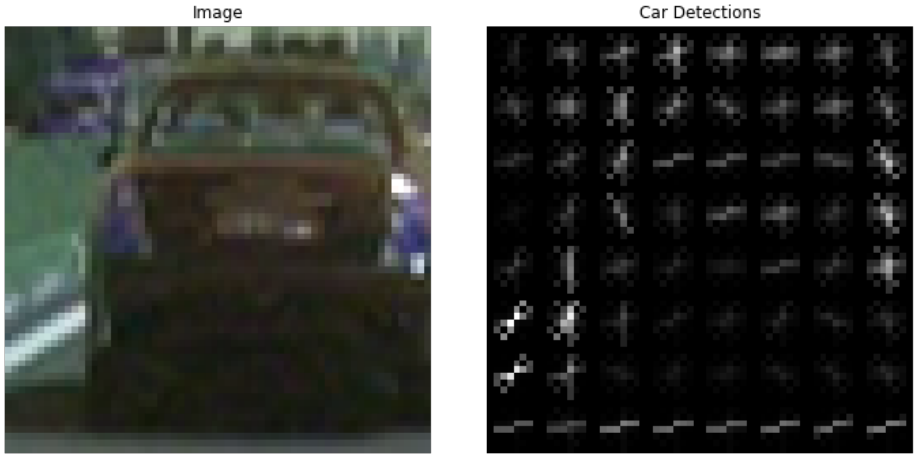
Histogram of Oriented Gradients (HOG)

1. In the section Functions is defined the function to obtain the HOG in the same way as was seen in class. The function is called “get_hog_features”.

I started by reading in all the `vehicle` and `non-vehicle` images. Here is an example of o the `vehicle` class:



In the section “Function Testing” the first part is about obtaining the HOG. Here is an example.



For the last image the next parameters were used.

orient = 9
pix_per_cell = 8
cell_per_block = 2

2. I tried various combinations of parameters and did various test with the svm (LinearSVR). For every test i calculated the accuracy many times to obtain a mean. Finally the best parameters I obtained were:
color_space = 'YCrCb' # Can be RGB, HSV, LUV, HLS, YUV, YCrCb
orient = 10 # HOG orientations
pix_per_cell = 8 # HOG pixels per cell

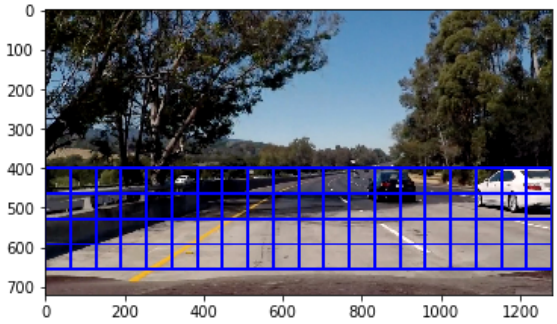
cell_per_block = 2 # HOG cells per block
hog_channel = "ALL" # Can be 0, 1, 2, or "ALL"

3. At the same time I was testing the parameters for HOG I tested the parameter for the Histogram and the spatial binning. After testing many times i chose these ones:
color_space = 'YCrCb' # Can be RGB, HSV, LUV, HLS, YUV, YCrCb
orient = 10 # HOG orientations
pix_per_cell = 8 # HOG pixels per cell
cell_per_block = 2 # HOG cells per block
hog_channel = "ALL" # Can be 0, 1, 2, or "ALL"
spatial_size = (16, 16) # Spatial binning dimensions
hist_bins = 48   # Number of histogram bins
spatial_feat = True # Spatial features on or off
hist_feat = True # Histogram features on or off
hog_feat = True # HOG features on or off

With these parameters I trained a linear SVM in the section SVM. I split the data in test and training data with a 20% of test data. Then i decided to save the values of the SVM and the Scalar of the parameters:
 # Save best clf and scaler
joblib.dump(svc, 'clf.pkl')
joblib.dump(X_scaler, 'scaler.pkl')

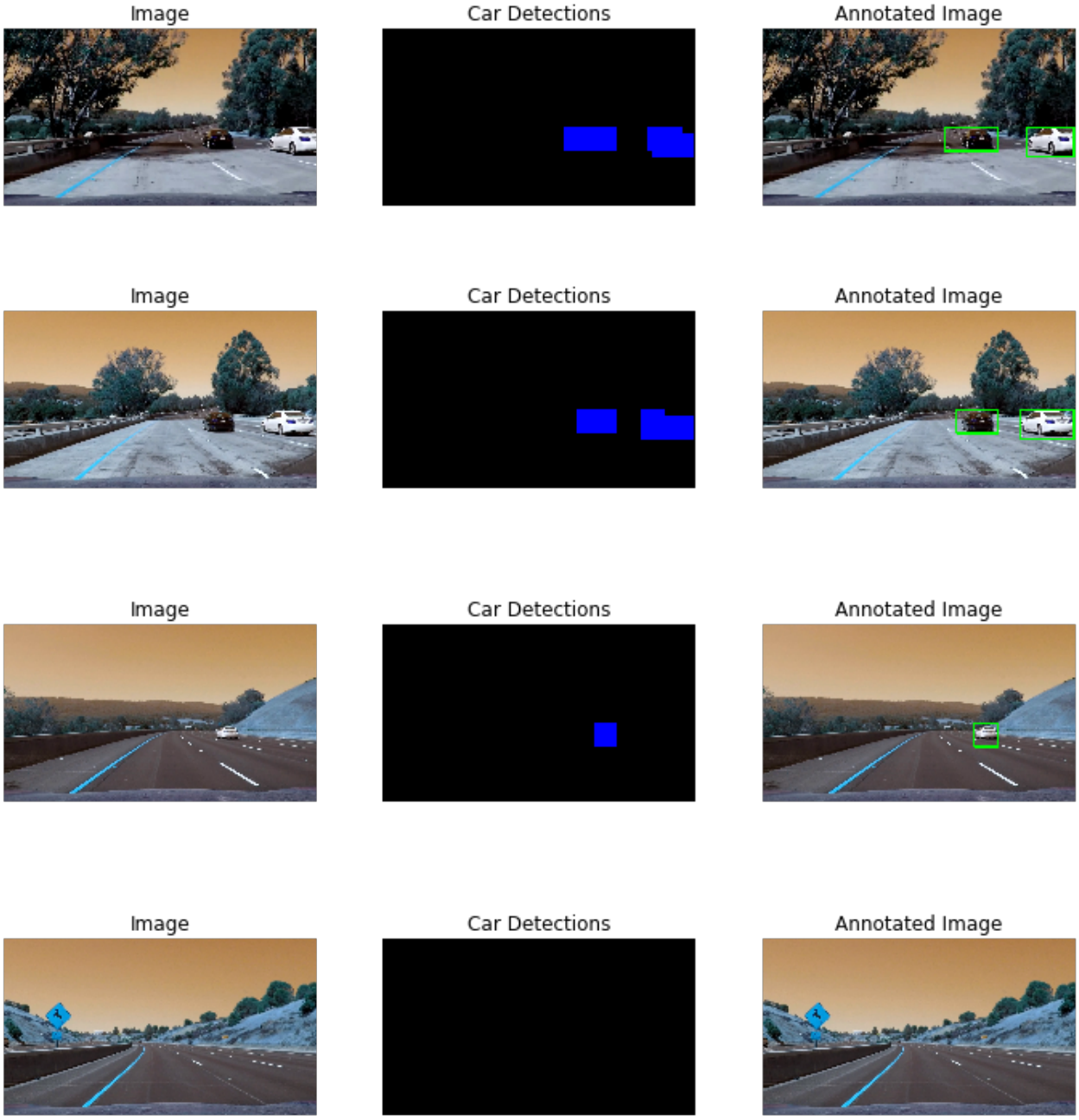
The accuracy was superior of 99%
Sliding Window Search

1. First i tested the sliding windows function in the section “Sliding Window” using an **y_start_stop=[400, 680]**, a windows size of (128,128) and an overlap of (0.5, 0.5). Here is an example.



For sliding window in he pipeline i decided, after many test, to use a combination of windows. At first I use three sizes of windows (64,96,128) but the code was too slow. Finally i decided to use only two **sizes (96,128)**. The best results were with an **overlap** of **(0.75,0.75)**

2. I used the parameters mentioned above to obtain the next Vehicle Detection in the test images, without false positives.



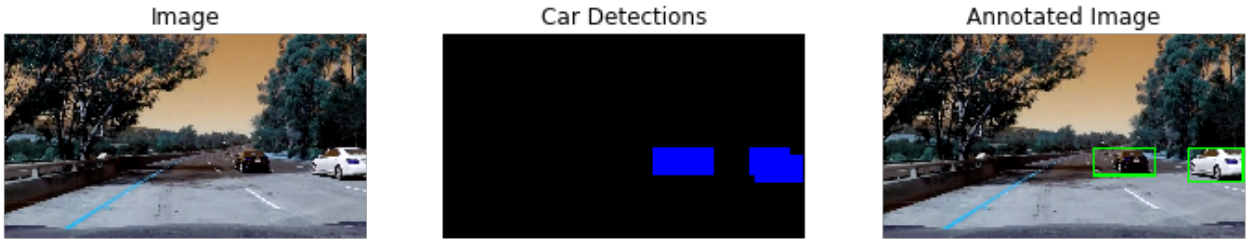
Video Implementation

1. The video is in the zip File

2. As i mentioned before i did a search with two windows sizes (96,128) with an overlap of (0.75,0.75). With the same parameters for the SVM training and testing I predict if a vehicle was in a window or not. Doing the experiments It was obvious that the majority of the false positives were in the windows of small size that is the main reason for choosing windows sizes of (96,128).

When the rectangles from the cars were obtained, in a blank image i drew everyone. Then i obtained the contour of the overlapping of the rectangles. Finally i obtained a bounding rectangle for each contour.

Once in the pipeline in order to reduce False positives i saved the position of the bounding rectangles from the last 30 frames. With this information I grouped the rectangles with the function **cv2.groupRectangle** using the threshold of seven. If the function finds that a group of rectangles appears near each other more than seven times in 30 frames, it means that a car was detected.



Discussion

1. The main problem was the time it took the code to process all the images. For every full test of the video, it took more than three hours. Finally i could obtain a time of one hour reducing the number of windows.

The pipeline is good enough to detect cars with minimum (in the majority of the cases zero) of false positives. But a problem can occur when two cars are near each other, because for a moment one car blocks the other and the pipeline only detects one of them, also when the cars are so close each other the pipeline believes there is only one big car.

I believe in order to make the project more robust it is need to change the method for sliding. Instead of calculating all the HOG features for every window would be better obtain the parameters once and use the values for every window. In the overlapping of the windows it could be the same, a way of obtaining all the parameters once and only do a math operation to obtain the values for every window.