Vehicle Detection and Tracking

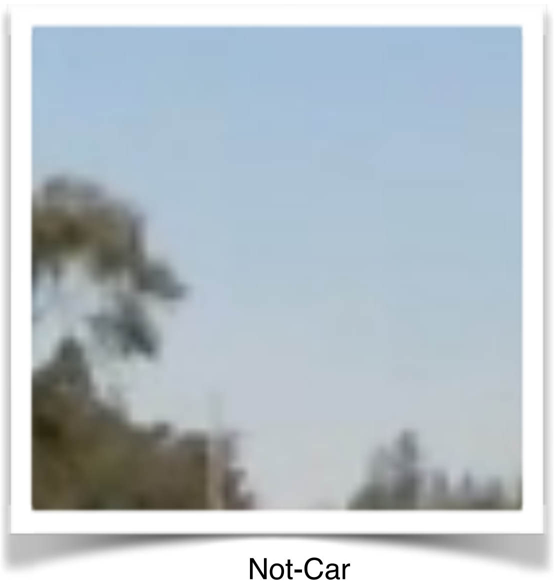
<u>The goals of this project are the following:</u>

- Perform a Histogram of Oriented Gradients (HOG) feature extraction on a labeled training set of images and train a Linear SVM classifier
- Implement a sliding-window technique and use your trained classifier to search for vehicles in images.
- Run this pipeline on a video stream (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.

<u>Histogram of Oriented Gradients (HOG)</u>

The code implementing HOG can be found in lines [16:32]. I have started by creating two lists for 'cars' and 'notcars' and then traversed directories and appended corresponding images to the lists. I then declared the required parameters and set their values. This can be found in lines [356:366]. I have found through trail and error that YCrCb colour space outputs best results. Below is an illustration of car and not-car image.

Car


Not-Car

Through experimentation, the following parameters produced optimal results:

```
color_space = 'YCrCb'
orient = 9  # 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 = (32, 32) # Spatial binning dimensions
hist_bins = 32    # 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
y_start_stop = [400, 700] # Min and max in y to search in slide_window()
```

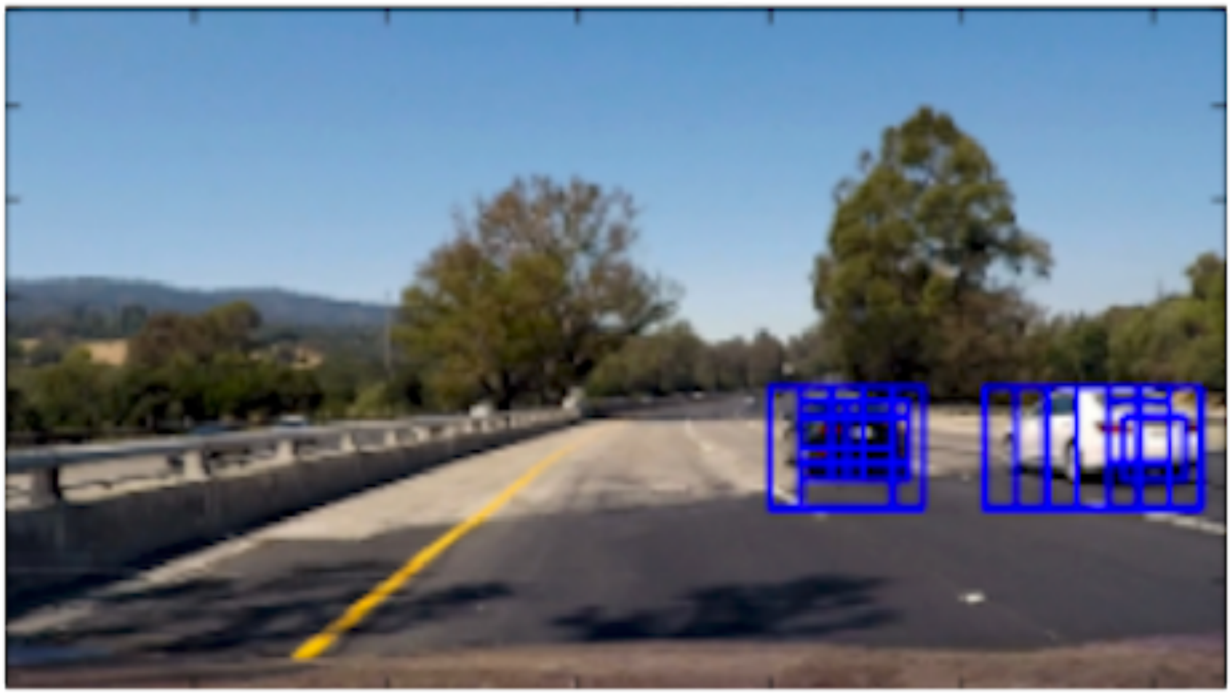
Classifier

The above set of parameters yielded an accuracy of 99% on the test set. I have used a Linear SVM classifier to classify images as cars or not-cars. Before training, I have split my training data randomly into training, validation sets by following a ratio of 70:30. The code for classifier can be found in lines [400:403]. I trained the SVC using the following parameters:

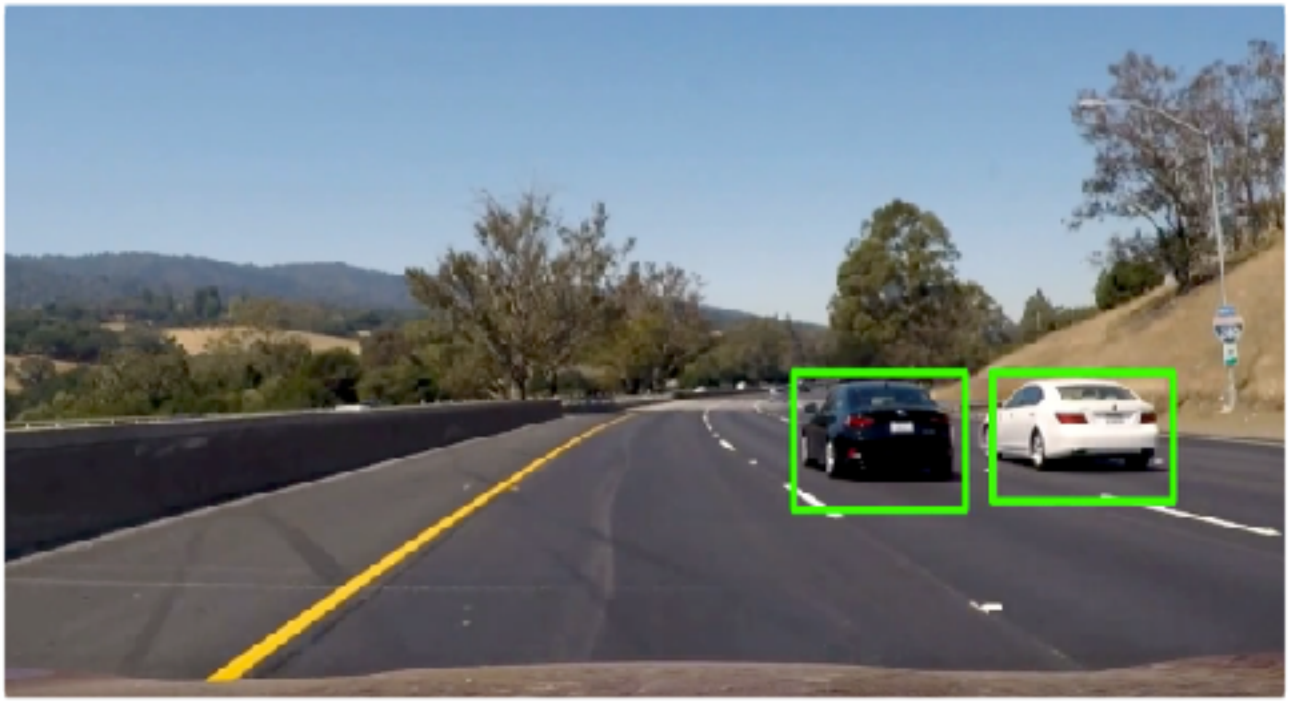
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True, intercept_scaling=1, loss='squared_hinge', max_iter=1000, multi_class='ovr', penalty='l2', random_state=None, tol=0.0001, verbose=0)

Sliding Window Search

This algorithm can be found in lines[114:152]. I have restricted my y axis search space to y[400:700] to speed up the computation. I have called the slide window module 4 times with the corresponding xy_window sizes: (96,96), (150,150), (256,256), (300,300). In all these setting, I have used a overlap of 50% in both 'X' and 'Y' axes. Multi-scale windows has allowed me to capture and bound cars of varying size. Below is an example of an image depicting the different boxes produced by the sliding search window

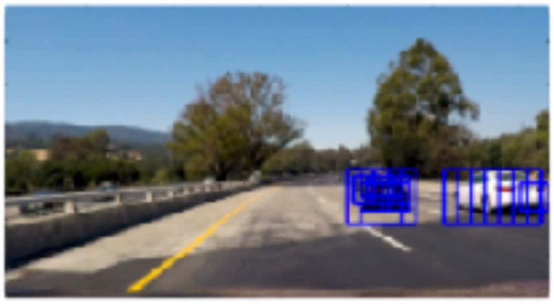


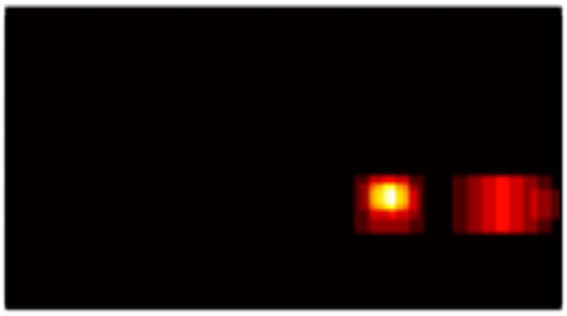
To avoid false positives and multiple boxes on the same object, I have used heatmaps to average the position of the bounding box. I searched on two scales using YCrCb 3-channel HOG features plus spatially binned colour and histograms of colour in the feature vector, which provided a good result. I have used a combination of sliding window technique and HOG sub-sampling window search to obtain optimal results. This can be found in lines [416:437]. Below are few examples of how my pipeline works on images.



Video Implementation

After gaining the bounding boxes for cars, an initial plot revealed overlapping boxes and many false positives. I have handled this by employing heat maps. Heat maps reveal the position of repeated detections. The implementation for this can be found in lines[439:443]. Below is a sample representation of an image and it's correspondence heat map.





Discussion

This pipeline fails to detect cars when they are in certain perspective. This is mainly because, training data does not contain such images. Also, this pipeline doesn’t work in realtime and consumes considerable amount of time to produce the result. These constraints can be possibly overcome by augmenting the dataset by adding flipped images, introducing random brightness/shadows, adding more images into cars and non-cars lists. Deep learning models can be employed when there is more data and this will more likely increase the performance and efficiency of the pipeline.