Vehicle detection

Classifier

I used the vehicle/non vehicle images data base from the Udacity project page link.
I used hog features, spatial binning and color histograms as the feature vector.

Initially, I used the the gp_minimize function from sklearn to zero-in on good set of parameters for bins,orient, pixel_per_cell and cell_per_block. However, the results Were not very promising. Using the best results obtained from gp_minimize, I hand tuned the parameters to get the best verification accuracy.

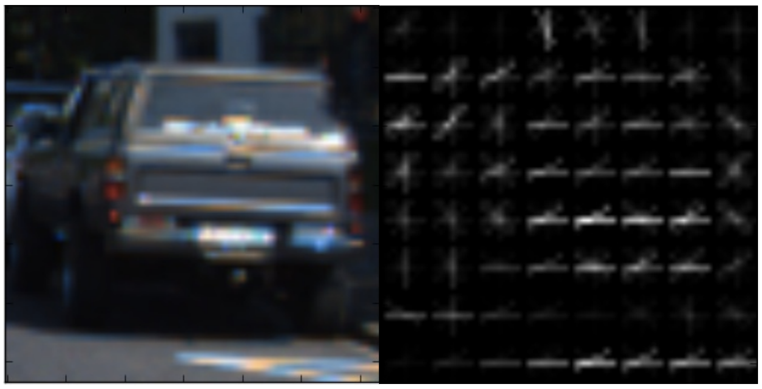
I plotted the 3D plot of car and non car images for RGB, LUV and HSV and found that LUV represented the color saturation of car features in a manner that was most separable. The classifier performed the best when the color space was LUV. Cell block [5] to Cell [7] in the jupyter notebook.

Below are the parameters that I chosen

```
color_space='LUV'
hog_channel = 0
orient = 8
pix_per_cell = 8
cell_per_block = 2
hog_channel = 0 #0,1,22,ALL
spatial_size = (16, 16) # spatial bin dimensions
hist_bins = 32
```

With this set of parameters, the linear SVM classifier on the car/non car data base gives a 98.8 % accuracy. Cell [8] in the jupyter notebook

Example hog feature for a car image is shown below



Sliding windows

Sliding window was used to slide across the image in block in search of a window that might contain a car. The selected window size is resized to a 64x64 image and given to the feature extractor, which returns the features. It was found that if the window size is not comparable to the car image, ie for example, too large or too small, the car detection would fail.

I made sure to limit the window search to exclude the sky portion and only use a limited ROI.

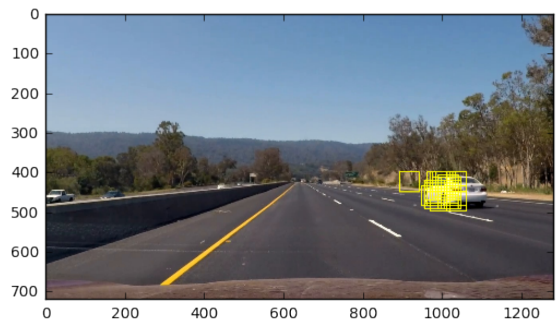
For a single image test of the pipeline, I experimented with 2 sizes of windows. 80x80 window for ystart,ystop of 500,670 of the image which might contain larger car images and 50x50 window to catch smaller car images for cars further away from the present car.

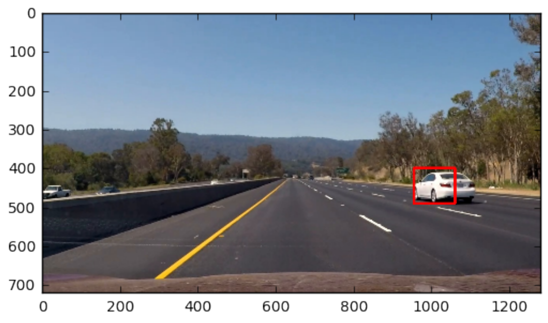
There were a few issues with this design

a) False positives
b) Multiple detection/ overlapping windows for a single car
c) Failed detection

I used a thresholding of number of boxes to overcome the false positives and had a function which is able to take in multiple images to generate a single bounding box.

Results of this can be seen in cell[18]. However, in some cases these static search sizes Were unable to detect cars, as the search box size was not comparable with the car size. Hence we need multi scale windows

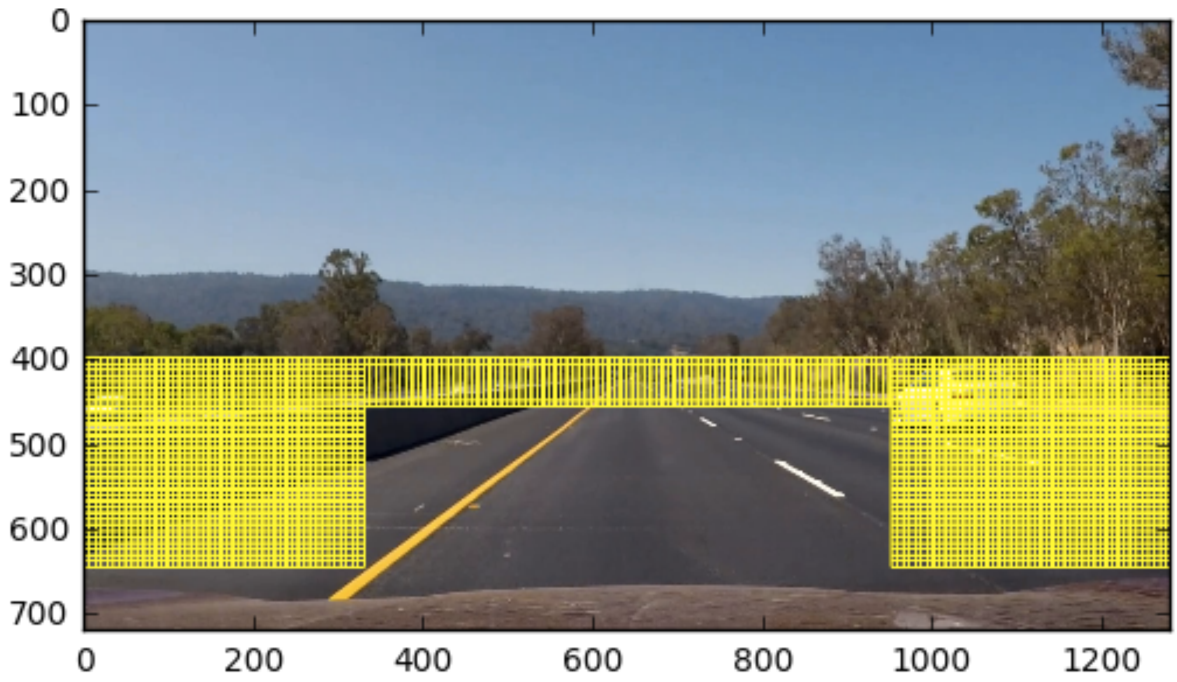




Multi scale windows

Implementation of this uses windows of different scales to scan regions of interest.
I am restricting the window search to 3 regions. Left and right side of the car to look for passing cars and one region to look for cars further down the road. Left and right side of the

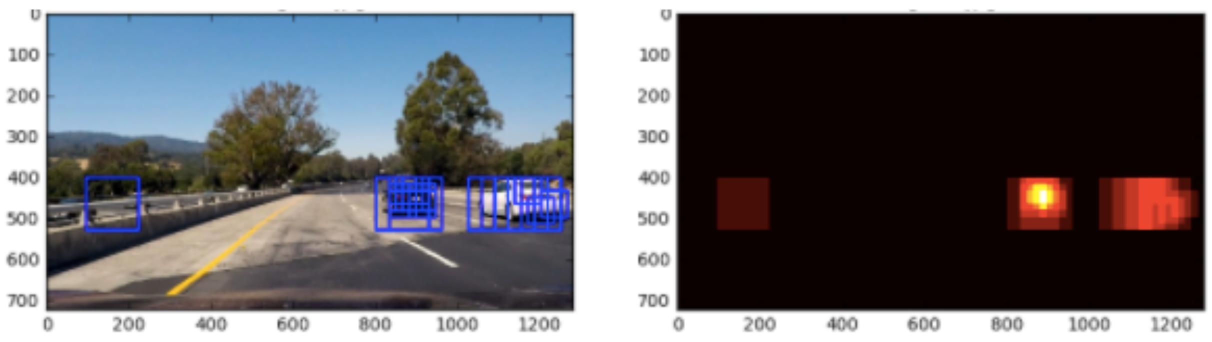
car use scaled up windows, the window further down the road uses a scaled down window to better enable it to look for smaller car images. Cell 20, line 74 to 78

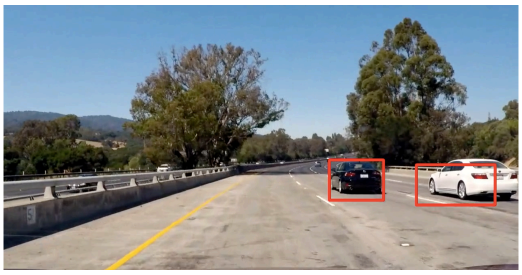


Also, once a region of interest has been identified, I look within and around the region of interest. Cell [80] to cell [102]

To alleviate false detection, data was augmented by flipping the images. Cell 3, line 63
To reduce the false positives, heat map thresholding accumulated over 2 frames was used. Cell 20, line 109
To reduce the jitter, a low pass filtering was introduced. Cell[20], line 13

The below image demonstrates the false positive having a low heat map, while the car images have a stronger heat map





Improvements

- Frame rate improvement can be made by keeping only features which are absolutely necessary
- We could possibly scale down the input image, perform all the necessary extraction and detection and translate the results back onto the original image size
- If the car does not have saturated colors, the classifier may fail
- Classifier may fail under different lighting conditions, eg under a tunner, night time etc . Probably add IR images?