Udacity Self Driving Car Nanodegree
Term 1, Project 5
William J. Keller
May, 2017

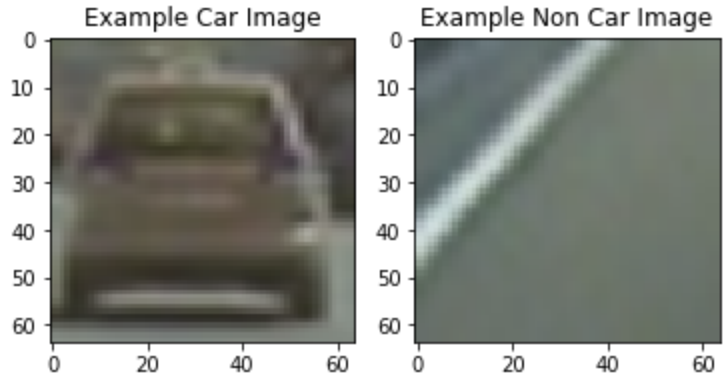
ReadMe

This is the writeup to discuss my fifth project for term one of the Self Driving Car Nanodegree.

Histogram of Oriented Gradients

The code for this section is in the second through fifth cells in my jupyter notebook. I read in the GTI images only - for reasons that are as yet unsolved, I could not get any results with the KITTI images.

Here are examples of a car and a non-car image used for training:



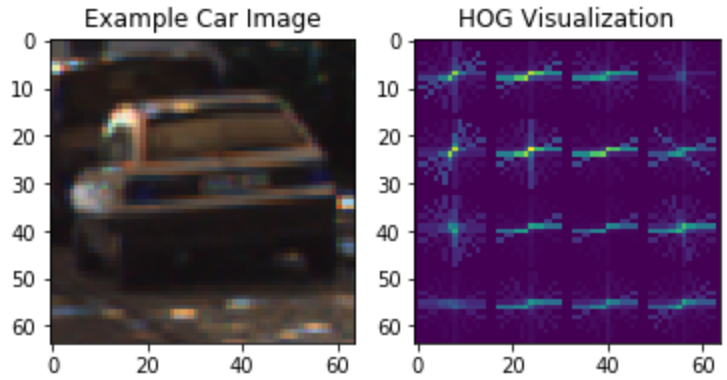
I explored with a plethora of color spaces (all but RBG and BRG) and HOG parameters to find what might work best for this project. I actually took the opportunity offered within the lesson to try to nail this down before I started the project proper, but found it did not seem to translate as well as I had hoped when I started in earnest.

After, in all honesty, weeks of trial and error (and perhaps some lapses in note-taking) I arrived at the combination used in my final version.

```
color_space = 'YCrCb' # Can be RGB, HSV, LUV, HLS, YUV, YCrCb
orient = 11  # HOG orientations
pix_per_cell = 16 # 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
```

Here is an example of an image before and after HOG processing:



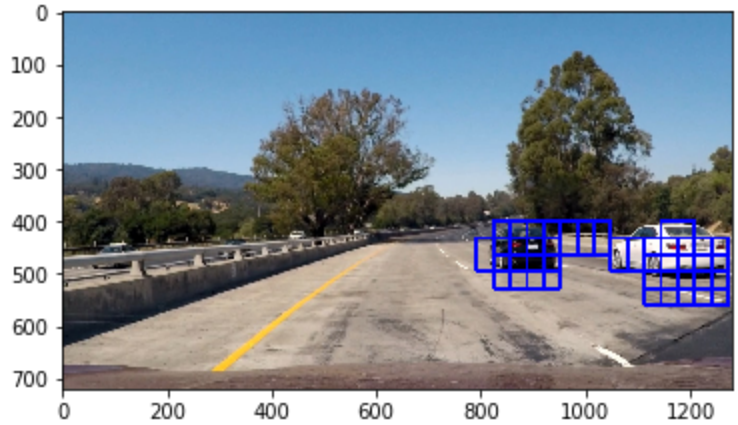
The classifier is trained in the sixth cell of my pipeline. I used a linear SVM with the parameters above. I also used color histograms and spatial binning to classify. I tried the latter two without HOG and vice versa, but found the best results by combining all three techniques. Is wasn't until I began experimenting with the C value for the SVM that I got results close to what I needed. I tried moving it up to 2.0 (from the default 1.0), and then down to 0.3, which is where I left it in the final version.

Sliding Window Search

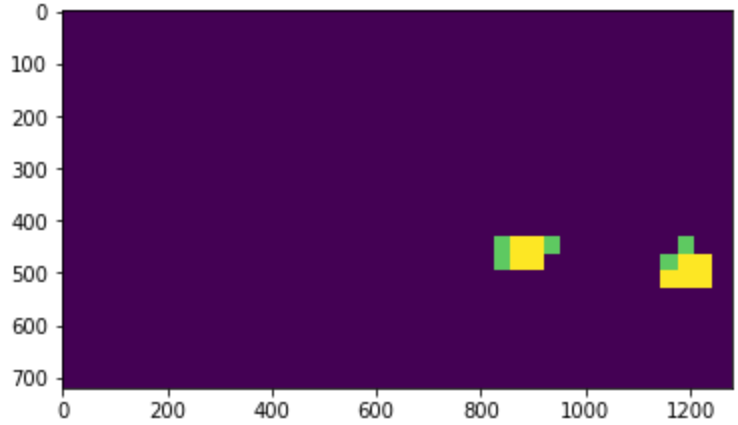
The sliding window search functions are back up in the third cell, but called from the ninth. I kept this pretty standard - using the code from the lesson with minimal changes. I did narrow the search area to a band in the middle of the video vertically, and toward the right side horizontally to eliminate the oncoming lanes. In a real-world application, that would need to be handled differently, but for this test video of a divided highway it works.

Here is a test image pre-processing:

Then the same image after going through the sliding window processing; you can see where the two vehicles are detected:



Here is the same test image, but with the heatmap applied:



I used a deque with a length of 10 to keep a running tally of located heatmaps, averaged the tally for each frame (for the last 10 frames), and then applied a threshold of 2 to the average to weed out most of the false positives and smooth the results a bit.

Lastly, I added a few lines to make sure the windows were sized reasonably and centered on the vehicles.

Video

The resultant video is included in the github repository for my project.

Discussion

This one was frustrating - I felt I was moving in circles on the parameters, and I still don't know why the KITTI images would not work for me. I think more data for training might make the classifier a bit more robust. I also had my laptop time out if I tried too few pixels per cell or cells per block - not sure if that was a limiting factor too.