CarND - Vehicle Detection

By James Marshall

HOG Feature Extraction

For my HOG feature extraction I settled on my parameters as follows:

-Colourspace: YUV
-Orient: 11
-Pixels per cell: 16
-Cells per block: 2
-HOG channels: All

I settled on these parameters after reading both an article written on exploring HOG features using this project as an example (linked here) and also looking at the sample implementation of Jeremy Shannon (linked here). After sampling some basic parameters myself, namely parameters similar to the lessons, with just slight changes to pixels per cell and cells per block, I wanted to find some parameters that would give me very solid results. Therefore, I tested the top 2 test accuracies from the medium post, and Jeremy's chosen parameters, as show in code cells 8-13. Unfortunately, with the parameters shown in the medium post, I was not able to achieve the same accuracies as they did. The highest accuracy I achieved was 98.38% (shown in code cell 9), using Jeremy's parameters and using a linear SVC classifier.

Sliding Window Search and Video Implementation

For my sliding window search I followed the code from the lessons quite closely, and initially tried two sets of windows for each scale, ranging from 1.0, to 3.5. That left me with 12 sets of sliding windows going across my images, which can be seen in code cells 18-23. However, when I was using all of these sets of windows, I was getting a very large amounts of false positives. These false positives were all quite large windows, so initially, I got rid of all sliding window sets that had a scale greater than that of 2.0. When I got to testing on the video however, it took a while to detect the cars, as the remaining sliding windows were too small to correctly classify cars that were close to the camera. To get around this problem, I reintroduced the sliding window set with a scale of 3.5, which found the cars a lot earlier and didn't bring back the false positives. Upon further investigation, the scales of 3.0 and 2.5 seemed to bring back the false positives, so those sets remained unused.
The other thing I implemented in an effort to eliminate false positives as recommended was to use heatmaps with thresholds. For the test images, I made the threshold 1, which worked for the meantime, but didn't work when it came to the video stream. For the purposes of this project, I knew I had to identify 2 cars, so I set the threshold to the

number of rectangles in the image plus 1. This worked well enough to eliminate almost all false positives, however it meant that in one part of the video, the white car doesn't get detected as the other car comes into frame. All coding for the heat maps are in code cells 24-28, with the final threshold shown at the bottom of code cell 35.
There are two image pipelines in my code (cells 30 and 35), one which is just used for single images (30) and the other, which also uses a detection class that tracks previously found rectangles, that is used for the video stream. This class, written in code cell 34, is needed as using the pipeline in cell 30 doesn't detect vehicles in every frame, so keeping track of previous detections is necessary to have a smoother detection output.

Discussion

The major issues I faced in this project was getting good enough parameters for my HOG feature extraction, as well as finding sufficient parameters for both the sliding window searches and heatmap thresholds. I was able to get satisfactory results for the sliding window and heatmap parameters through trial and error, but to get HOG parameters to a higher standard I did some sourcing and found the medium article as well as the sample implementation.
However, these parameters could still be improved, as there are still instances in the video stream where the detection isn't perfect, sometimes with the rectangles falling into the back half of the vehicle, and at one point, one vehicle not being detected for about a second.
With my current implementation, a scenario where there are many cars to be detected would not work, as well as scenarios where cars are at the very side of the camera image, aren't detected, which is very important. With the tracking of previous rectangles, a car that moves across the image a large distance in a small amount of time, ie a fast lane change, could also break my pipeline as I'm quite sure the heatmap would fall under the threshold and would not be detected.