Sebastian Nietardt

Udacity Self-Driving Car Nanodegree

Project 5

April 18, 2017

Vehicle Detection and Tracking

Histogram of Oriented Gradients (HOG)

Explain how (and identify where in your code) you extracted HOG features from the training images. Explain how you settled on your final choice of HOG parameters.

The code for the HOG feature extraction is located in the fourth cell of the jupyter notebook. It is called from the generateImageFeatures() function which is located in the tenth cell.

The first step was loading the vehicle and non-vehicle images and counting them. The number of vehicle examples and non-vehicle examples was quite balanced and the size of all the images was 64x64 pixels. That means the dataset was ready for usage.

After that some basic functions have been implemented. The function to extract the HOG features (from a single-color channel) is called from the function generateImageFeatures() which calls the HOG function for each single color channel. The generateImageFeatures() function got extended by histogram and spatial feature extraction functions later.

The data has been loaded, scaled, shuffled and split into a training and a testing set several times with different feature extractions, which came from several color spaces. The linear SVM Classifier has been applied to all the datasets using cross validation. The color space HLS in combination with the gray scaled color channel showed the best accuracy score using hog feature extraction. That why the three HLS channels and the gray scaled channel have been chosen for the final model.

Describe how (and identify where in your code) you trained a classifier using your selected HOG features (and color features if you used them).

The training for the classifier happens in the fourteenth to sixteens cell in the jupyter notebook. The fourteenth cell builds the classifier and does a cross validation in order to tune the C parameter. The fifteenth cell displays the best parameter and the sixteenth cell trains the classifier with the best parameter again.

Before the training started, the fractures needed to be extracted first. The generateImageFeatures() function extracts the HLS hog features, the gray scaled hog features, a histogram of color features and the spatial features of a 16x16 pixel image. These features have been concatenated in a single feature vector. In the end, the total number of extracted features were 8592 per image.

After the linear SVM classifier was trained on these features, it got an accuracy score higher than 0.99 in the testing set.

Sliding Window Search

Describe how (and identify where in your code) you implemented a sliding window search. How did you decide what scales to search and how much to overlap windows?

The code for the sliding window search starts in the 23rd cell of the jupyter notebook. The sliding windows are calculated in the 19th cell.

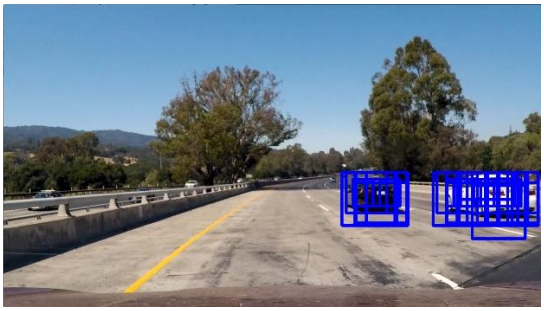
Several windows sized slide over the images. A 64-pixel square window sliding from an y value of 400 to an y value of 480, a 96-pixel square window sliding from an y value of 400 to an y value of 526, a 126-pixel square window sliding from an y value of 400 to an y value of 558 and a 400-pixel square window sliding from an y value of 400 to an y value of 656. All the windows slide with an overlap of 0.75 in the x as well as the y direction.

The values for these sliding windows have been measured on the test images. The smaller sliding windows just search in higher regions of the images. The larger the sliding window gets, the lower will it look for vehicles. The following picture shows the total area over which the 64-pixel square windows slide.

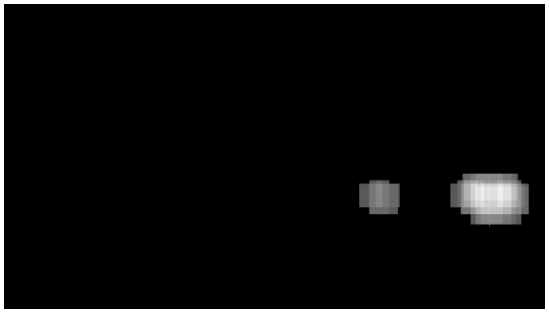
Show some examples of test images to demonstrate how your pipeline is working. How did you optimize the performance of your classifier?

The classifier does a good job identifying cars, but it does detect false positives from time to time. The classifier has been optimized by adding spatial binned and histogram binned features as well as the gray scaled color channel hog features. In addition, it was optimized using cross validation which tuned the parameter C. The last step was defining the sliding windows size, the sliding area and how much the sliding windows should overlap.

Sliding windows which detected cars:



Heatmap of the same image:



Sliding windows which detected cars (one false positive):

Heatmap of the same image:

Video Implementation

Provide a link to your final video output. Your pipeline should perform reasonably well on the entire project video (somewhat wobbly or unstable bounding boxes are ok as long as you are identifying the vehicles most of the time with minimal false positives.)

Link to the video

Describe how (and identify where in your code) you implemented some kind of filter for false positives and some method for combining overlapping bounding boxes.

The code to filter the false positive detections is located in the 20th to the 23rd cell in the jupyter notebook. The cells 20 to 22 define the functions which are called from the detectVehicles() function in the 23th cell.

In order to sort out the false positive detected windows, a threshold of minimum overlapping windows has been defined to generate a heatmap. With this heatmap, the position of the cars could be detected and surrounded by a rectangle. In the video, it is important to keep track of the cars by summing up the overlapping detection from former images. A round buffer with five lists has been generated to memorize the sliding windows which detected cars from the last five images. The threshold for a detected car has been set to at least 6 overlapping sliding windows within the last five images. The result can be seen in the video. The two cars are detected almost the entire time of the video with very little false positive detections.

Discussion

Briefly discuss any problems / issues you faced in your implementation of this project. Where will your pipeline likely fail? What could you do to make it more robust?

The first challenge was to find a good color space for the hog feature extraction. I needed to try several color spaces for the hog feature extraction, spatial binned features and histogram features separately as well as in combination. After I got a good combination, the classifier was more than 99% accuracy for making predictions on the test images. The second challenge was to find out how many windows should slide over each image, what size should the windows have and over what region should they slide. This decision must be made to get a good accuracy/speed tradeoff. The third challenge was the threshold for the heatmap to filter out the false positive detections while keeping the cars detected.

I can imagine, that the pipeline will fail in a very hilly road. Since just a particular region is searched for cars, this region need to change on hilly roads. The cars might be higher or lower in the image. To make the pipeline more robust, the search area could be increased. I can also imagen, that a very winding road could bring the pipeline to fail, since the cars are moving left and right quicker and the threshold must be adapted. A bad illumination could also be an issue. Maybe the classifier will not recognize the cars that accurate anymore and additional features must be used to classify the images.