Self Driving Car Nanodegree Project 1

Michael Person

February 17, 2017

1 Pipeline Description

My pipeline consists of 9 parts. I start off by converting the three channel image into a single channel grayscale image. Next I apply a Gaussian blur in order to reduce noise when finding edges. In order to find the edges, I use the Canny edge detector. After the edges in the image have been found, I apply a trapezoidal mask to focus on the ROI of our lane. Next the found edges are run through the Hough transformation in order to filter out the noisy lines which are less likely to be the lane lines themselves. After all the major lines have been found in the image I separate the found lines into those that are on the right half of the road and the ones that are on the left half of the road. After I have separated all the left and right lines that appear in the image I fit a line through both sets of points which represents the lane lines that have been found in the image that span our ROI. Next these found lane lines are annotated onto a blank image and then the blank image and the original image are combined to show where the found lane lines are.

2 Shortcomings

There are some major shortcomings with this pipeline. On of these is that the vertices of the mask are hard coded to a straight lane. This will lead to less precise annotations when the lanes are curved such as seen in the challenge video. The largest drawback to this pipeline is that the lighting must be excellent and the road must be homogeneous and in good condition in order for false detections to not occur. The pipeline works perfectly on the white and yellow lane line sample videos but it completely crashes for the challenge video which is evidence that proper lighting is imperative to this pipeline's success.

One method that was tried to combat the lighting failures was to filter out lines that were found based upon the magnitude of their slope. The logic behind this was that if a line that was found did not have the slop that was in the neighborhood of what a lane in front of you should have then it is probably noise and can be discarded. However either due to an implementation fault or faulty logic, this method required being overly aggressive in discarding lanes and ended up giving worse annotations than the without on all three sample videos.

Another method that was implemented but abandoned was moving the image processing into a different color space than grayscale or RGB. I believed that maybe there was a nonlinear transform that would be able to more readily distinguish what were lanes and what was the road but this had little effect.

3 Possible Improvements

On the forums I saw that the most likely option to improve the pipeline was to move the processing into a different color space. I believe that however simply moving to a different space is not enough and a different approach would be needed entirely such as processing all three channels independently and finding a way of combining them back again by filtering out lines that were not common between all three channels.

I also know that very robust and accurate lane finding systems can be implemented with CNNs as was demonstrated by DeepLanes from the research group at Ford however that is beyond the scope of the course at this point in time.