Finding Lane Lines on the Road

Finding Lane Lines on the Road

The goals / steps of this project are the following:

- Make a pipeline that finds lane lines on the road
- Reflect on your work in a written report

Reflection

0. Second project submission.

This is my second submission of this project. The first one suffered from shaky lines (videos) and lines that were not over the lane lines (images) especially on dashed lane lines. I made the following changes which reduced these errors and a couple sample pictures follow and are labeled “pre-2nd submission”:

* Adjusted the Guassian parameter:
    Increased kernel size from 5 to 9 (although I'm not sure this made a difference)
* Tried different Canny low & high thresholds but didn't notice a difference so left them unchanged.
* Changed the Hough parameters (this made the largest difference):
    threshold from 1 to 35
    min_line_length from to 5
    max_line_gap from 1 to 2
* Removed vertical slopes when calculating slope averages
* Adjusted the vertices enclosing the analyzed region to be based on the image size rather than on fixed values.

After the first set of updates, the lines still weren’t following the lane lines as closely as the sample images and video and there was still the occasional wild line in a frame of a video so I made the following changes to address these:

* Calculated a center point for each line and used these to calculate the line formula instead of the edge y that I had been using previously.
* Set a threshold on acceptable slopes for each side such that slopes close to 0 are discarded.

Figure 1 - 1st submission: solidWhiteCurve.jpg

Figure 2 – pre-2nd submission: solidWhiteCurve.jpg

Figure 3 - 2nd submission: solidWhiteCurve.jpg

Figure 4 - 1st submission: solidYellowLeft.jpg

Figure 5 – pre-2nd submission: solidYellowLeft.jpg

Figure 6 - 2nd submission: solidYellowLeft.jpg

1. Describe your pipeline. As part of the description, explain how you modified the draw_lines() function.

My pipeline consisted of 5 steps. First, I converted the images to grayscale, then I ran the Gaussian blur with a specified kernel size. Next I ran the Canny algorithm specifying low and high thresholds. I next set a region to analyze. Next I ran the Hough algorithm to generate lines and then applied those lines to the original image.

In order to draw a single line on the left and right lanes, I modified the draw_lines() function. I cycled through the lines generated by the Hough algorithm calculating the slope on each in order to separate then into segments of the left lane line and segments of the right lane line. I averaged the slopes in each of the left and right sides to generate one slope for each side. I used the slope and a point from each side to generate the single lines for the right and left sides.

Here are a couple of my images.

Figure 7 line segments shown

Figure 8 full lines shown

The remaining images and the videos are in the zip file under test_results.

Figure 9 - 1st submission: solidYellowLeft.jpg

2. Identify potential shortcomings with your current pipeline

An issue I ran into is determining an accurate slope. A more robust method would generate more stable lines. I attempted using stats.linregress from scipy but had trouble figuring out how to use it.

Another limit to my pipeline is that the analyzed region is pre-specified so that when the road curves to quickly, it might go out of scope forcing inaccurate calculations.

Yet one more limitation is the hard-coded parameters. Since they are tuned during execution, environment changes such as lighting or fading lines could interfere with the calculations.

3. Suggest possible improvements to your pipeline

One improvement is to use a method such as stats.linregress to generate a more accurate slope.

Another improvement is to adjust the parameters dynamically.