Finding Lane Lines on the Road

Reflection

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

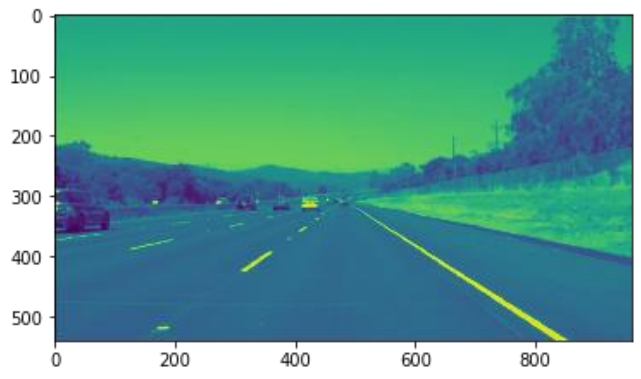
My final pipeline consisted of the below 6 steps in order.

1) I converted the original image to grayscale.
2) I applied Gaussian smoothing using the kernel _size of 5.
3) I ran Canny edge detection with low_threshold of 60 and high_threshold of 120.
4) I masked the image and blackened out unwanted regions.
5) I ran the Hough transform on detected edges and created a single line for left lane and also for right lane. For Hough transform, my parameters were rho of 1, theta of 1 radian, threshold of 30, minimum line length of 10 and maximum line gap of 5.
   In order to draw a single line on the left and right lanes, I modified the draw_lines() function by classifying detected edges into two groups, a left group and a right group. If an edge had a slope less than or equal to zero and the x-coordinate of first point was less than 480, which is the horizontal mid-point of the image, it was classified as a left group. If it had a slope greater than or equal to zero and the x-coordinate of the first point was greater than 480, it was classified as a right group. Otherwise, the detected edge was not used.
   The end points of the detected edges were stored for each group. Each detected edge that met the right or left group classification would provide two points of data since an edge has a begin point and an end point. After that, I ran the numpy polyfit function to compute the coefficients of a line that minimizes the sum of the squared errors for those points. I used those coefficients to draw left and right lines from a y-coordinate of 320 to 540, which is a little more than the half-way from the top to the bottom of the image.
6) I combined the Hough transformed lines with the original image.

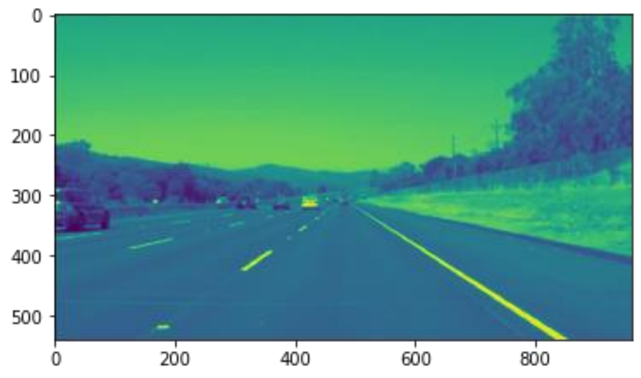
Here are images from each step taken in my pipeline. The original image is solidWhiteRight.jpg.

0) original image

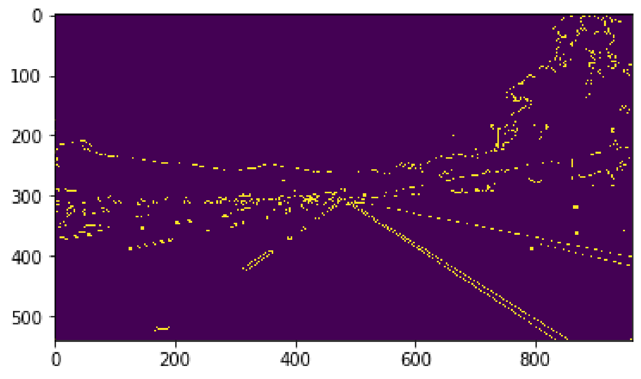
1) after grayscale



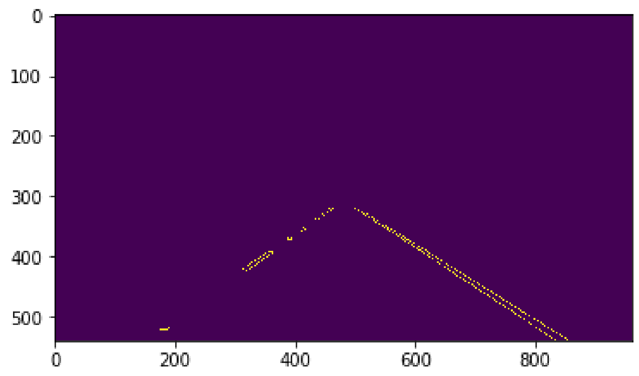
2) after Gaussian smoothinig



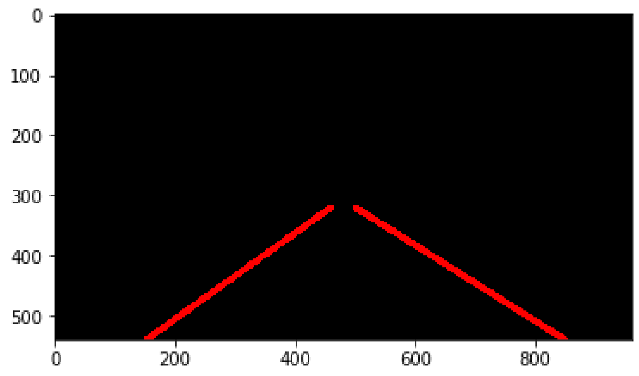
3) after applying Canny edge detection



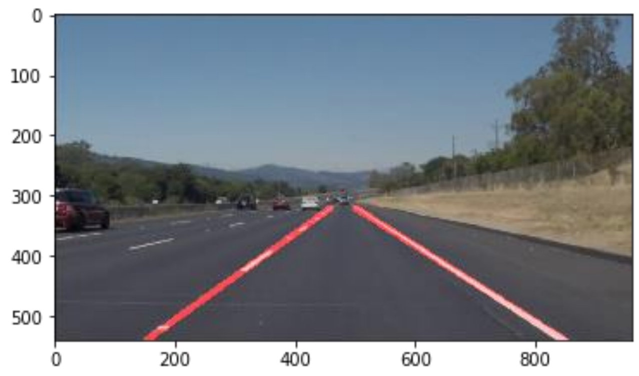
4) after masking



5) after hough transformation and extrapolating into a left and right lines



6) combining to the original image



2. Identify potential shortcomings with your current pipeline

Given that this is a fairly simple pipeline, there are several potential shortcomings.

- If a road has a sharp curve or a turn, a straight line will not be able to cleanly fit through the curve.
- The provided images and videos had the car in the middle of the road. However, the criteria for identifying left and right lanes will fail if the car is not in the middle of the road (for example, when a car is changing lane).
- This pipeline currently does not distinguish single line from double lines, and dotted lines from solid lines. We need ability to identify different types of road lines.
- Lines may not be clearly visible on an image because they are worn out or are in darkness. Also, yellow lines in shades are not well recognized in grayscale. These will cause lines to be not detected in the pipeline.

3. Suggest possible improvements to your pipeline

- A possible improvement would be to use a higher degree polynomial fit. While this could help fit better on the sharper curves, I could see it causing problems when there is a need to extrapolate far when detected edges make up only a small section of the whole line. To counter this, a straight line can be used when there are not enough detected points across the expected line, while higher degree polynomial fit can be used if there are enough data.
- Color differences, hence edge detection, does not work well for yellow lines when there are shades or dark spots in grayscale images. By changing the color space to HSL, both white and yellow lines can be better detected in these cases.