Finding Lane Lines on the Road

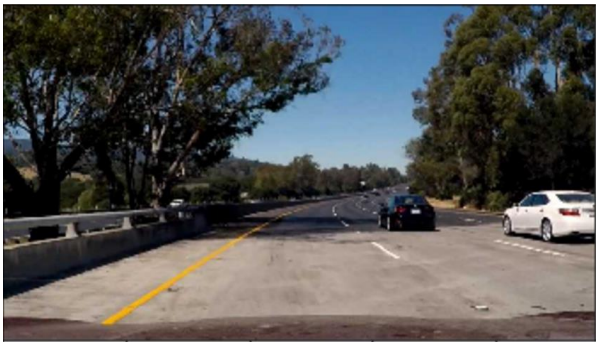
The goals / steps of this project are the following:

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

Reflection

1. Pipeline

To create a pipeline that can be used in the challenge section, a screenshot of the challenge video was taken. The screen shot was resized to 960x540 pixel and JPG compression to about 60 kB. Below is a picture of the screenshot taken.



Before Canny is applied, a greyscale image is needed. Below is a greyscale picture of the challenge right curve.

Notice the left lane is hard to distinguish. Looking at 3 different color channels, red, green, and blue, the blue channel makes the left lane visible again. Below is the grayscale of the blue channel.

Now, the above findings are incorporated into a canny pipeline called “rgb_canny”:

1. Separate input image into red, green, and blue channel and store result for next step.
2. Convert each of the color channels to grayscale and store result
3. Apply Gaussian blur to each of the grayscale channels, using kernel size parameter, and store result
4. Calculate high and low threshold for Canny filter based on RGB to Grayscale equation and low threshold input parameter. (g = 0.299R + 0.587G + 0.114B) i.e. The low threshold for red channel should be 0.299*threshLow. If thresholds are not scaled correct, everything will be filtered out.
5. Apply Canny filter on each blurred grayscale image using calculated high and low threshold. Store result of Canny filter for each channel.
6. Return the logical or of the 3 Canny filters. Output have to be formatted to 0 or 255 to display a black and white image.

The lane line pipeline was built starting from “rgb_canny”:

1. Apply “rgb_canny” to input image with specified input parameters and store result.
2. Build a region to show mainly where the lane line should be.
3. Apply image mask to result of “rgb_canny” to reduce numbers of lines calculated and generated in Hough line function. Store result of masked image.
4. Apply “hough_line” function using specified input parameters and store result.
5. Return a compound of input image with generated Hough lines on top using “weighted_img.”

The “draw_line” function was changed to “draw_polyline” and was enhanced accordingly:

1. Slope was calculated, from Hough line points, and is filtered out if the angle of the line is less than 20º or greater than 80º, accounted for sign of slope.

2. The Hough line points that passes the filter are essentially reshaped in array of x and y points. This is done separately for two lines because one line will have a positive slope while the other will have a negative slope.
3. If array of Hough point exists, based on parameters of Canny and Hough lines, calculate polynomial fit of the points. Linear (degree of 1) is the safest. However, if enough points exist, curve in lane line can be fitted by a polynomial equation.
4. Generate line (curve) based polynomial equation.

2. Potential Shortcomings With Current Pipeline

The main shortcoming with the current pipeline is the polynomial fit is very susceptible to noise when there is low amount of data. If the parameters for Canny and Hough lines are not tuned optimally from frame to frame, then the lane line can have great variance. The challenge video shows lane line jump to almost horizontal for a few frames before correctly display lane lines.

3. Possible Improvements To Current Pipeline

A possible improvement would be limit the change of slope from frame to frame. Since lane lines does not turn 30º after one frame, limiting the change of slope to a specified amount should take out lane lines that is not physically possible for limited noisy data.

Another potential improvement could be to have better quality images with contrast boosting. This will help Canny filter detect more edges, in turn, helps Hough line with more data points.