Writeup Template

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

My pipeline consisted of 9 steps, detailed as follows:

1. Import relevant packages
    - 1.1. Originally, I placed the packages to import and defined relevant helper functions inside the 'process_image' function, so that it would be fully self-contained. I subsequently removed both and placed each in its own cell (packages & helper functions) to aid in the process of debugging.
2. Import image
    - 2.1. Image is imported with mpimg.imread() outside of the 'process_image' function
3. Grayscale image
4. Gauss blur grayscale
5. Apply Canny edge detection
    - 5.1. Canny bounds were automatically specified as +/- one third the median image channel intensity.
6. Trim image to region of interest
    - 6.1. Trapezoid edges specified as half y-dimension plus t_shift buffer, half x-dimension +/- t_shift buffer (to change triangle to trapezoid), and x-axis edges pulled in by b_shift.
7. Generate Hough transform lines
    - 7.1. Constants set based on trial & error/fit optimization
8. Fit line to Hough points and generate image of lines over black mask
    - 8.1. Detail below
9. Overlay lines on original image and output

In order to draw a single line on the left and right lanes, I modified the draw_lines() function by creating a line of best fit given Hough line points for left & right lane lines. I created the left/right line of best fit based on the following pipeline:

1. Generate array of Hough line edge points from cv2.HoughLinesP function
2. Split edge points into left lane/right lane line buckets based on calculated segment slope

3. Fit line to left/right lane arrays using np.polyfit, which returns the slope/y-intercept of left/right lane lines.
4. Calculate edge points of left/right lane lines using region of interest boundaries

2. Identify potential shortcomings with your current pipeline

One potential shortcoming would be what would happen when:

1. The image horizon is significantly above image y-axis midpoint
2. Large/hard curves in lanes

Another shortcoming could be:

1. The overflow error encountered in the ‘challenge’ video
    1.1. I am in the process of fixing this
2. Empty arrays throwing exceptions to np.poly function
    2.1. Short term workaround of passing if array is empty, and making np.poly outputs global so they keep previous values in case of empty array

3. Suggest possible improvements to your pipeline

A possible improvement would be to make sure constants can be specified in one place and apply to all instances throughout the process_image function and helper functions (t_shift, b_shift).

Another potential improvement could be to apply more temporal smoothing to lane lines so they aren’t so jumpy. Also, if there were a way to optimize constants (Hough lines, Canny edges) so that there were less bad inputs to the np.poly function, the lines would likely better fit the lane edges on a more consistent basis.