Eric Victorson
2/21/2017

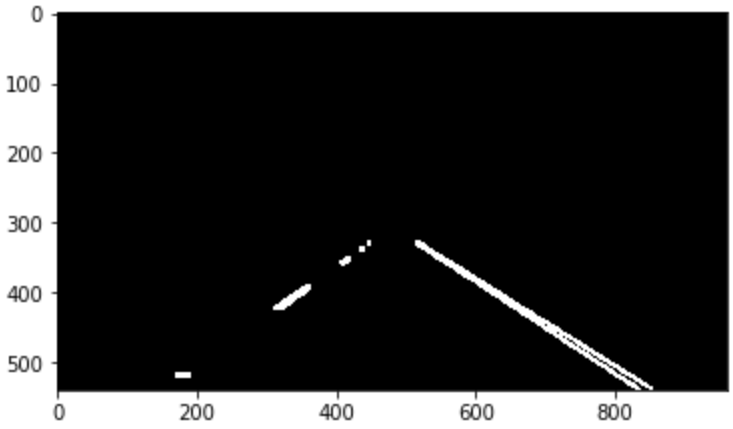
Self Driving Car Term 1 Project 1
Detecting Lane Lines

Reflection

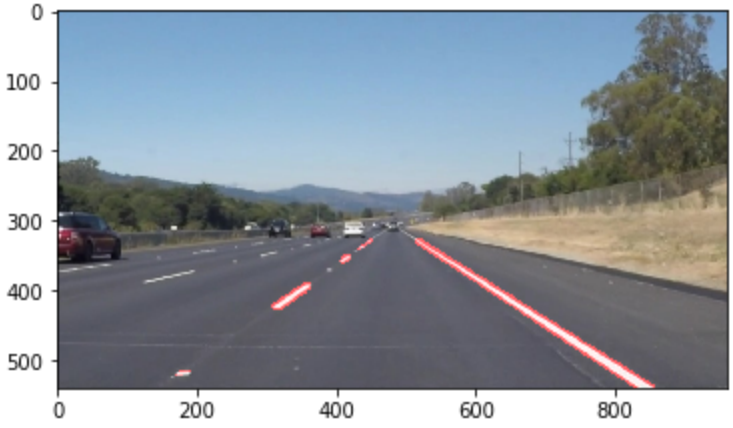
1. Describe your pipeline. As part of the description, explain how you modified the draw_lines() function.

My pipeline consists of the following steps:

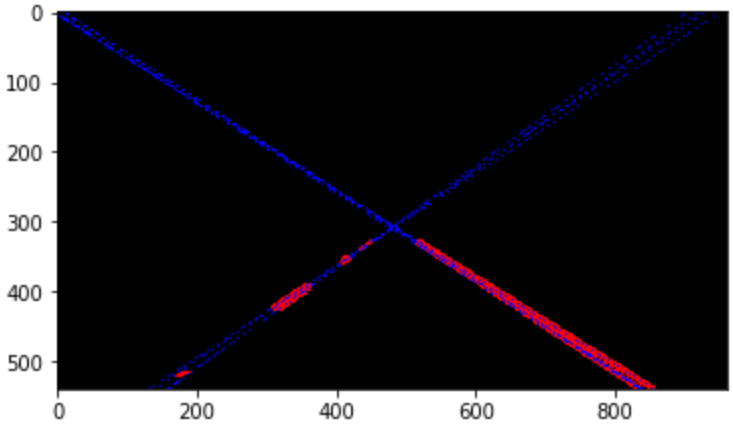
1) Convert image to grayscale.
2) Threshold the grayscale image.
3) Apply region of interest.
4) Perform canny edge detection (dilated here for easier viewing):



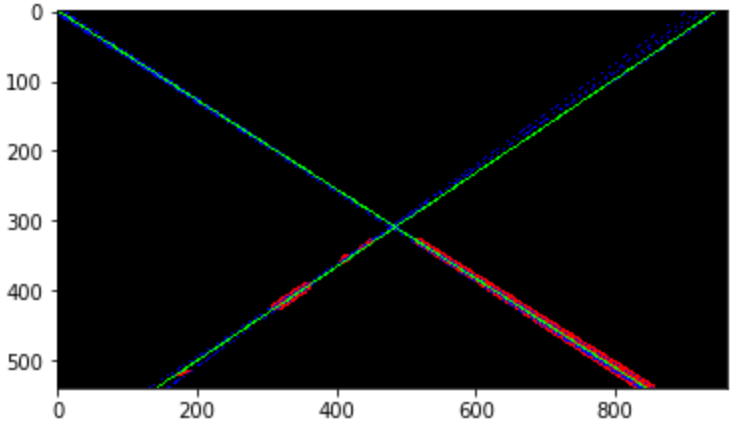
5) Perform hough transform (here with lines drawn)



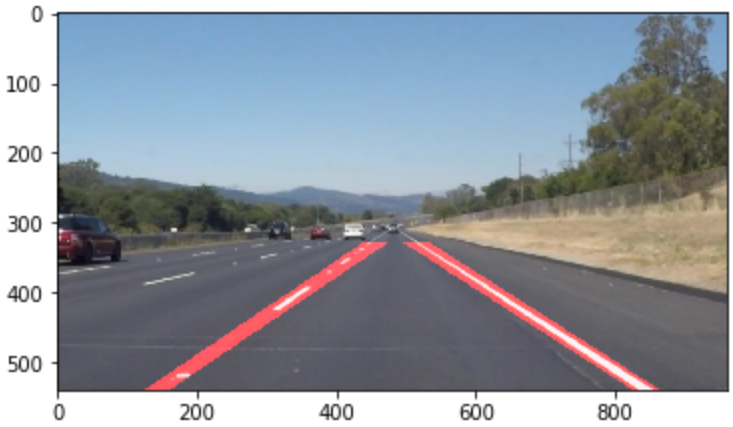
6) Extend hough transform lines and bin them by positive and negative slopes:
    a) Extending is done by taking the two tuples (x1,y1), (x2,y2) output from houghLinesP(), determining the slope for each line, and then determining a leftmost and rightmost point across the image to draw a line between.



7) Determine if both lane lines are detected, if one is not, draw one that is offset from the other lane line by a nominal amount so it is a function of the detected line.
8) If both lane lines are detected, perform least squares fit on positive and negative sloped lines:



9) Draw these least squares fit lines on an empty array and perform region of interest trimming to get rid of line segments extending beyond the horizon.
10) Overlay these lane lines on the original image and display them:



Steps 6 through 10 are my improvements to the draw_lines() function, although not integrated into standalone functions; everything is kind of a bit of spaghetti code currently. If this is an issue I'd be happy to refactor the code into a cleaner workflow.

2. Identify potential shortcomings with your current pipeline

Potential shortcomings arise from situations where lane lines do not have high contrast with their background, when lines are occluded, or they are shadowed. For these situations there is no recovery behavior and vision can only be of so much assistance.

Another shortcoming is that if lane lines become too misaligned from the slopes expected they will not be registered as lane lines due to the binning process I, have which sorts things by their determined slope.

3. Suggest possible improvements to your pipeline

One large improvement would be to low pass or kalman filter the positions and slopes of the lane lines so that they are not memoryless like they currently are. This would dampen (add inertia) to the system and cause the lines to jitter much less.

Another potential improvement could be to filter the color image according to an RGB mask, as opposed to my simple grayscale threshold.

A further improvement would be to monitor the residual error terms or the covariance of the least squares line fitting to determine how much certainty there is in the fit. This could be used to monitor the state of the detection algorithm, and if the covariance becomes large some kind of recovery behavior within the code could be triggered.