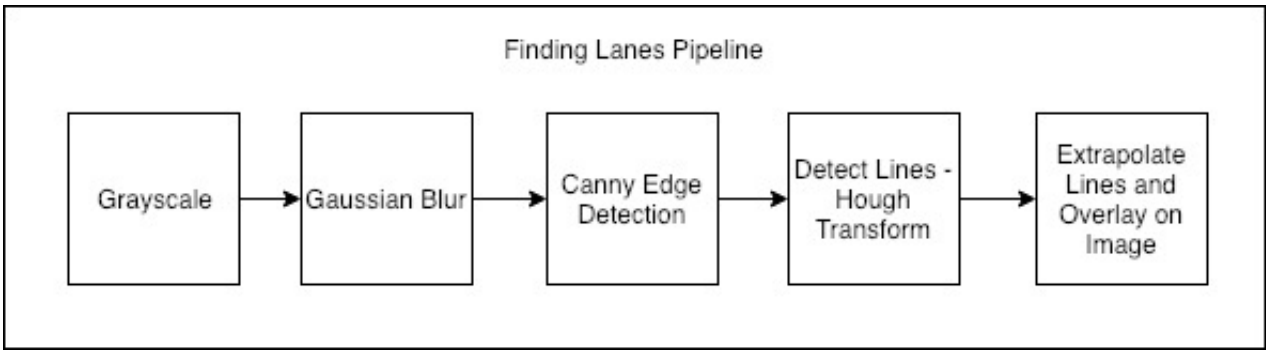
Finding Lane Lines on the Road

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

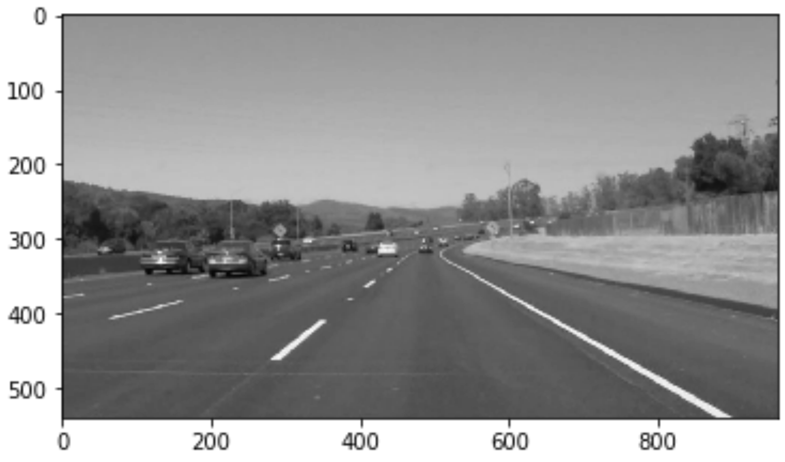
Reflection

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

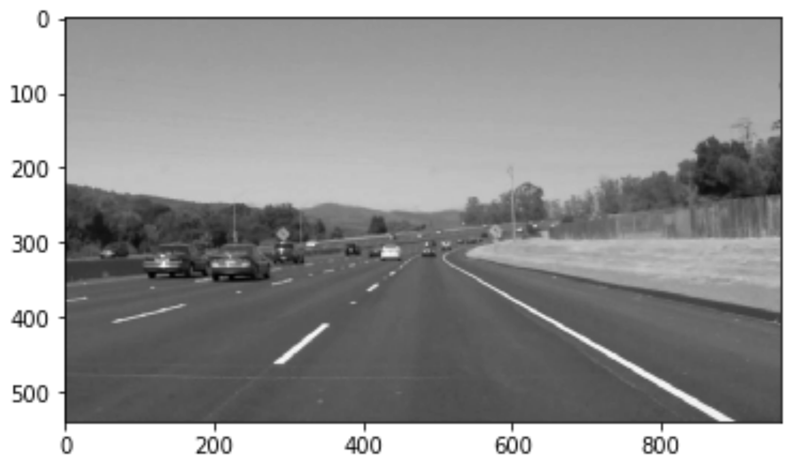

Finding Lanes Pipeline
Grayscale
Gaussian Blur
Canny Edge Detection
Detect Lines - Hough Transform
Extrapolate Lines and Overlay on Image

My pipeline consisted of 5 steps:

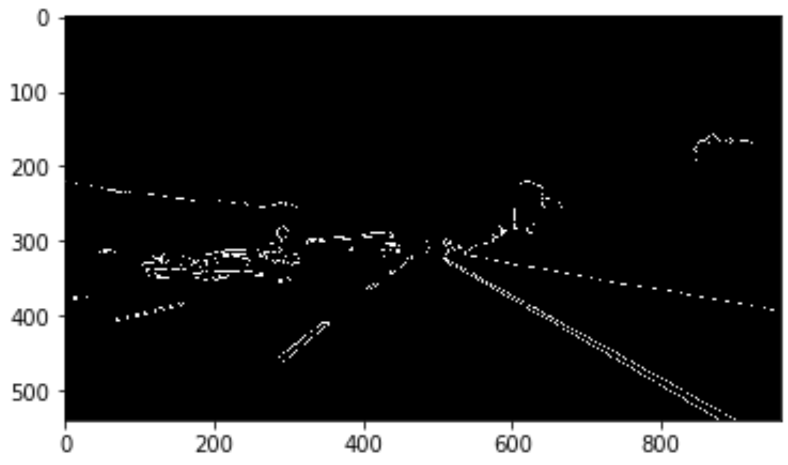
- 1. I converted the image to grayscale.



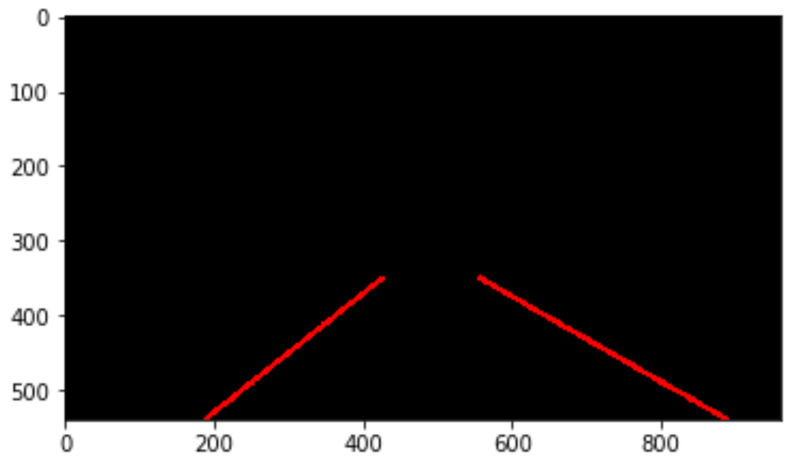
- 2. I applied the gaussian blur image filter on the grayscale image with a kernel size of 3.



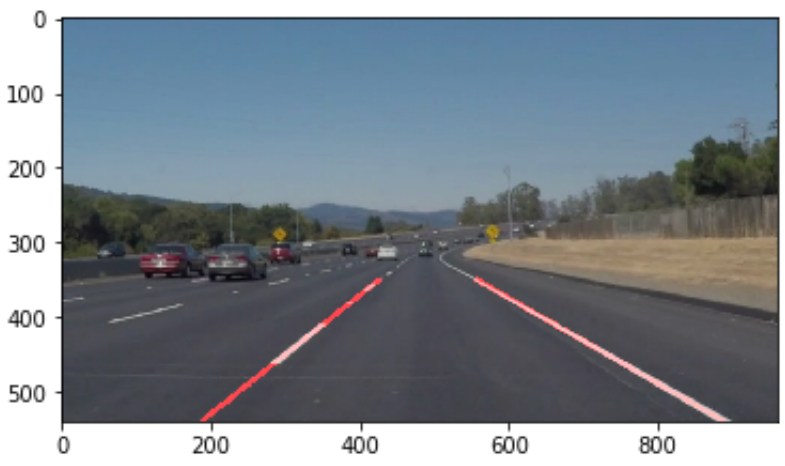
- 3. I applied the canny edge detection over the blurred image.



- 4. I detected lines using probabilistic hough transform algorithm based on a region of interest.



- 5. I extrapolated the detected lines and drew the lines over the image.



In order to draw a single line on the left and right lanes, I modified the draw_lines() function by calculating the slope of each detected line to categorize each line to either left lane or right lane, I then extrapolated the lines with gradient above threshold of 0.5, then for each set of lines belong to each lane, I found a fitted line using the polyfit algorithm in mumpy.

2. Identify potential shortcomings with your current pipeline

1. If the gradient thredhold used is not properly choosen, it will lead to poorly extrapolated lines, this is very apparent towards the end of the test_videos/solidYellowLeft.mp4 video.
2. When there is very bright sun, the edge detection algorithm will not be able to detect the lanes correctly.

3. Suggest possible improvements to your pipeline

1. Using an improved algorithm to extrapolate the lines, preferrably one that does not depend on a constant value (gradient threshold), but on a variable that can be calculated based on the lines or region of interest.
2. Using an improved way of detect the lanes.
3. Find a better way of extrapolating the lines for crazier scenarios like the challenge video.