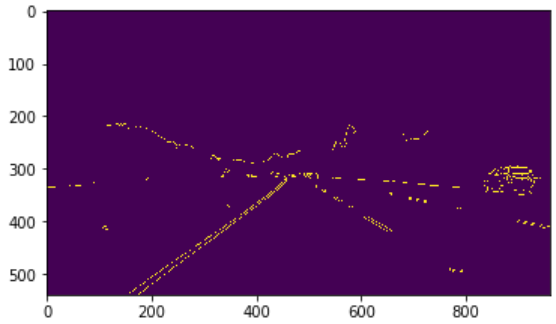
Meredith Meyer

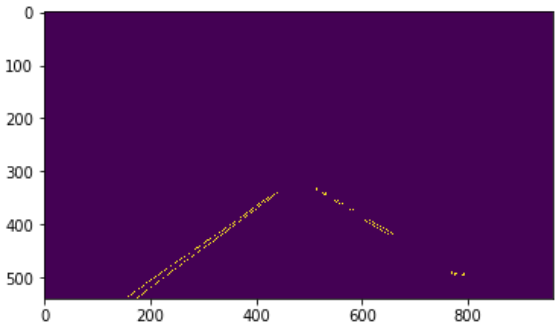
Project 1: Finding Lane Lines

1. **Pipeline**:

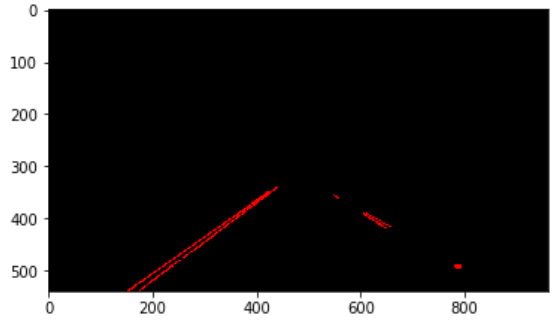
I used the general approach taught in the lesson to construct my pipeline. First I converted to a grayscale image, then applied a gaussian blue of kernel size 5. I used the Canny function to detect edges:



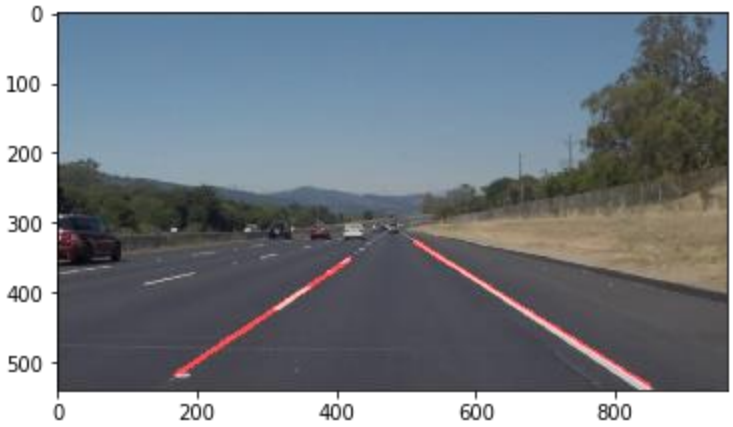
Then I defined a region of interest- an almost-triangle quadrilateral that would include the lane lines:



After defining the area of interest, I used the hough_lines function to find lines in the image, which would hopefully be the lane lines:



After finding the hough_lines, I chose to find a single line representing each lane using my own algorithm. Using the Hough lines, I categorized the lines by slope, and found the points which defined the starts and ends to each of the left and right lanes, so that missing pieces would be connected. Finally, I used the weighted image function to overlay red lines over the lanes:



2. **Shortcomings**:

There are many things I can see going wrong with, or have gone wrong with, my method of drawing the lines. The first major shortcoming I anticipate is identification and characterization of non-lane lines as lane lines. As coded, any line left in the region of interest is assumed to be a lane line, although that is not necessarily the case. Another issue is with incomplete lines, such as is typical between lanes of the same direction. If the bottom of the video/image does not start with a line, the drawn lane line will not extend there, which is likely to be an issue in the future for an actual self-driving car.

3. **Improvements**:

I had attempted, but ran out of time and was unable to complete, a better algorithm for drawing the complete lane lines. The still images in test_images all came out fine with my test algorithm, but some frames of the videos caused something to go awry. I'd like to do further investigation to figure out what is being miscalculated in those frames, and make corresponding changes to my pipeline. See below an example image showing the before and after of this algorithm. With this, there would always be 2 lane lines extending fully from the bottom of the frame to the top of the region of interest. I suspect the current issue is actually in the definition of the line, rather than the extension of the available line. I could also condense and better comment my code for improvements.

Before line extension

After line extension