###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 gray scale
2. Apply a 5x5 Gaussian blur
3. Apply canny edge detect transform with the following threshold
    a. Lower – 100
    b. Upper – 200
4. Apply a trapezoidal mask to isolate the area of interest with the following coordinates
    a. [130,540]
    b. [430,330]
    c. [530,330]
    d. [900,540]
5. Apply Hough transform to detect lines
    a. rho = 1, theta = PI/180, threshold = 20, minLineLength = 10, maxLineGap = 80
    b. Extend lines to the bottom of the image
    c. Draw lines on a blank canvas
6. Combine original image with Hough transform lines superimposed on top

I modified the draw_lines() function by creating a new function called draw_final_lines()

In this function, I found the end points of the extrapolated lines based on the slope and y-intercept derived from line segment outputs from the Hough transform. The endpoint y coordinates are the bottom of the frame and y = 330, which I have determined to be a reasonable estimate of where the linear region of the line marking ends in the test videos. I then "filtered" the slope of the line segments to remove lines based on a reasonable expected range given the perspective of the camera in the test videos (this is done by checking the "slope" variable in the function).

###2. Identify potential shortcomings with your current pipeline

One deficiency of the current pipeline is that it does not scale to arbitrary video frame resolutions. Another short coming is that it makes the highlighted line wider than the actual lane markings overall.

###3. Suggest possible improvements to your pipeline

A potential improvement is to make the coordinates for masking and slope filtering parametric based on the input video dimensions (as supposed to hardcoded in the current pipeline implementation).

The extrapolated line can also be a single line (vs multiple lines overlapping) if the slopes of all "valid" Hough transform lines are averaged before drawing.

The challenge video seems to throw off the canny edge detection algorithm when the color of the pavement marking changes. This suggest that perhaps an alternate to the grayscale transform is needed (e.g. color space separation, YUV, YCbCr) to make edge detection more consistent and robust.