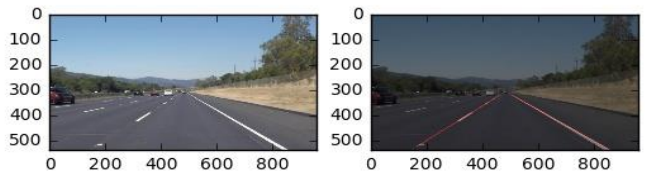
1. Describe your pipeline. As part of the description, explain how you modified the draw_lines() function.

   My pipeline consists of following steps

   a. First step is to prepare image for canny edge detection. This leads to following steps
      a. Capture the image in gray scale.
      b. Apply gaussian blurring.
   b. Canny edge function to detect the edges. Instead of hard coding the thresholds, median value of pixel intensity is selected and +/- sigma values = 0.40 is applied to define low and high thresholds.

      V = median of pixel intensity
      lower = int(max(0, (1.0 - sigma) * v))
      upper = int(min(255, (1.0 + sigma) * v))

      This simple stats formula gave good pixel gradient selection. More details found at research link
   c. Areas of interest vertices were calculated and applied on canny edge function's output to filter out all edges other than once within area of interest.
   d. Output Image from above step(i.e. after segregating area interest) is processed using hough's line generator function. Hough's lines generator function identifies many lines and therefore, some custom code written in helper functions(draw_lines) to identify min/max coordinates that can be used for extrapolate into single large line.
   e. Actual image(input image) and image with extrapolated lines are overlapped by adjusting some parameters in weighted_img functions to display extrapolated lines transparently over lanes edge markings.



2. Identify potential short comings.
   a. Lines detection does not work on videos or images with curved lane edges. Primarily because extrapolated line generated is linear and does not consider joining curved lane edges.
   b. In of presence of car or person in front of car, algorithms might need more tuning to filter out car or person to visualize the lane smooth. As recognizing car or person to stop the car is not part of this project and it can be ignored.

3. Suggest possible improvements to your pipeline
   a. If "area of interest" identification can be done as first step in the pipeline, it can improve performance of code a lot. This allow canny_edge also to apply only on area of interest. When I tried this approach, trapezium used to identify area of interest is also getting identified as edge. Some more thinking can be put in to handle this approach.
   b. Area of interest boundaries(trapezium edges) are kind of pre-defined. Some more thinking can be put in place to dynamically identify required boundaries.

c. Identify an method to optimize Hough’s line algorithm parameters.
d. Improve draw_lines function to handle curves as well.
e. Overlaying of extrapolated line image with actual image using weighted_img function was reducing the brightness of final image. This can be modified to improve brightness in final image.