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 using some Helper functions.
- Apply this pipeline on some test images in order to check its performance.
- Extend the testing phase by applying this pipeline over some test videos. Actually, Videos are considered as group of images.
- Enhance this pipeline by modifying draw_lines() function. The aim is not only drawing Lane Lines on the Road and having many lines for each lane, but also applying some enhancements over these Lines. These enhancements can be formulated in applying extrapolation over each lane line in order to have only one Lane Line starting from the beginning of the Region of Interest (ROI) till its end. At the end of this point we should have two continuous Lane Lines.
- Apply the modified pipeline over both: test images, and test videos.
- Try to apply the modified pipeline over a challenge video contains different colors especially for yellow color.

Reflection:

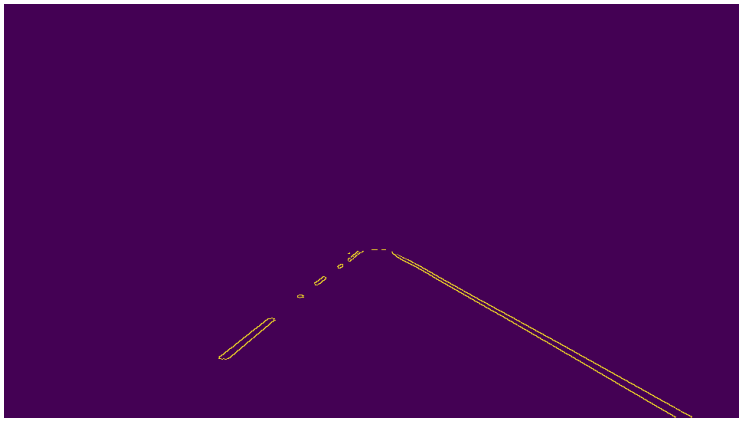
1- Pipeline Description:

My pipeline consists of 5 steps which are as follows:

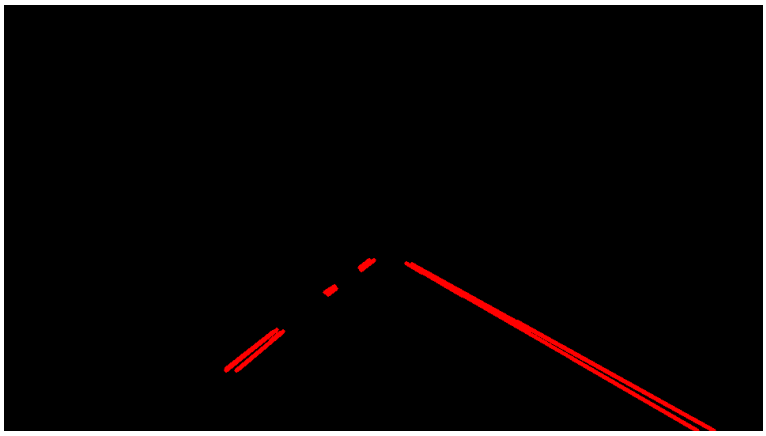
a- Read the image and change it into GrayScale, then apply some Gaussian blur with a square kernel of 5 pixels each side for the aim of smoothing.

b- Apply a cv2 function called Canny to extract the Lines edges.

c- After that we defined the vertices of the Region of Interest (ROI) and apply it over Lines edges



d- Apply Hough transform that returns every line in the ROI with the definition of (starting point and ending point).



e- Iterate over the whole lines extracted from the previous step, then use a cv2 function called line to draw these lines over the original image.

2- Pipeline Modifications:

In order to draw a single line for each of the two lane lines, I have modified in draw_lines() as follows:

a- Every line can be expressed by its (Slope and Intercept) parts according to the simple equation of line: $y = mx + c$.

b- I have created a new function called Average_slope_and_intercept() which separates the lines into two categories left lines and right lines, then depend on the length of the line to be act as a weight for this line. Finally this function returns the slope and the intercept parts for both left and right lanes.

c- Actually cv2 need the starting and the ending points for the created line, so I have created a new function called make_line_points() which takes the slope and the intercept part, then depends on the y component of the Region of Interest to calculate x component. Finally, now we have the starting and the ending points for each line.

d- The modified draw_lines() function now is responsible for drawing the lines only for each lane using cv2.

3- We have applied the original pipeline and the modified pipeline over test videos in order check the performance of each pipeline, and then we saved these videos in the folder structure of the project.

4- Identify potential shortcomings with your current pipeline:
    a- One potential shortcoming would be what would happen when the vehicle is moving in a curved lanes, our pipeline will fail because it depends on extracting Lanes lines not Lanes curves.
    b- Another shortcoming could be when the lanes color changed as in the challenge video, the lane line will fail to extract the lines well.

5- Suggest possible improvments to your pipeline:
    a- A possible improvement would be to extend our pipeline to detect and extract the curved lanes.
    b- Another potential improvement could be to detect lanes with any changing color, not only the white color.