A. Project introduction.

Project goal expects line detection both on picture and movie. Base on supported function definitions (and OpenCV functions) it was necessary to build pipeline (here draw_line()) function which manages to draw lines on pictures and movies.

Project complexity grows while moving forward therefore some back improvement of software (development process iterations) had to be considered.

Included JUPYTER project detects first line on single picture. In order to verify result of algorithm detection the step results are shown (proceeded picture). Afterwards, the same steps (as for the single picture) are applied for the other provided pictures (in loop) and saved to local catalogue. Finally, the line detection on the picture is replicate for detection line on the videos. Here modifications/troubleshooting in draw line() function had to be applied again (due to occurred problems). Lastly, detection algorithm is checked on optional challenge video. Based on last test, the improvement conclusions are drawn.

B. Reflection:

The main idea for pipeline implementation (draw_line() function draws the lines) is to find four (x,y) points (two for each line: left and right).
Points are found by use of interpolation function: polyfit. When the points are calculated so the lines can be drawn.
Pipeline and draw_function() can be described together in following steps:

1. Convert the image to gray scale.
2. Gaussian filter 5x5 is applied (this smoothen the image).
3. Apply canny edge detection algorithm.
4. Mask on the image is applied (removal of unwanted image area)
5. Apply hough transform and plot the lines (the draw_function() is activated):

    a. Detect the slopes on the lines detected by canny algorithm

    b. Add to matrices (x,y) point for all detected lines. Matrices include data separately: (x and y) for start and end point of each detected line.

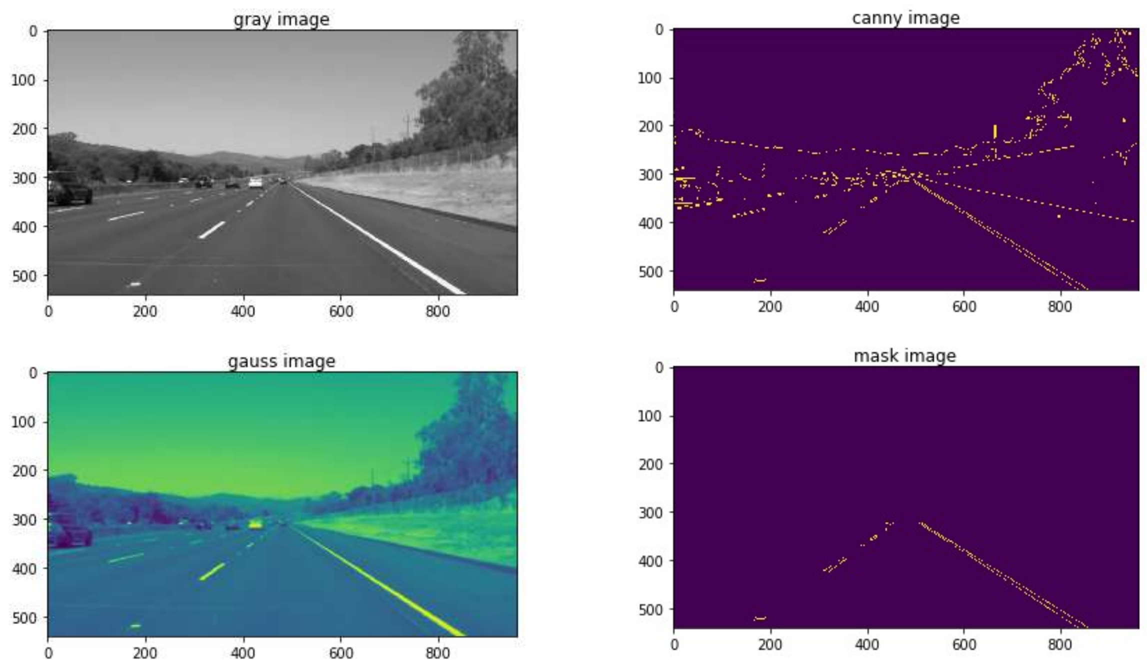
    c. Finding the extreme values (min and max) in above matrices.

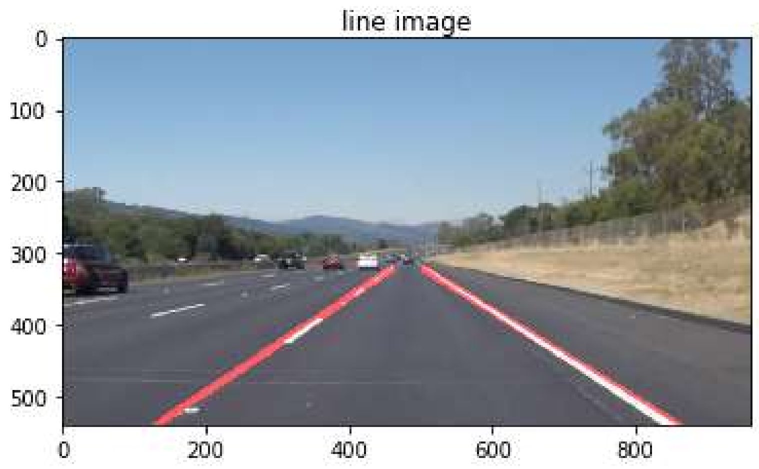
    d. Base on extreme (min and max) values perform the interpolation (finding m and b parameter of the interpolated line: y=m*x+b.

    e. Draw two lines using Opencv.

6. Connect image with interpolated lines in Weighted function.

Following figures depict how the implemented pipeline works:





C. Main shortcomings in current pipeline

a. While the pipeline in applied into video some fluctuation (in ends point) in interpolated lines can be seen.

b. Pipeline does not detect curves (it can be seen while the car moves on the curved road).

D. Possible improvements

a. Implementation of curve detections => with higher degree polynomial $p(x)$.

b. Introduction of buffer in order to predict lines while for some period of time any kind of lines can be detected (lines vanished out, other car hides the view or changes in brightness).

c. The pipeline should be enough robust to detect lines despite the road visibility, day time or condition on the road.

d. Interpolated lines should be averaged in order to reduce lines fluctuations.