<u>Finding Lane Lines on the Road</u>

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
- Reflect on your work in a written report

Reflection

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

My pipeline contained in five steps:

1. I converted the raw image into grayscale using open cv.

2. After that, I applied a Gaussian Blur with a kernel size of 3. The Canny algorithm provided by OpenCV already does that internally.

3. Applied Canny algorithm to detect the edges in greyscale image. This algorithm by detecting strong edges (strong gradient) pixels above a specified `high_threshold`, and reject pixels below `low_threshold`. It will then include pixels between `high_threshold` and `low_threshold` if they are directly connected to strong edges.

4. The next step is to define a region of interest. Because we know that the camera capturing the video is at a fixed position we can determine an area of interest using a 4-sided polygon starting from the bottom of the image. Every edge outside of this region will be discarded.

5. Finally, we apply a Hough Transform to detect line segments from edges detected by the Canny detector inside our region of interest.

In order to draw a single line on the left and right lanes, modified the draw_lines() function this way:

6. First, iterate over each line segments extracted by the Hough Transform step and fit a 2-degree polynomial on each extremity of the segment.

7. Then reject all segments where the slope is not matching certain conditions. We know we don't want any line segment that is roughly horizontal for instance so there is no need to include them in our calculation.

8. Then, based on the slope, sort segments into two arrays depending on whether they belong to the right or the left lane.

9. Once have all the segments for a given lane in the image, calculated the mean of all these line segments and use the slope and y-intercept to extrapolate the rest of the line.

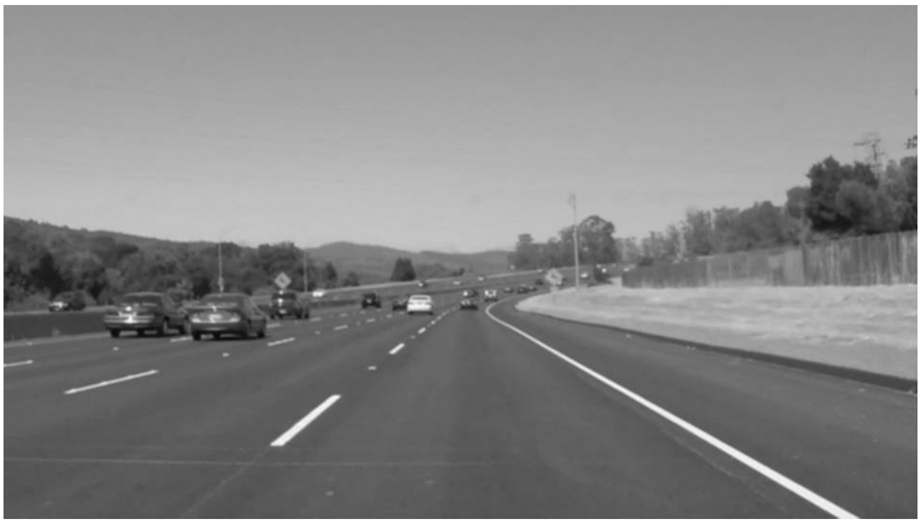
10. then draw the extrapolated line for each lane, and stores the polynomial coefficients so that I can use them on the next frame to smooth the transition between each frame.

11. Finally, I merge the image with the extrapolated lane lines onto my original image.

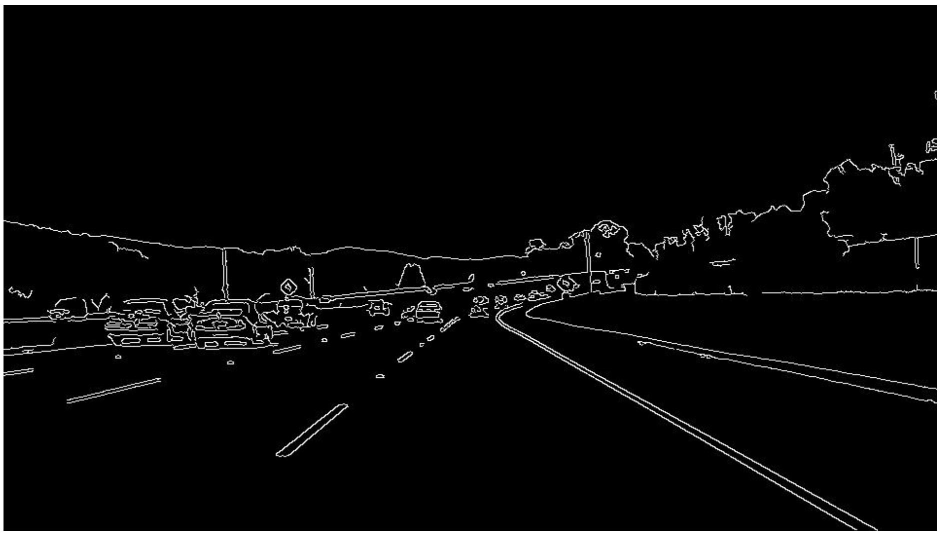
These are the Images which I got from pipeline step by step:

Raw Image -→ Grayscale

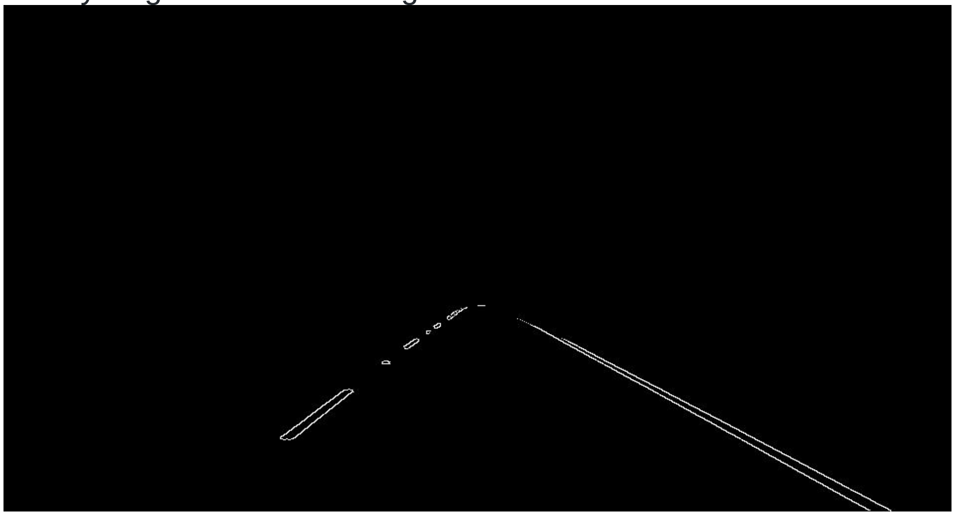
Grayscale→Gaussian blur



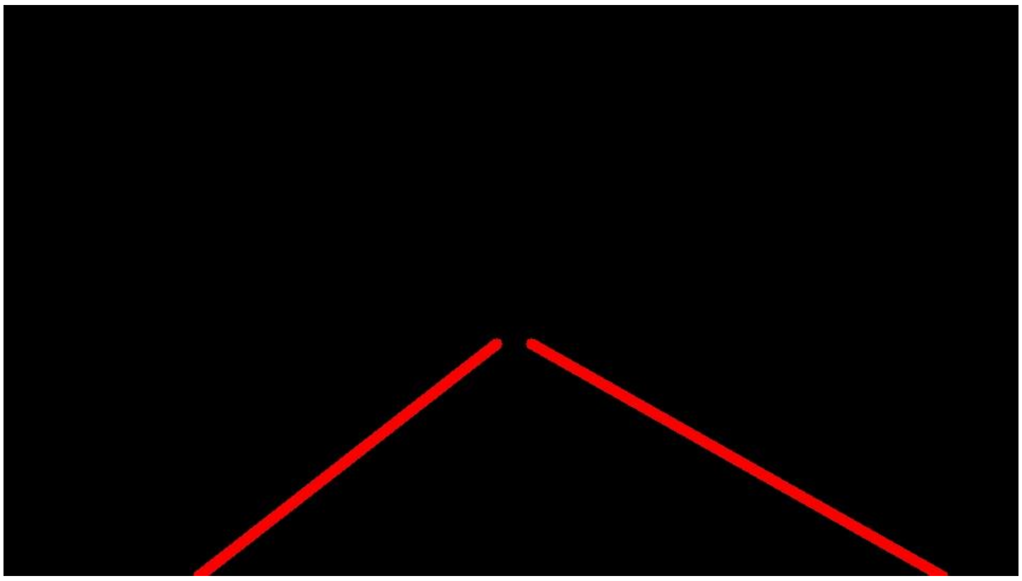
Gaussian blur→Canny Edge detection



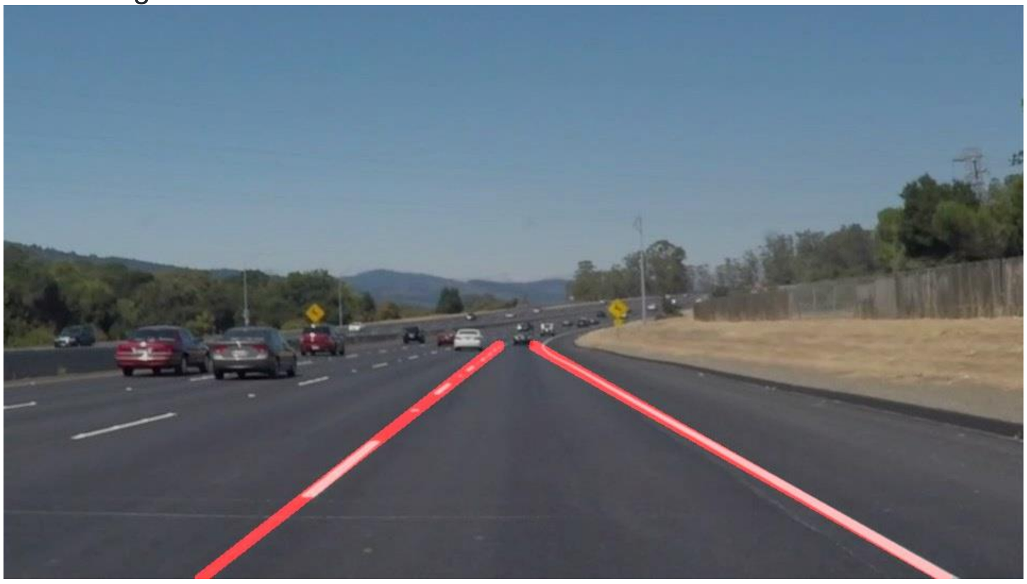
Canny Edge detection→ Region of interest



Region of interest→Hough transform and lane extrapolation



Final image:



2. Identify potential shortcomings with your current pipeline

There are many shortcomings with the current version of pipeline:

- I have not implemented any kind of color selection like HSV so in difficult color scenario my pipeline would not work so well. I am already getting some bad result on the challenge video.
- Straight lines do not work when there are curves on road.
- Hough Transform is tricky to get right with its parameters, I doubt I got good settings

3. Future Improvements

A simple improvement would be to store all past coefficient values and use a weighted average to compute the new coefficient.

A possible improvement would be to apply some kind of horizon detection so that we don't rely on hardcoded value to specify the region of interest. This mean the pipeline would work well in slopes.

In the future, I also plan to use deep learning to identify lanes and compare those results against what I obtained with a pure computer vision approach.