Project 1 - 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
- ❖ Get more idea about lane detection fundamentals
- ❖ First step towards documentations
- ❖ Identify potential shortcomings
- ❖ Possible improvements

My pipeline consisted of below steps :

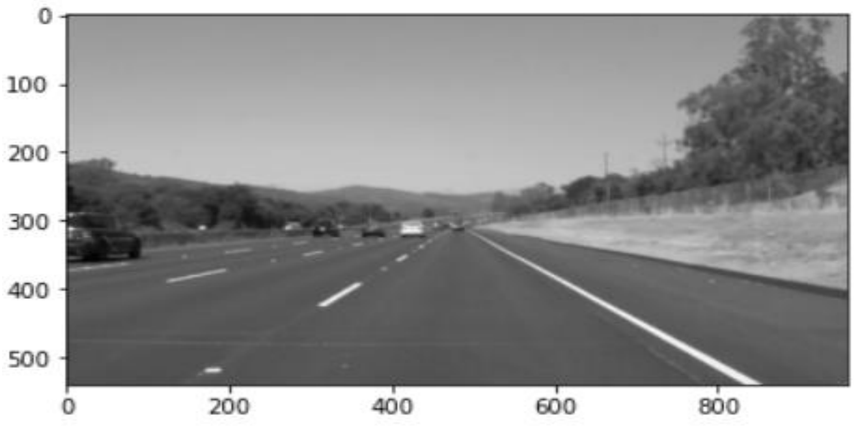
- Conversion of images to gray scale
- Finding region of interest
- Gaussian smoothing to remove the noise and blur the image .
- Canny Edge detection over gaussian image
- Apply the Hough transform on canny image.
- Applying these results on actual image as weighted image.
- Averaging the lane lines
- Applying results on video(which is nothing but series of Images)

If you'd like to include images to show how the pipeline works :

Read Image :

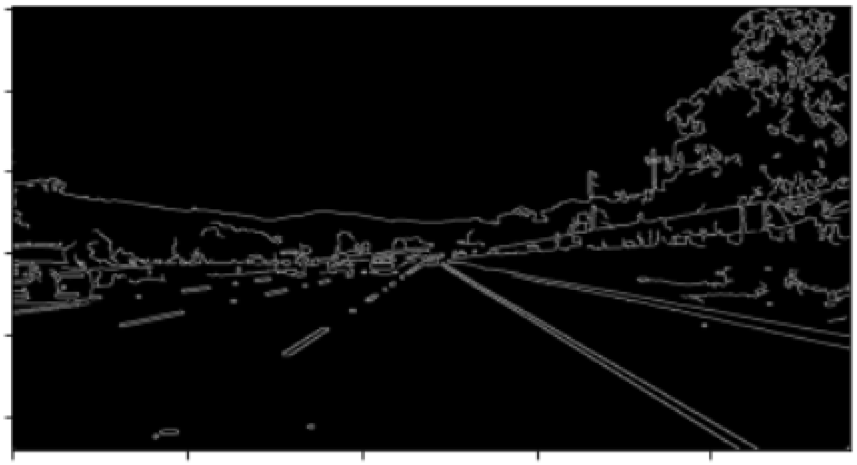
```
Out[4]: <matplotlib.image.AxesImage at 0x2212e940640>
```

Convert into Gray Scale:

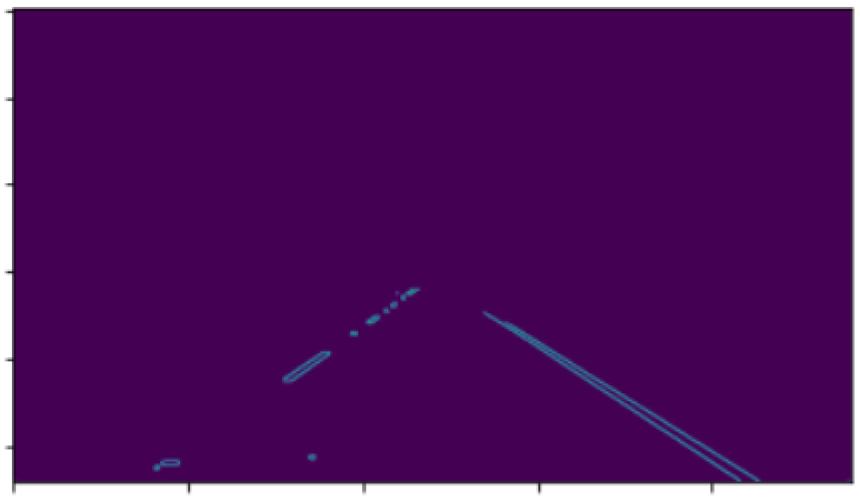


Gaussian Smoothing to suppress noise

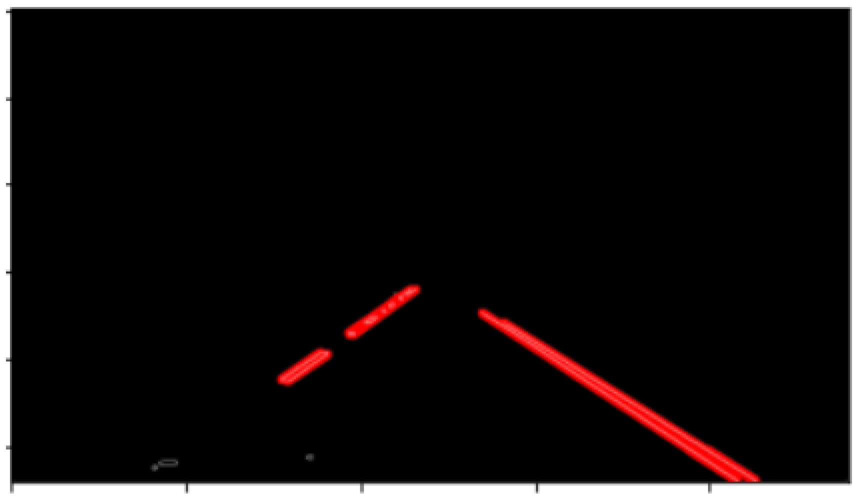
Canny Edge Detection



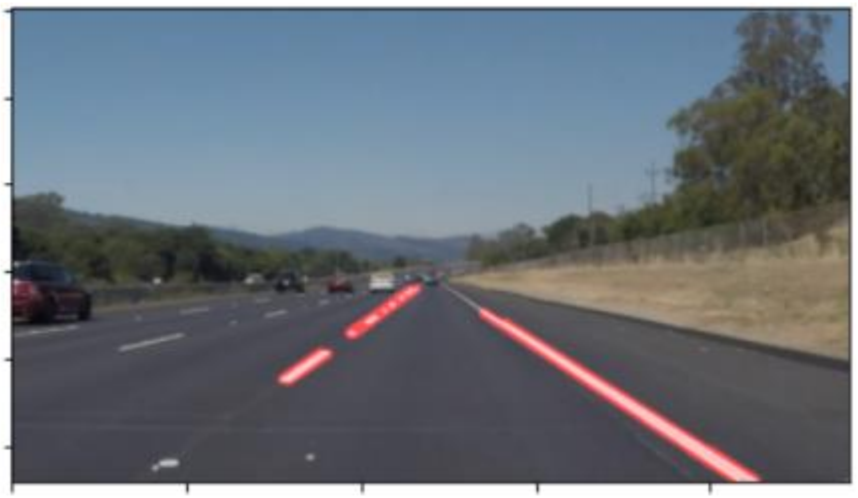
Finding Region of interest using polynomial filter



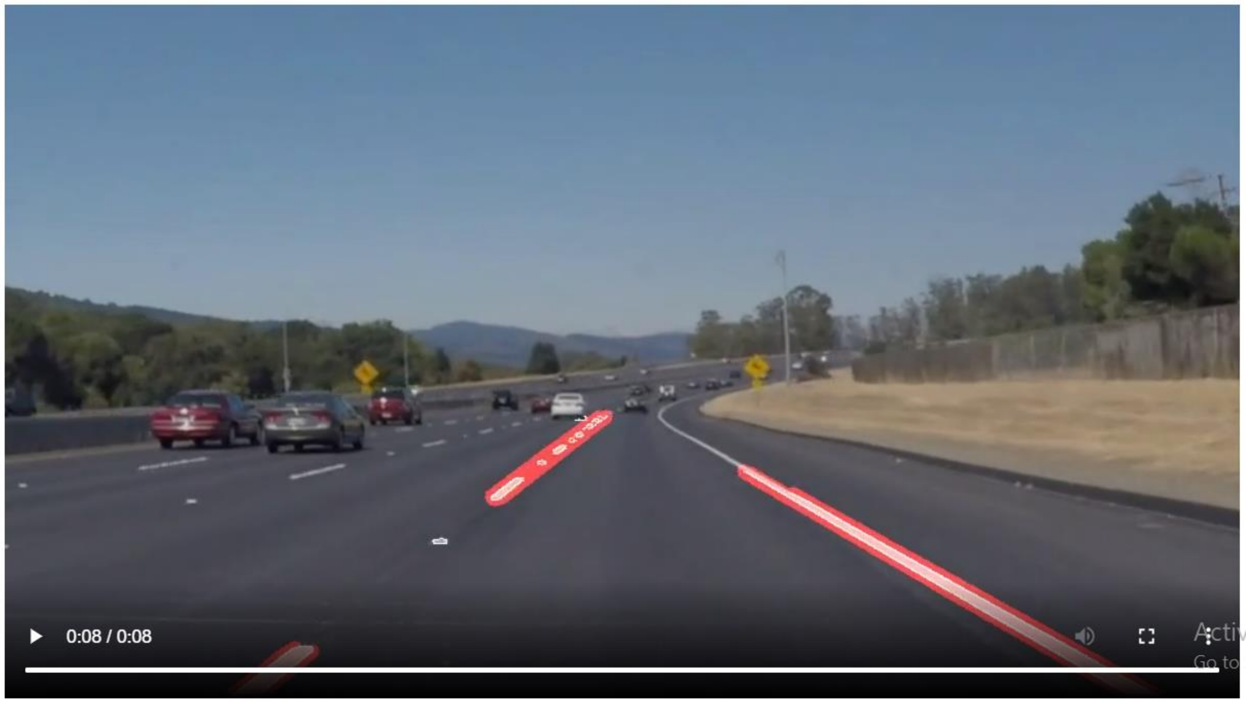
Hough Transform

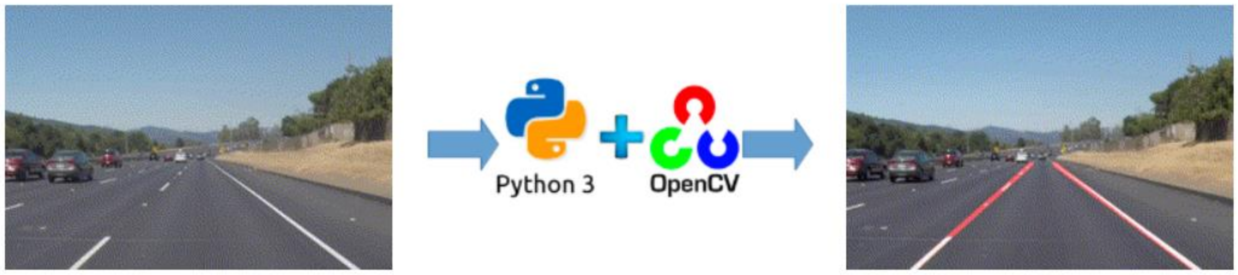


Apply on input image



Defining the functions at one place to apply results to video frames:





So far, journey for me was full of highs and lows. But now most difficult part for me personally was to average out the lines which are detected as shown in above video. I interacted with many Udacity students , GitHub repositories with YouTube tutorials and able to apply the theoretically understood concepts as :

Hough transform function returns the list of lines where every line is defined by 2 points and all these lines should be a part of the lane lines. There are 2 groups, the ones with positive slope and the ones with negative slope. For every group of lines, I found the average slope and created the line with that slope.

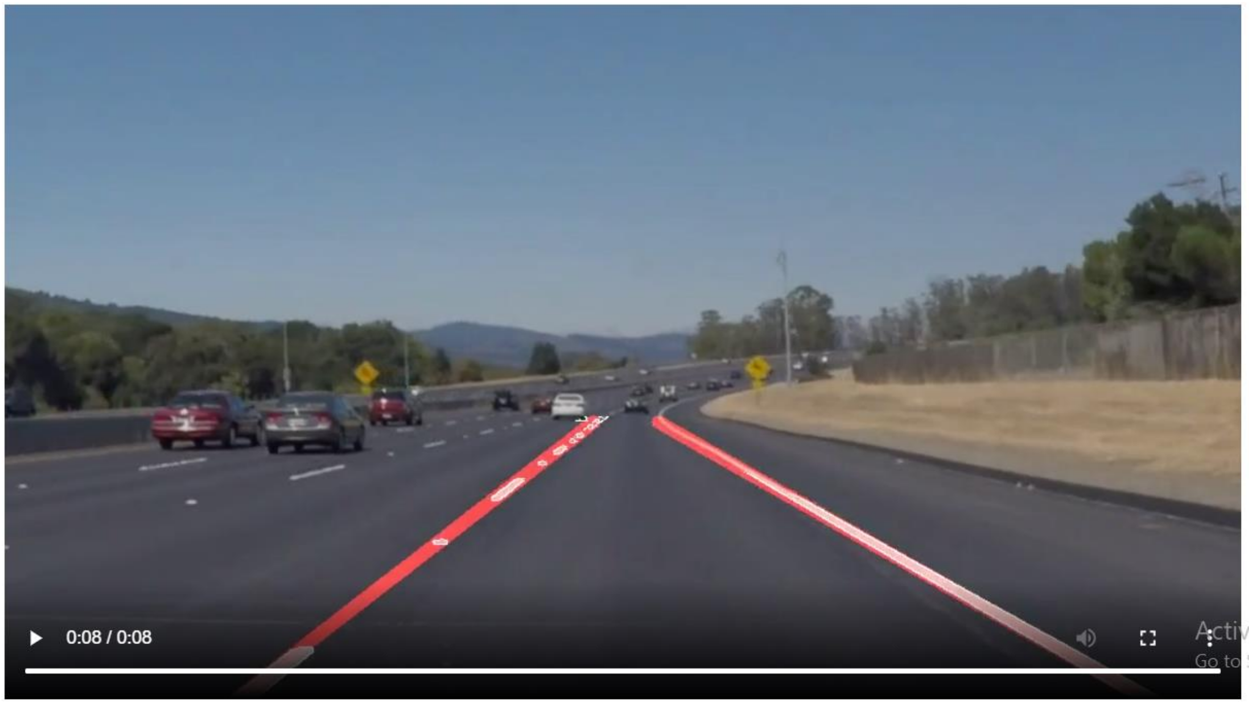
Equation of line definition at google search says:

"In the equation of a straight line (when the equation is written as "y = mx + b"), the slope is the number "m" that is multiplied on the x, and "b" is the y-intercept (that is, the point where the line crosses the vertical y-axis). This useful form of the line equation is sensibly named the "slope-intercept form".

Y: we can decide, x: We have to calculate, m: slope of line, b: constant (y –slope*x1)

So I have written the function where it will calculate left/right lines and used mean to add more weight on longer lines. Also converted the lines into point co-ordinate using make_lines_point function.

Below are the updated results:



So far potential shortcomings with your current pipeline would be:

1. I am not sure if they will work on road with curves since most of the code is hardcoded.
2. Also, since I added all of the code under one function, it can be separated out into different small functions to get more clarity over code review.
3. This technique of lane detection would not work if light from sun or headlight of another vehicle falls on camera of self-driving car.
4. We assumed that, camera is at fixed position.

Possible improvements in this pipeline would be:

1. Can try to explore different color combination of line which will help to understand RGB more.
2. Explore more about gradient function and slope of line which are important factor in deciding extrapolation of images.
3. Can try to implement video on different video clips and analyze the results.

You can check step by step code details at below link:

https://github.com/RITESH-Kapse/Self-Driving-Car-Engineer./tree/master/ComputerVisionFundamentals/ Project 1 - Finding Lane Lines Completed