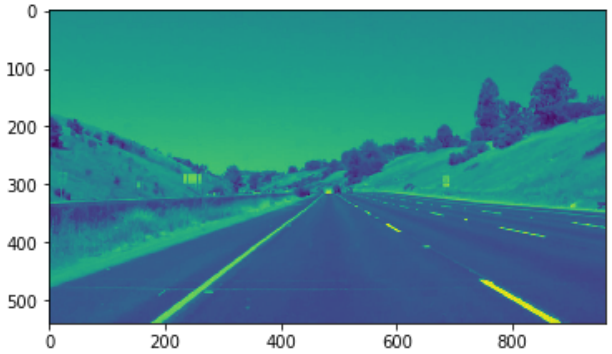
Writeup for P1: Finding Lane Lines

1. Pipeline:

Functions are presented in order. Below image showing the output of each step.

a. <u>Gray scale:</u>

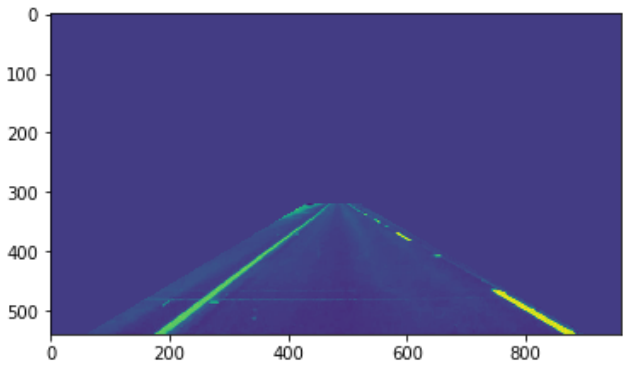
The picture is converted to gray. No modifications here.



b. <u>Region of interest:</u>

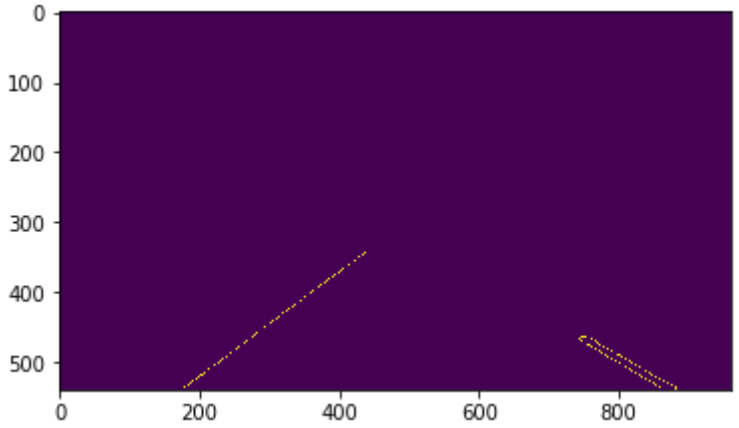
I modified the code for this function in such a way that the background color is the mean value of the color inside the polygon. The purpose of this transformation is to "harmonize the area outside the polygon and ensure a smooth gradient. As can be seen from the output of this function, background color is similar to box's color. This will make the job of Canny easier, because the algorithm is "forced" to focus only on the region of interest.

As approximation, I assume that lanes are symmetrical w.r.t. the vertical line separating lanes. The region of interest is chosen in that regard as can be seen below.



c. Gaussian Blur & Canny transform

No modifications here. The output is provided below.



d. Hough lines (which calles DrawLines):

This is where most modifications happened. HoughLines is not modified, all modifications are in DrawLines. Below is a description of the change.

i. Include vertices as parameter for Draw lines & Hough Lines: The vertices are useful for computing the intersection of the lane lines with boundaries of the region of interest.

ii. DrawLines:

Using the provided vertices, the region of interest can be divided in 2 symmetrical zones (each zone for one lane) defined by the central vertical axis. The changes to this function are:

-Distinguish if lane1 or lane2 by sign of the slope

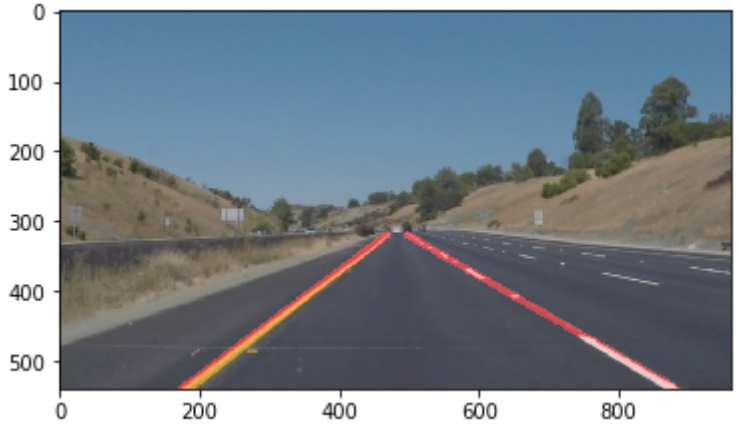
-To improve robustness against wiggling, I include only points which are entirely included in the relative zone.

-Compute an the center point for each lanes by averaging coordinates.

-Compute a weighted average of the slope. The weights are the length of vertice ⇔ The longer the vertices the more confident we're that it comes really from the lane.

-Having the slope and central point we're able to compute the intercept.

-Since we already know the boundaries y-axis value we extract x-axis values.



2. Running videos: Shortcomings & improvements

To run the videos, I first get shape of the images and decide a best possible region of interest. The try different combinations of parameters that provide the best possible result. The challenge is good because it highlights some shortcomings :)

a. **Insights about the challenge**

   -Image size is different which implies a modification of the region of interest

   -Road curve is higher (possibly an illusion due to the different shape). For that reason, I chose a lower upper boundary for the region of interest.

   -The car board is showing in the lower part of the image, therefore, we need to elevate the region of interest a bit.

   -There's an adjacent black car on the top right causing some instability to the gradient, therefore "lifting" the lane up. I also try to solve this by lowering the upper boundary a bit.

b. **Possible improvements**

   -I think the assumption of symmetry with regards to a vertical line is very simplistic. If we imagine a curved road it does not hold anymore. Possible think about a curvature coefficient?

   -Smoothing out objects not having a linear shape such as the car adjacent to lanes that are still in the region of interest.