LANE FINDING PROJECT

Human drivers simply use to their eyes to identify lane of a road and drive accordingly within the constraints of the lane. Similarly, identifying lane is a critical task for an autonomous vehicle to perform in order drive safely and avoid collisions with other vehicles. This is can be achieved by use of effective computer vision techniques. In this project we will cover how to use various techniques with python and OpenCV libraries to detect lane line of the road for the given images.

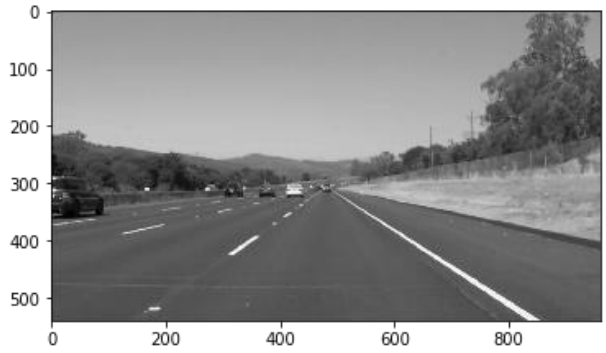
We will build pipeline that recognize lane lines on the roads using the following steps:

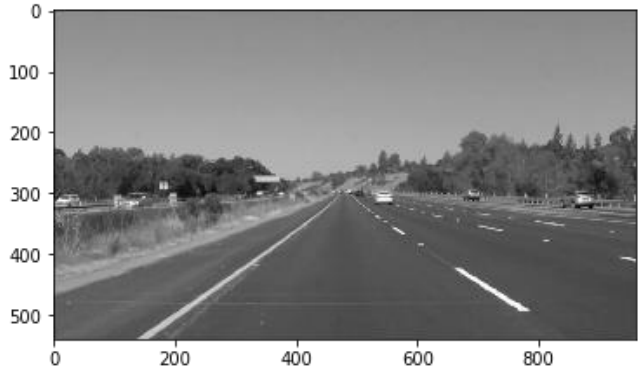
- Convert images to gray scale
- Remove noise from the images
- Detect edges on the images
- Trace region of interest
- Perform a Hough Transform to extract lines
- Separate lines into left and right lines
- Extrapolate and average the lines to create two solid lines for left and right respectively.

Original images

1. Convert image to gray scale

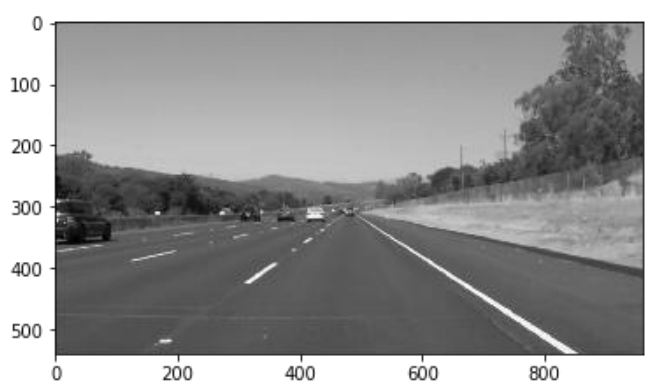
Our original images are 3 channel color images, we need to convert them to grayscale which has only 1 channel with each pixel has only one intensity value from 0-255. By using grayscale image you are processing a single channel which is faster and less computational intensive compared to 3 channel color image.

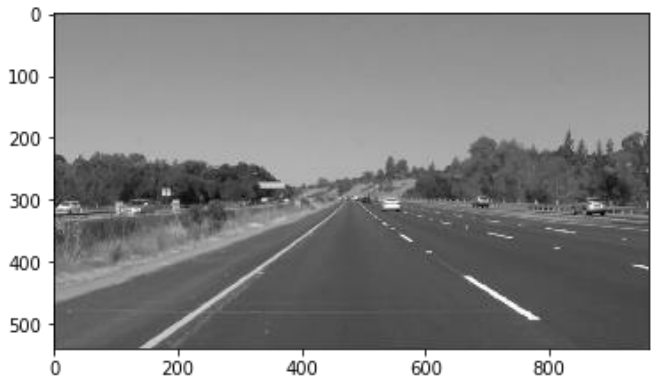




2. Gaussian Blur: Noise removal on images

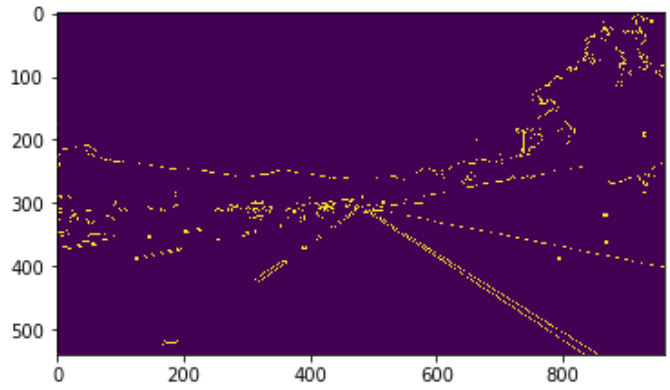
Gaussian blur filter noise and smoothen our images. Image noise can create false and unnecessary edges when edge detection algorithm is applied to the image.

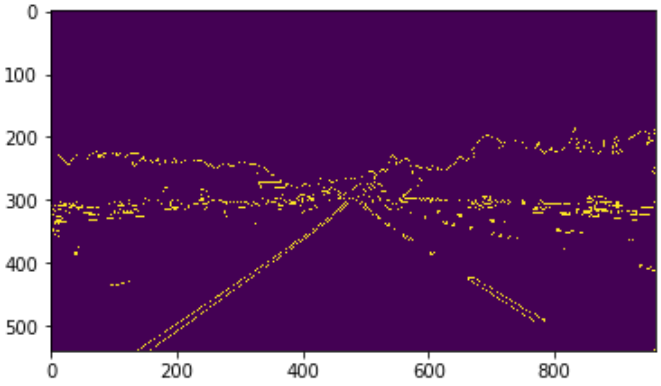




3. Canny edge detection

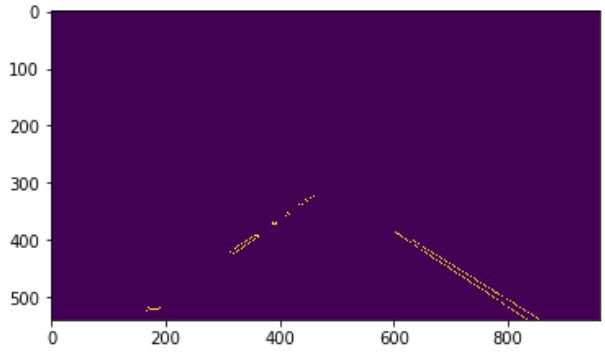
Now that we have preprocessed the images. The next thing in our pipeline is to apply canny edge detection on our blurred images. what canny function does is to perform derivative on our image in both x and y direction thereby tracing outline of the edges that correspond to the most sharp changes in intensity or gradient of the adjacent pixels.

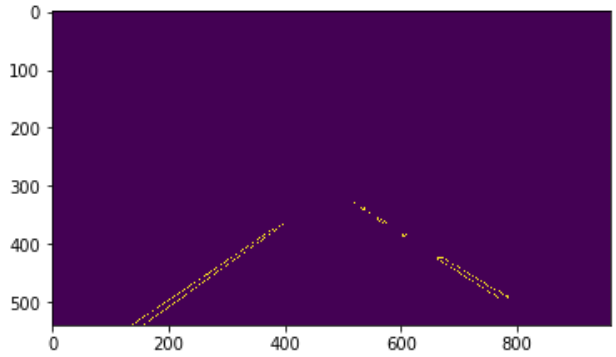




4. Region of Interest

We are only interested in the edges that correspond to lane lines. from the canny image our region of interest look like triangular polygon so we are going to discard any edges outside this polygon.



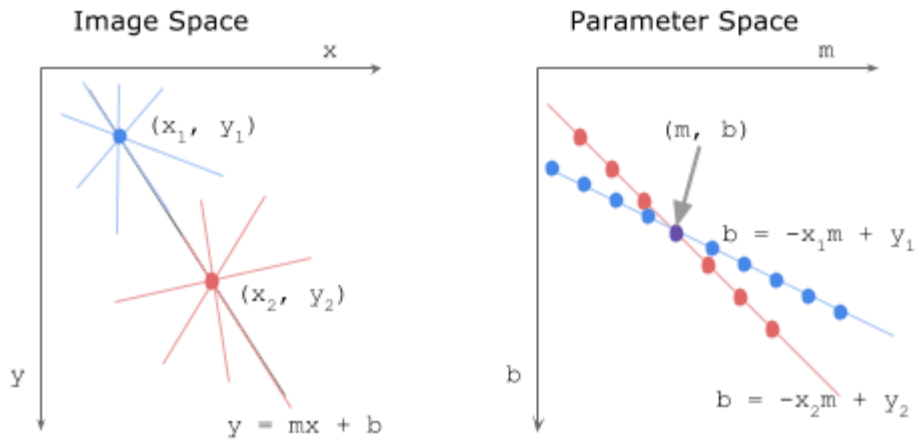


5. Hough Transform

The next step is to apply the Hough Transform technique to extract lines within the region of interest. Given a set of edge points or binary image, we want to find as many as lines that connect these points in image space.

The Hough Transform is just conversion from Image Space to Hough Space. A line in Image Space will be a single point at position (m, b) in Hough Space.

Say, we have 2 edge points (x1, y1) and (x2, y2). For each edge points at various gradients values, we calculate the corresponding b values. The image blow shows that the various lines through an edge point in image space and the plot of these line Hough Space. All points on a line in image space intersect at a common point in Hough space. This common point (m, b) represent the line in image space.

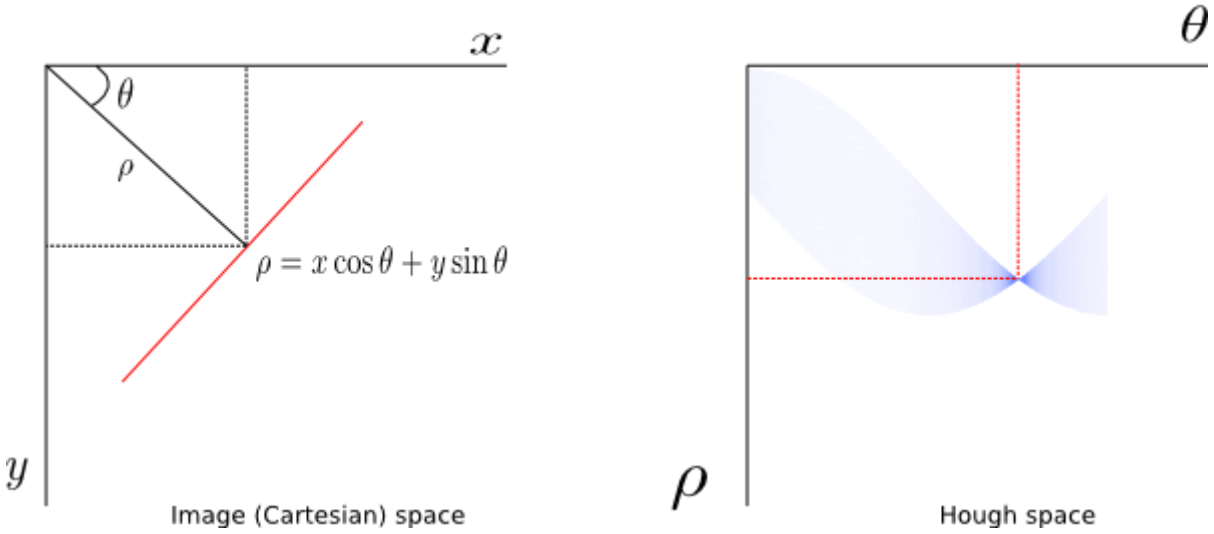


Unfortunately, the slope **m** is undefined when the line is vertical (division by zero). To overcome this, we use another parameter space represent a line polar coordinate. In polar coordinate a line has equation:

ρ = x cos θ + y sin θ

Where ρ represent distance from origin to the line and θ represent angle from the origin to the line.

In this parameter space straight lines going through a given point will correspond to a sinusoidal curve in the (ρ, θ) plane as shown below:

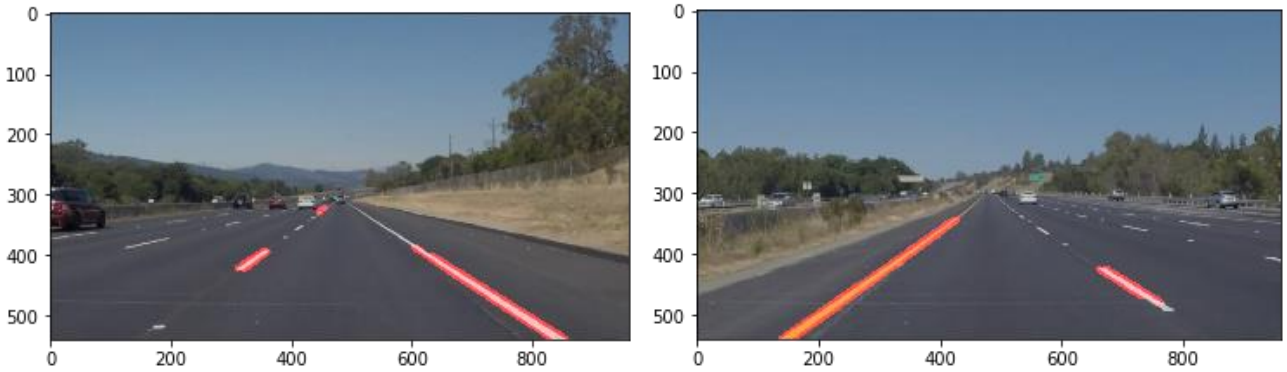


6. Separate Left and Right Lanes

From the high school linear algebra a line which its **x value** increases as the **y value** increases has a positive slope while a line which its **x value** increase as the **y value** decreases has a negative slope.

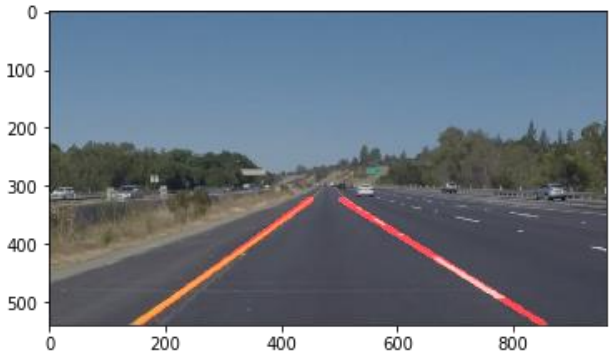
In our image the value of **y** increase from top to bottom. You can see in the left lane **x value** increase as the **y value** decreases while in the right lane which its **x value** increases as the **y value** increases.

So, we will iterate through the lines return by Hough transform marking line with positive slope as right lane and line with negative slope as left lane.



7. Extrapolate and average the lines to create solid lines.

Finally, we will take average of all the slopes and intercepts of the lines for the left and right lines to draw two one solid lines representing left lane and right lane respectively.



Shortcomings

- The pipeline works well on the straight lanes but does very poor on curve lanes
- We used gray image, so change in light like shadow on part of the lane can affect the lane detection.

Future improvement

- We should also consider expressing lines as second degree polynomials to apply for curve lanes.
- To the effect of variance in light in the image, we can applying robust color space such as the HSL color filtering as another pre-processing step.