**Finding Lane Lines on the Road.**

**The goal is to convert the image (first into grayscale) and then finding the lanes on the road as shown below –**

(using OpenCV – computer vision concepts)



**Main steps in the Pipeline –**

1. Converting colored image to **gray** **scale** using grayscale function of OpenCV library
2. Blurring the image using **Gaussian Blur** method of OpenCV library
3. Detecting Edges in the image using **Canny edge detection** approach of OpenCV library.
4. Finding **Region of Interest** using fillpoly method of OpenCV by providing vertices of the region where lines can be found on the image
5. Determining **Hough lines using Houghlinep method of OpenCV library**
6. **Drawing lines** on the original image:

It contains multiple steps-

1. Finding Slope
2. Then adding vertices of line whose slope is positive to the right line and others to the left line.
3. Also Filtering out the vertices or the lines whose slope absolute value is less than 0.5 value since the lane markings are made in such a way that the slope is greater than 0.5
4. Making sure the length of left and right lines is greater than 0
5. Setting y max and y min –
   1. min\_y = int(img.shape[0] \* .60)
   2. max\_y = int(img.shape[0])
6. Fitting the left and the right line points to the line and finding the top and bottom x vertices of line using poly1D function of numpy.

Once we get these x and y values, we can draw a line using

cv2.line(img, (left\_x\_top,max\_y),(left\_x\_bottom,min\_y) ,color, thickness)

cv2.line(img, (right\_x\_top,max\_y),(right\_x\_bottom,min\_y) ,color, thickness)

where Left\_x\_top, Left\_x\_botton, right\_x\_top, right\_x\_bottom are basically the points we got after using Poly1d function (after fitting the line – using polyfit)

left\_fit = np.poly1d(np.polyfit(left\_y, left\_x, deg=1 ))

left\_x\_top = int(left\_fit(max\_y))

left\_x\_bottom = int(left\_fit(min\_y))

right\_fit = np.poly1d(np.polyfit(right\_y, right\_x, deg=1 ))

right\_x\_top = int(right\_fit(max\_y))

right\_x\_bottom = int(right\_fit(min\_y))

**2. Potential shortcomings with the current pipeline –**

Need to come up with the better model for averaging of lines – may be smoothing it in a better way either by finding the moving average of the lines by assigning more weight to the more recent points( still not sure how to implement this – but will try this later on) so that it can perform on the curvy roads also.

There might be some other approach to make it work on the sharp turns.

**3. Possible improvements to your pipeline**

There is still lot of improvement required in averaging of lines and finding the exact parameter for creating Hough lines to be able to achieve a perfect non-wobbly line.

May be using some other approach to identify the lanes (deep learning – not sure though).

There might be lot of curvy roads so averaging of lines might not work for sharp turns, need to figure out some other way to be able to capture the sharp turns and curvy roads.