Finding Lane Detection

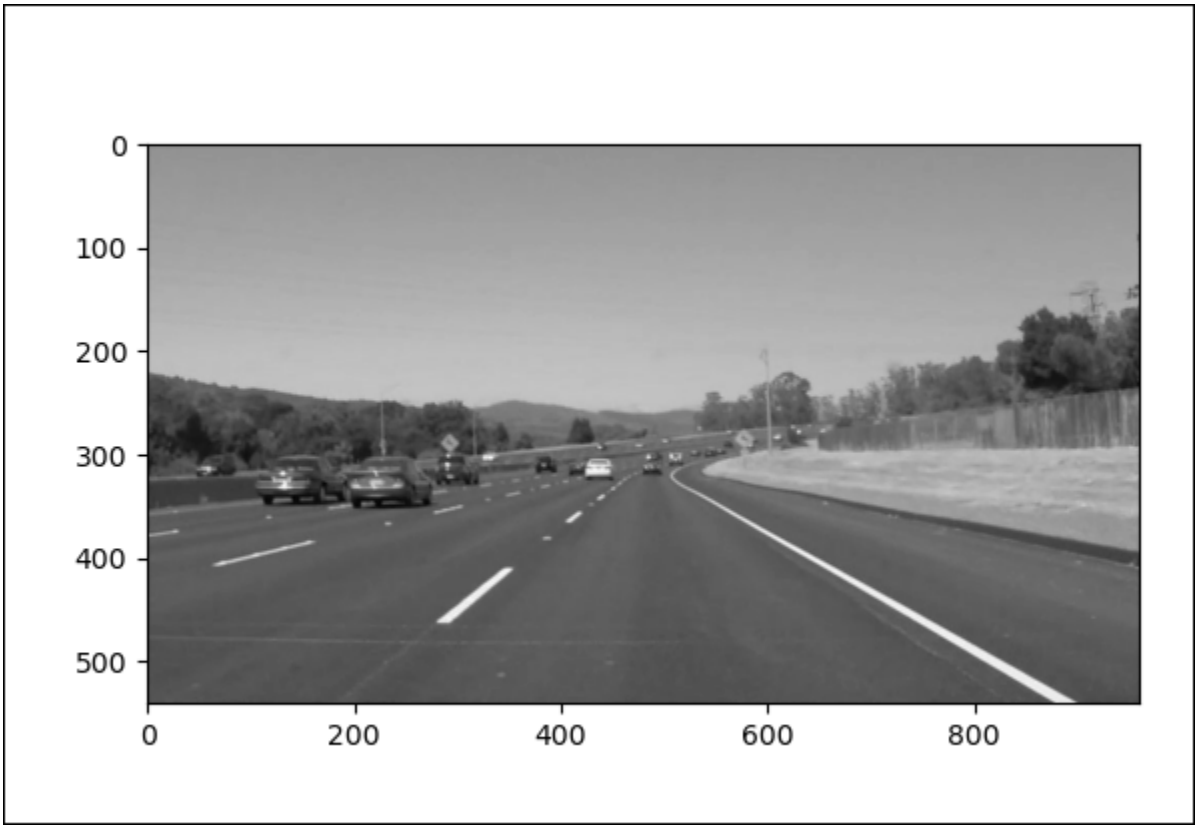
Description: - In this Project, the idea is to use Open CV library and find a lane with continuous lines.

Please use below table to navigate to different section

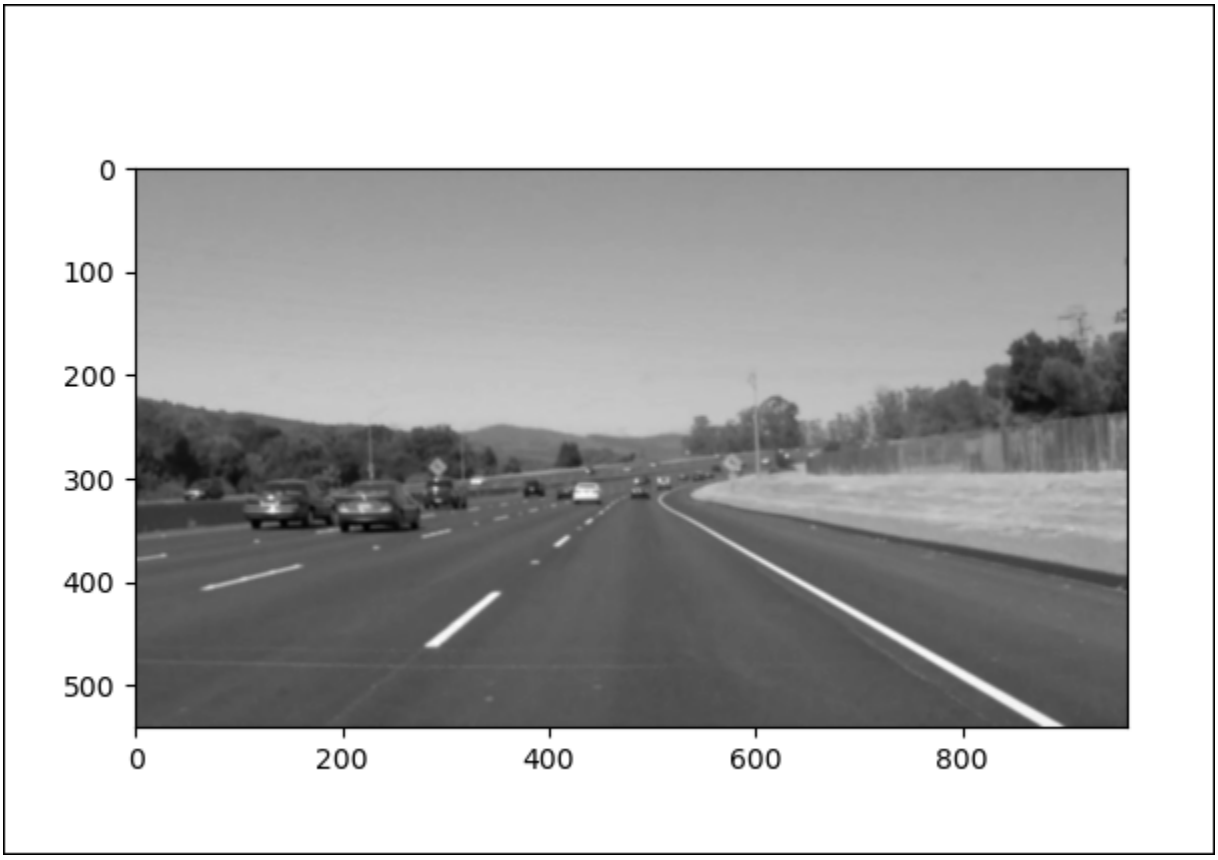
Pipeline Description: -

The Pipelines consist below steps

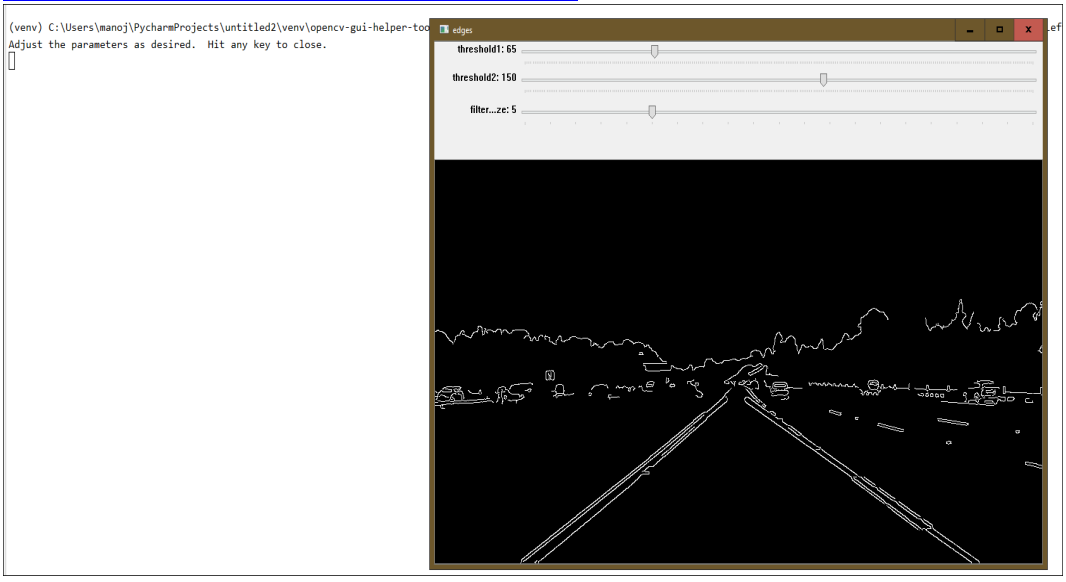
- Reading the images from the folder and getting the shape of the image
- Converting the image to **grayscale** images

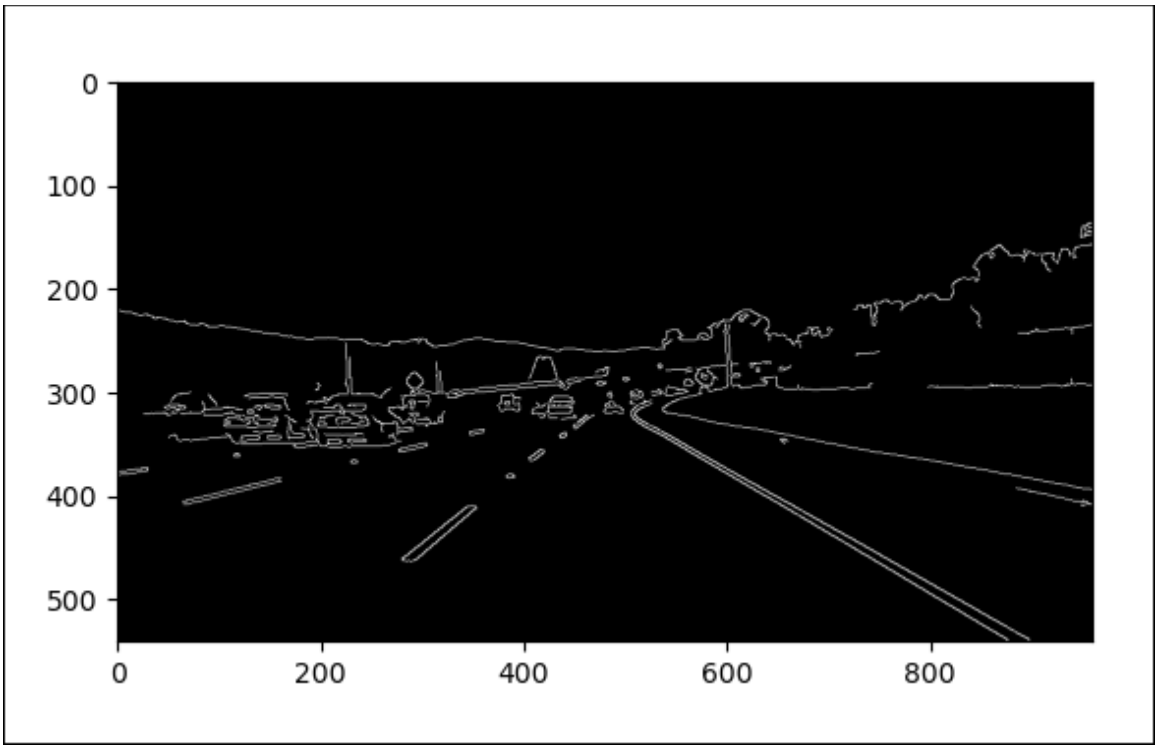


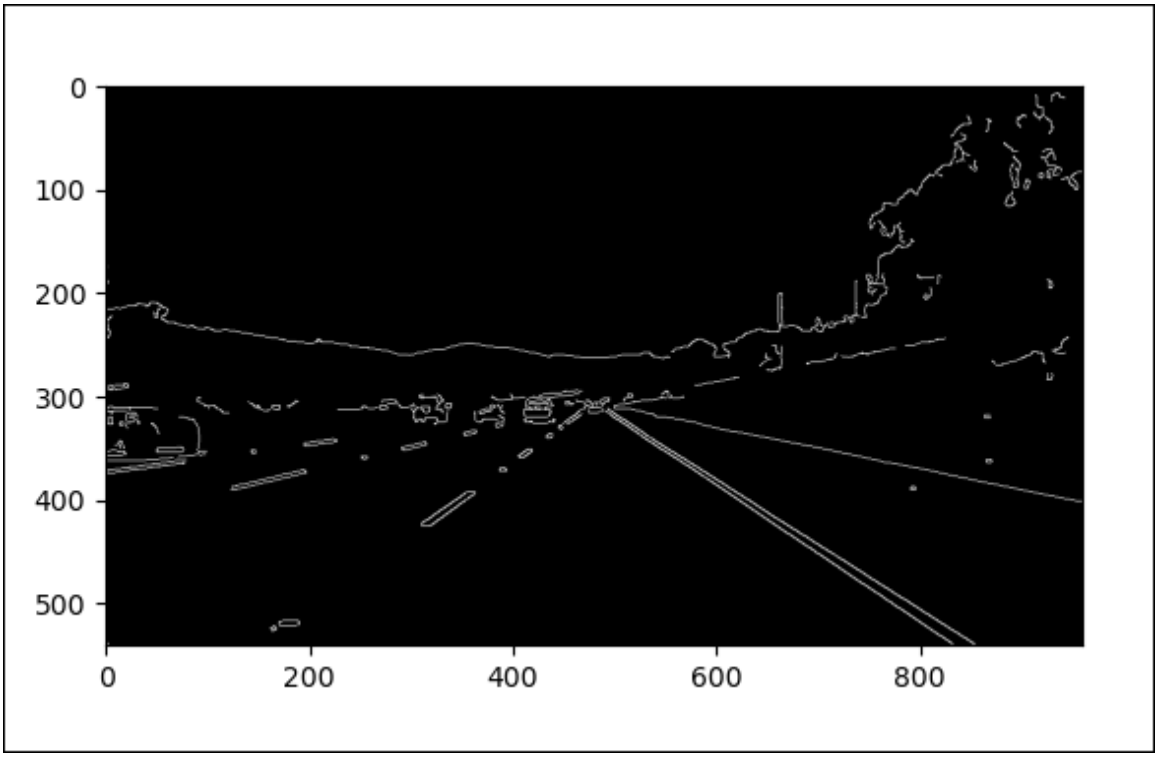
- Define the gaussian kernel size (used 5) and convert the gray image into **gaussian** image

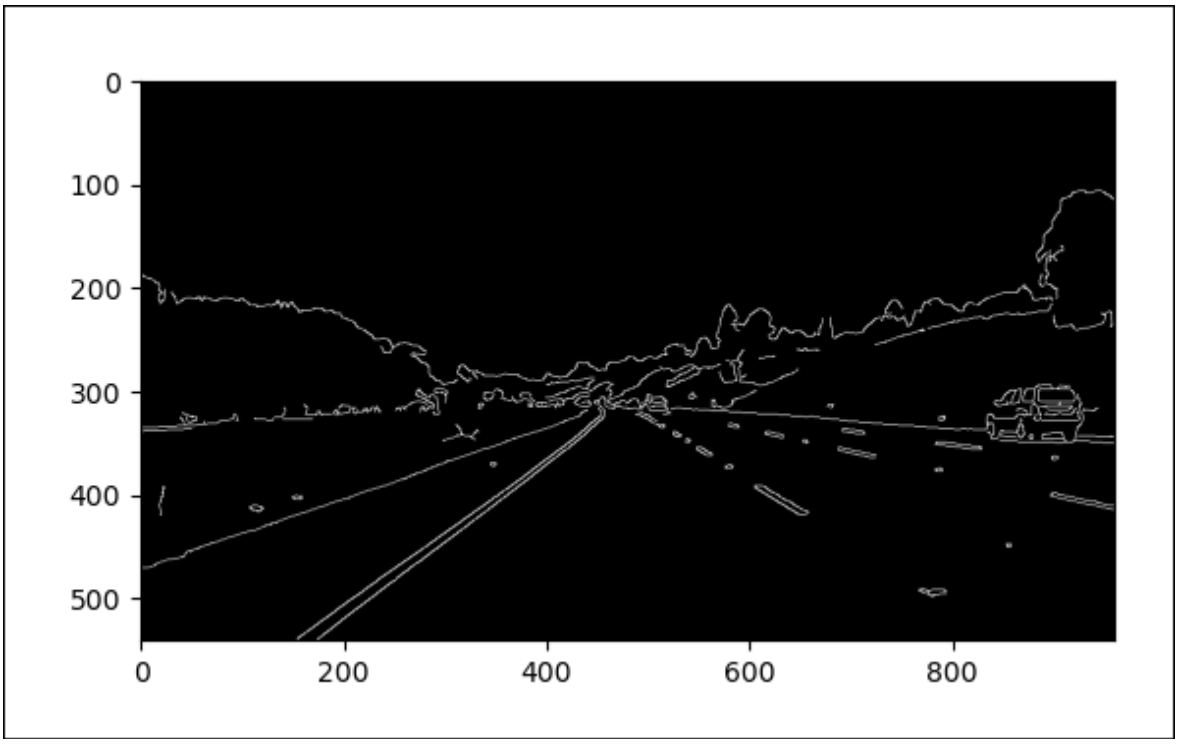


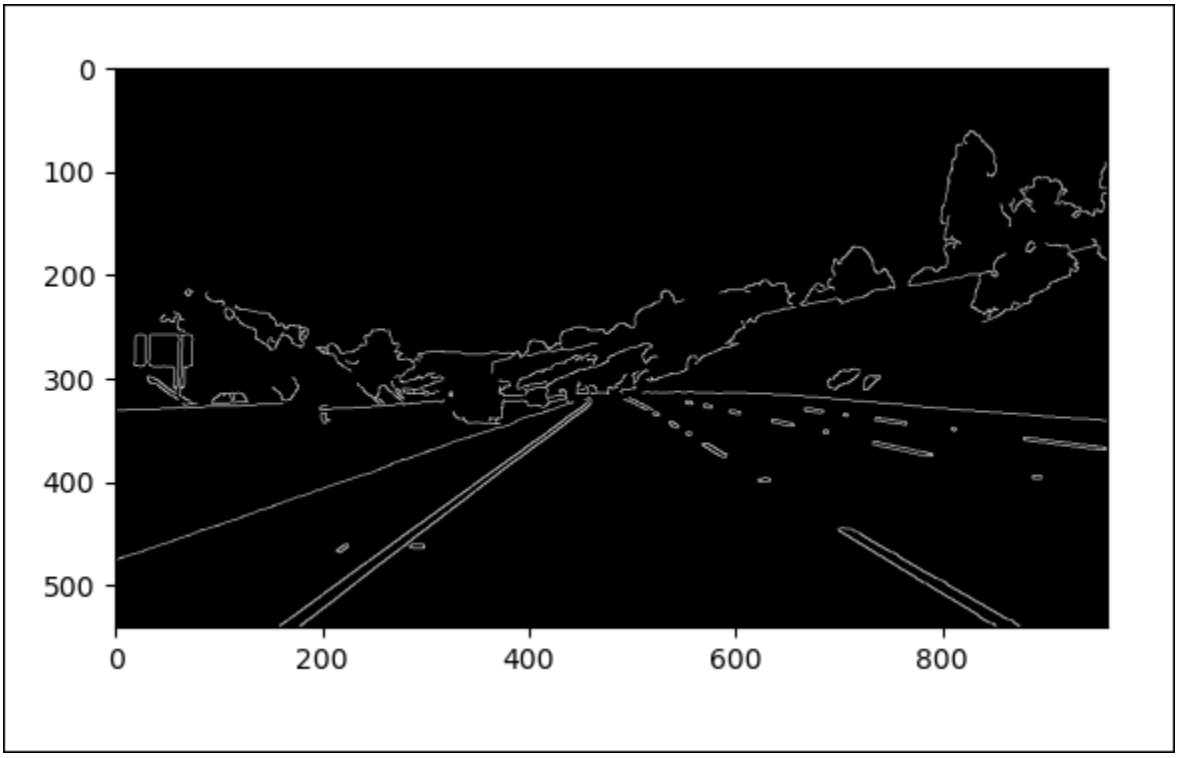
- Apply the **canny edge detection algorithm** on the previous image. To get the minimum and maximum threshold quickly I have used below algorithm to identify the min and max threshold
  https://medium.com/@maunesh/finding-the-right-parameters-for-your-computer-vision-algorithm-d55643b6f954
  https://github.com/maunesh/opencv-gui-parameter-tuner

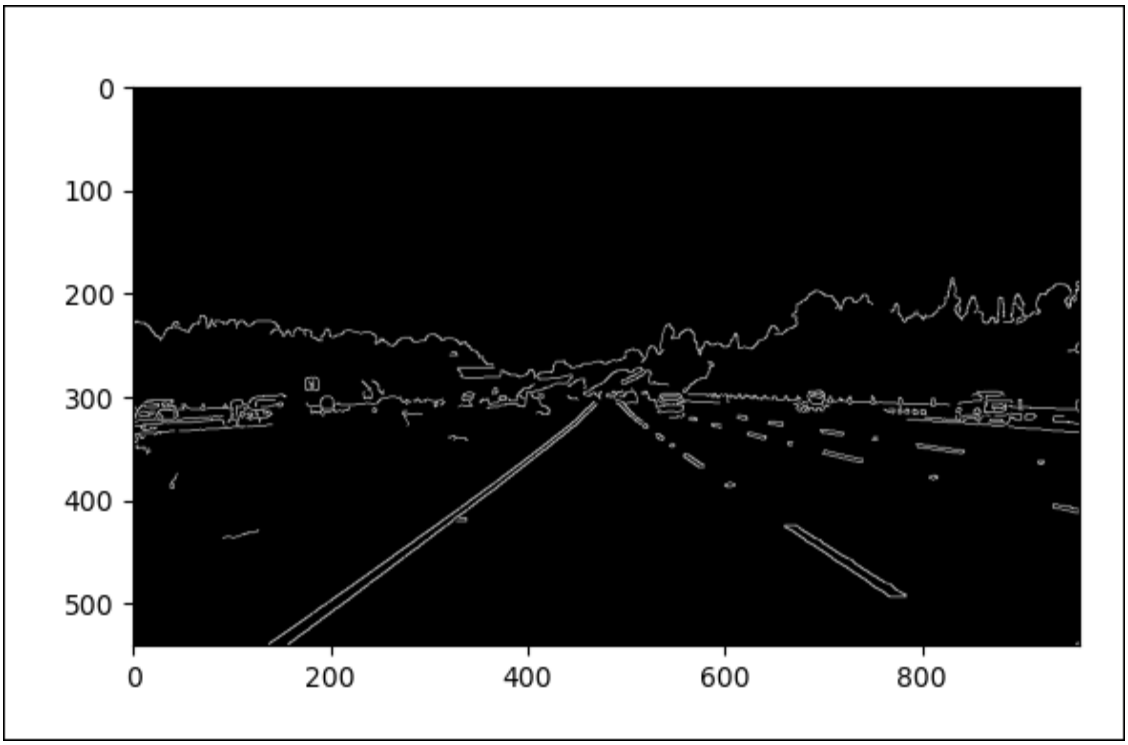



0
100
200
300
400
500
0
200
400
600
800

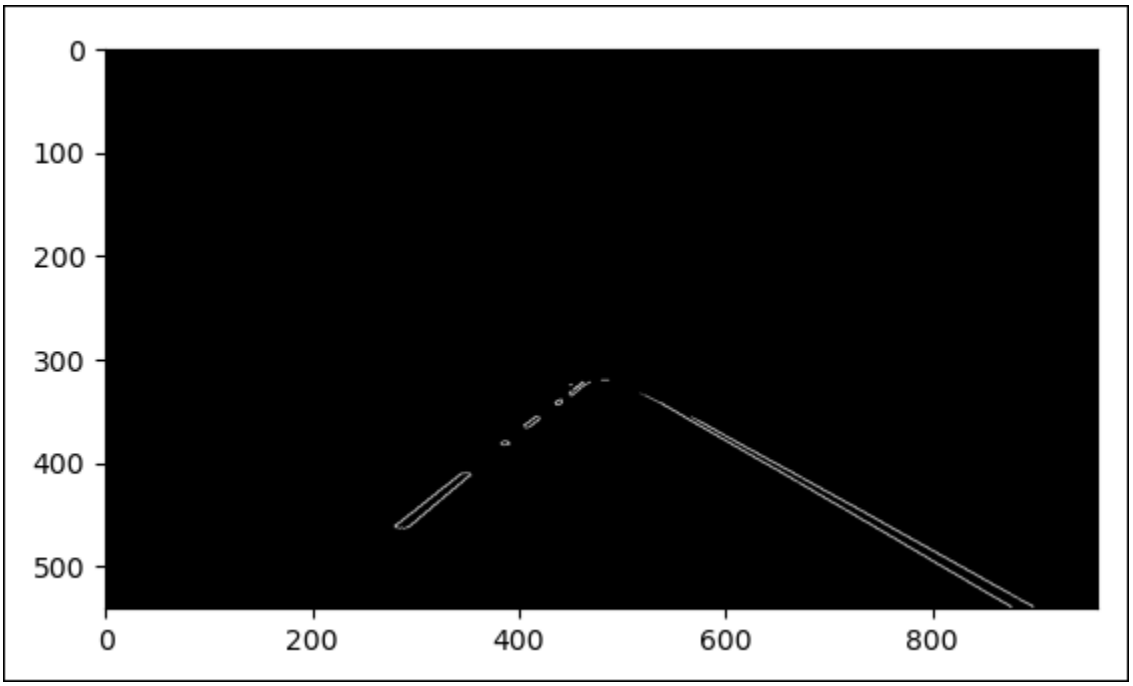

0
100
200
300
400
500
0
200
400
600
800

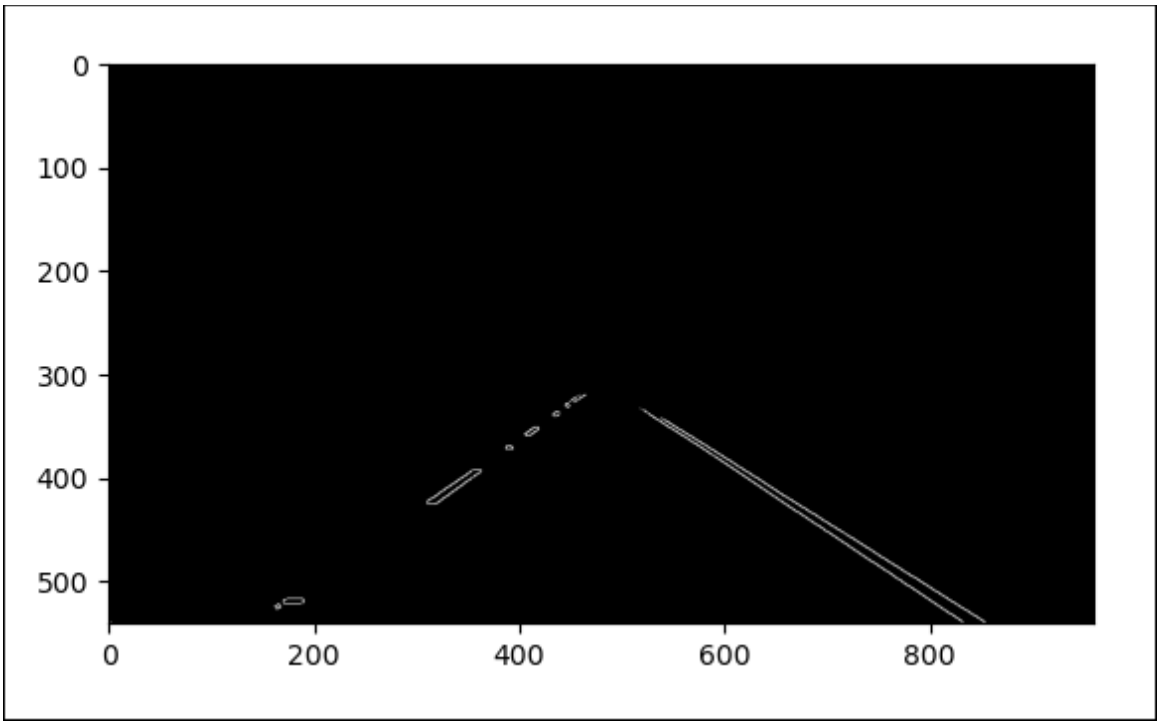

0
100
200
300
400
500
0
200
400
600
800

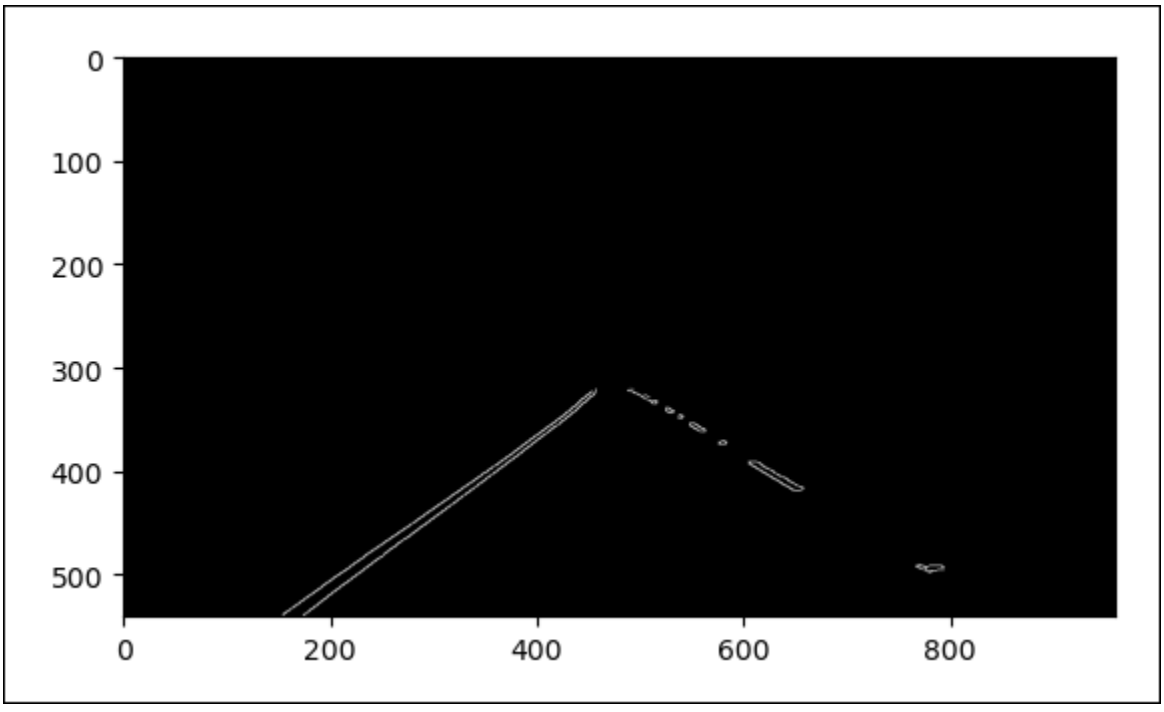

0
100
200
300
400
500
0
200
400
600
800

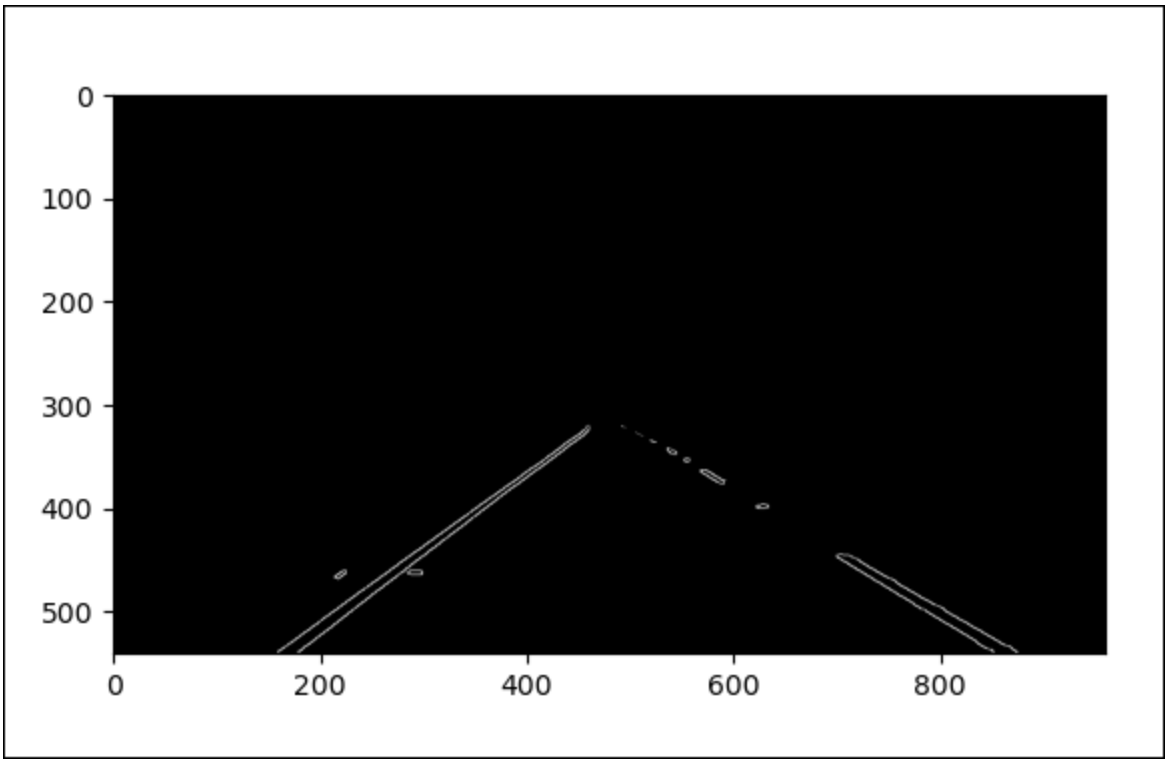


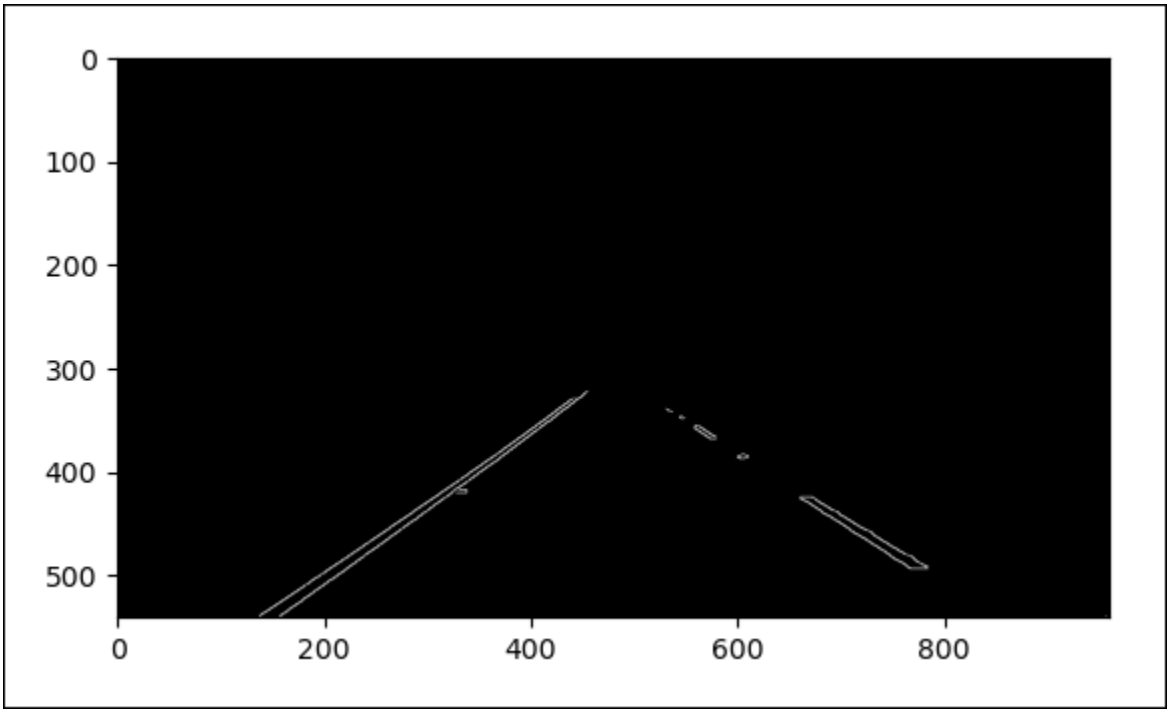
- After getting the edges in the images the next step is **to define the area or region** where the lane lines exist and masking the other area. For area I have defined 4 vertices of the polygon which covers the lanes in the images

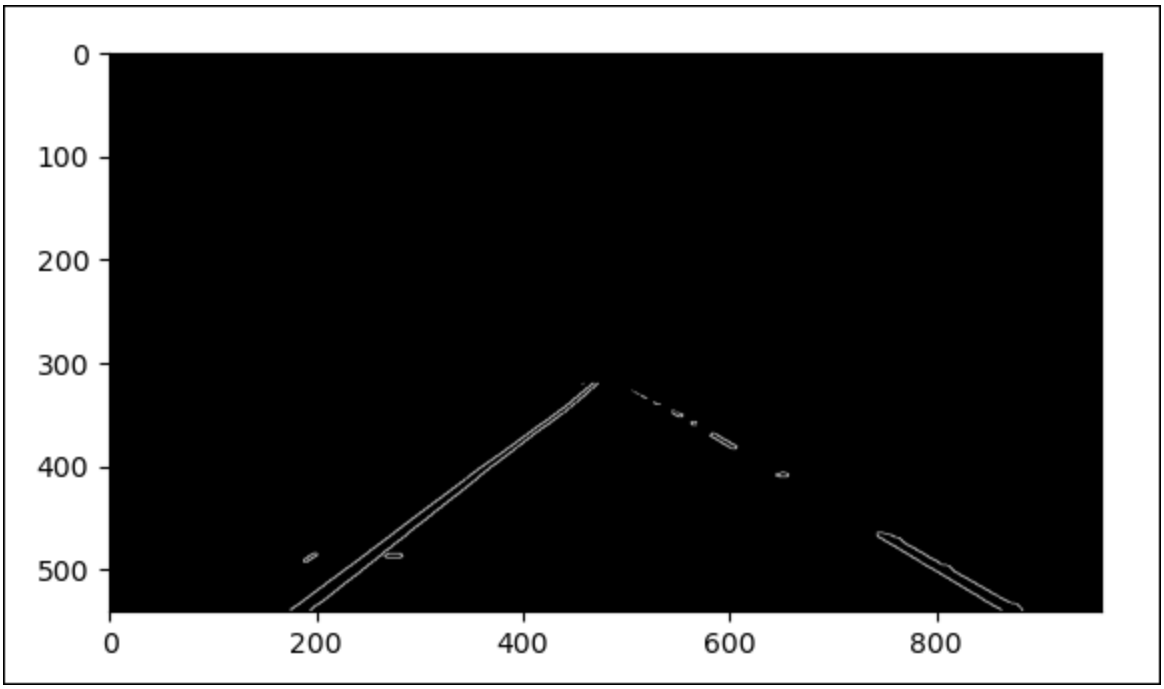


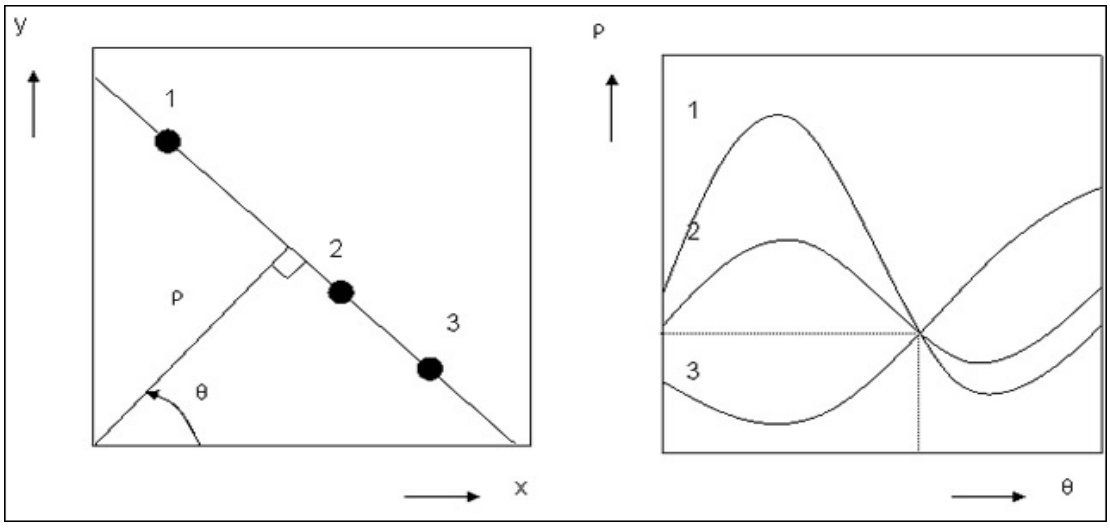

0
100
200
300
400
500
0
200
400
600
800


0
100
200
300
400
500
0
200
400
600
800

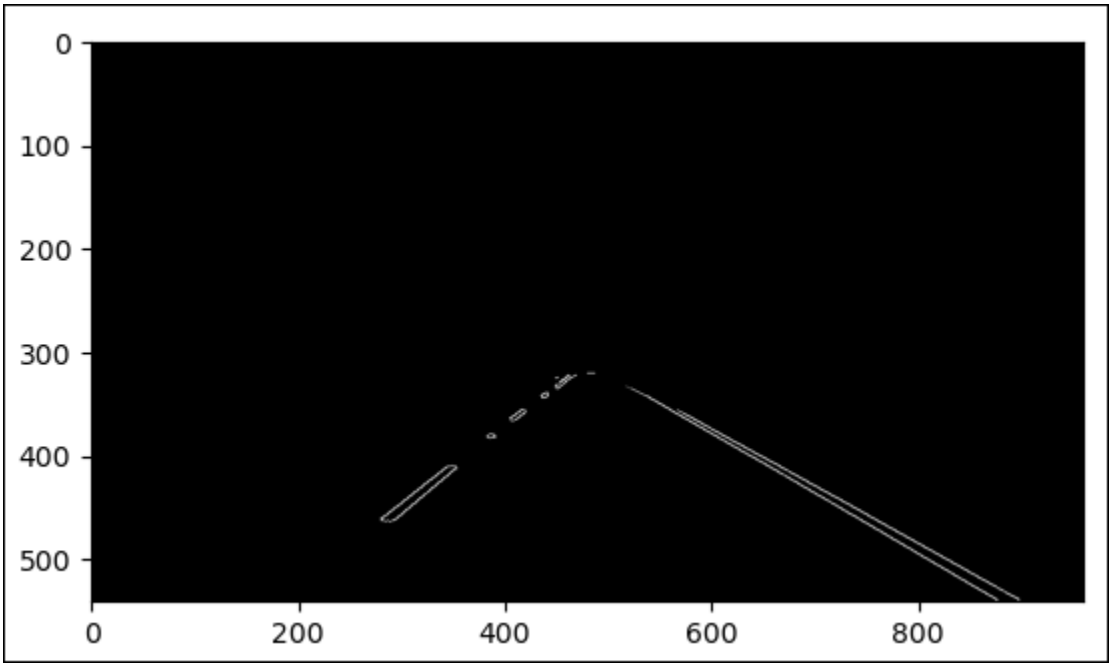

0
100
200
300
400
500
0
200
400
600
800

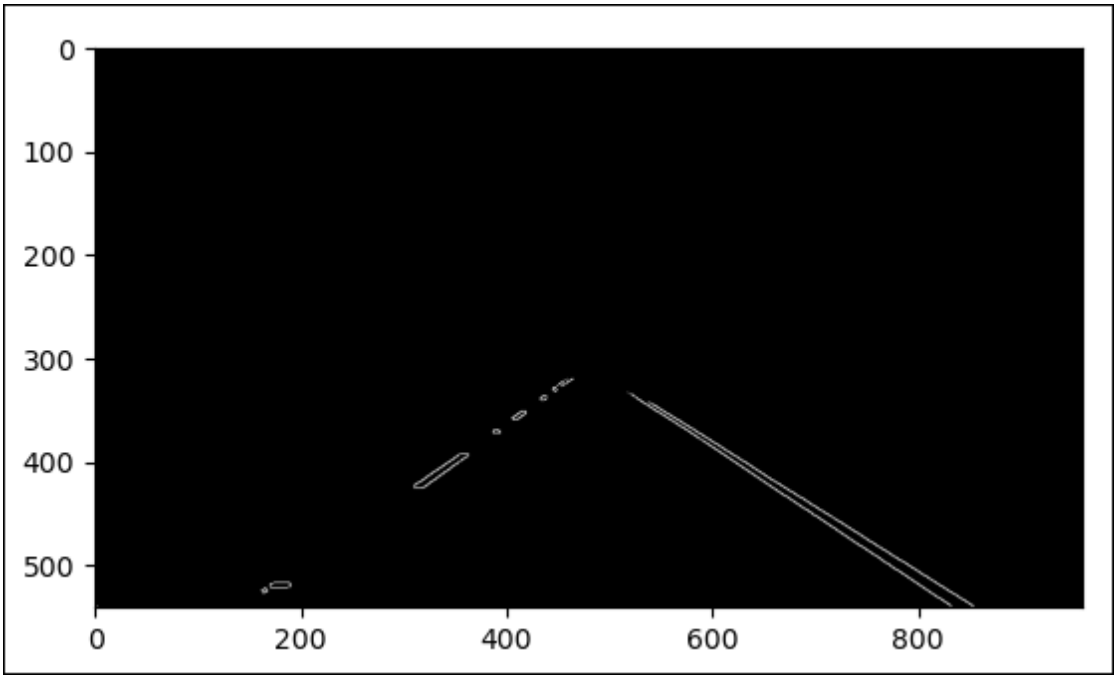

0
100
200
300
400
500
0
200
400
600
800

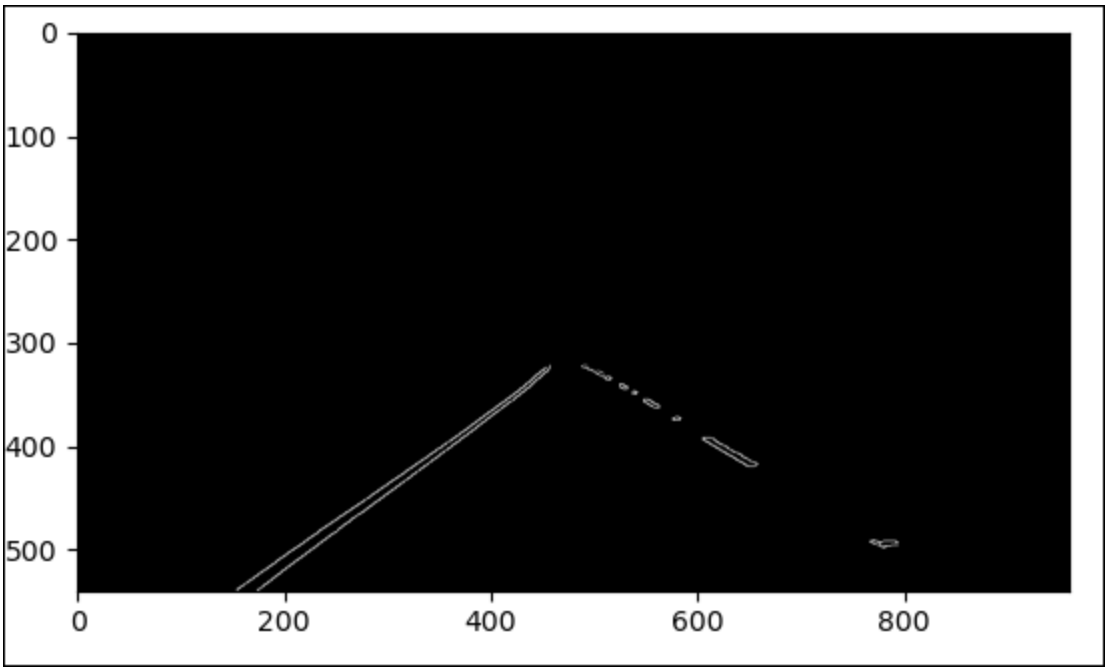


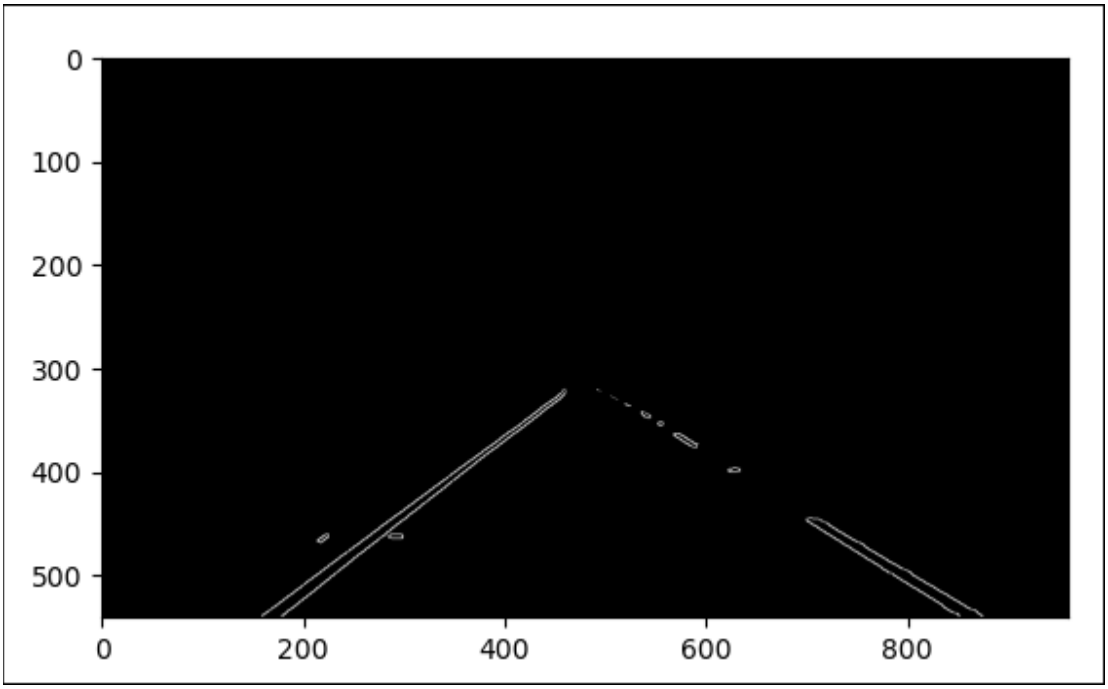
- Then using theses vertices and images from Canny Edge detection algorithm to get the region and mask the remaining pixel in the images
- In the next Step use **Hough Transformation** and tried with different values for the parameter to get a clear line.
    - Hough Transformation is the technique to get all the lines by identifying all the points on the lines
    - 

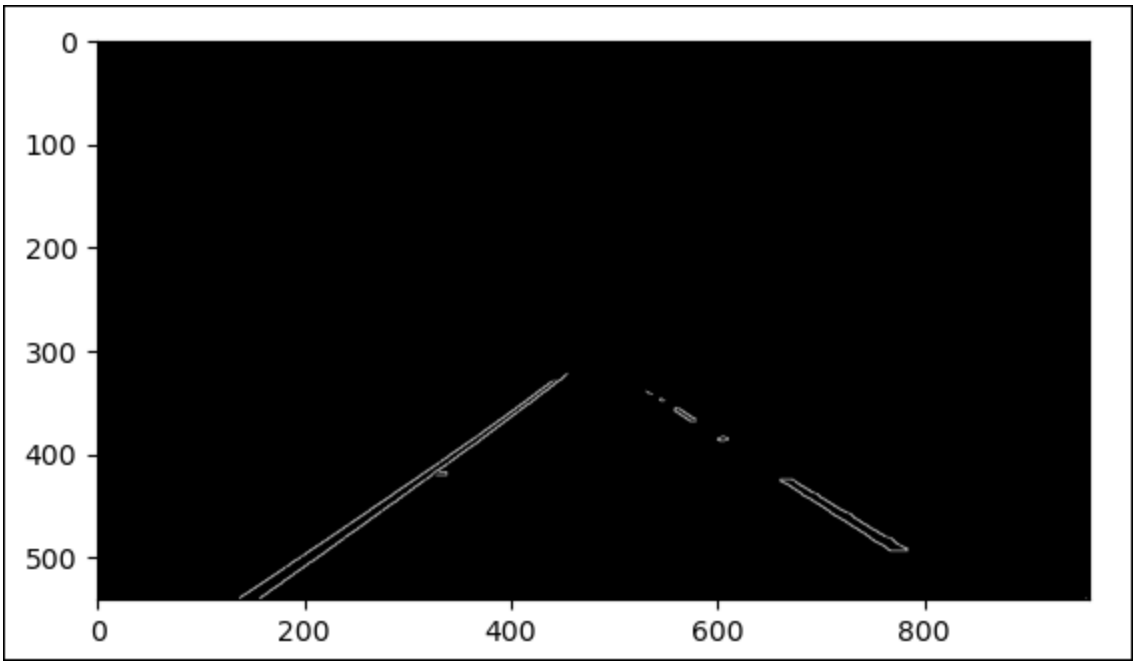
Further Information
https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html

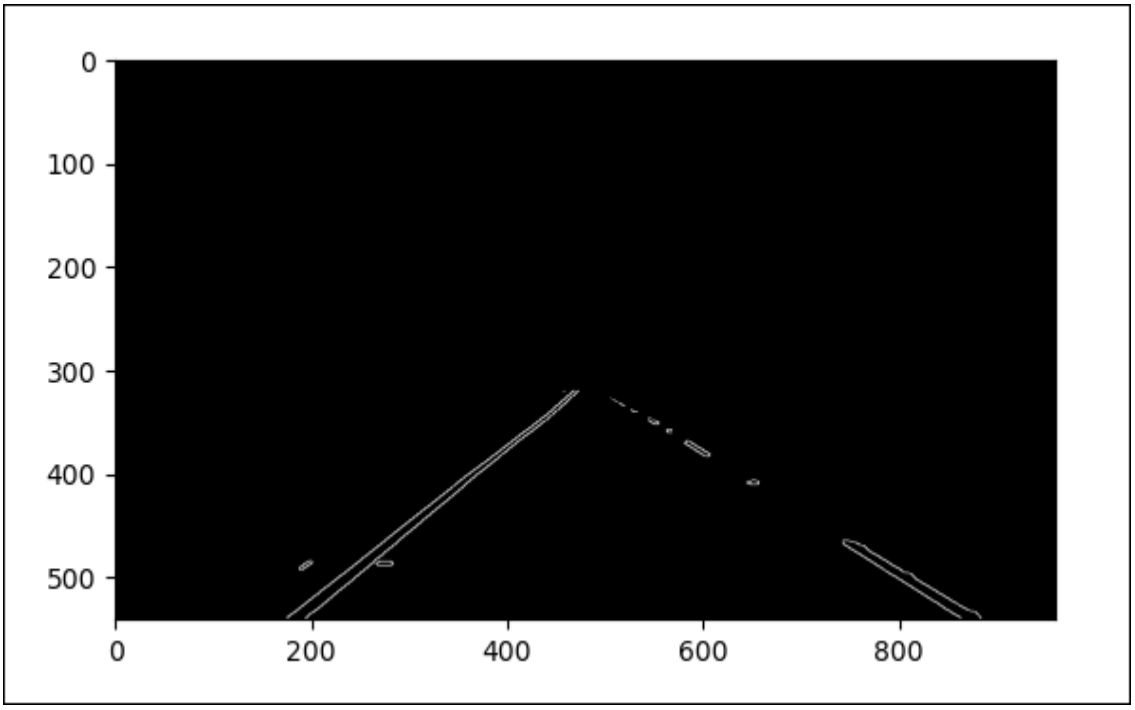




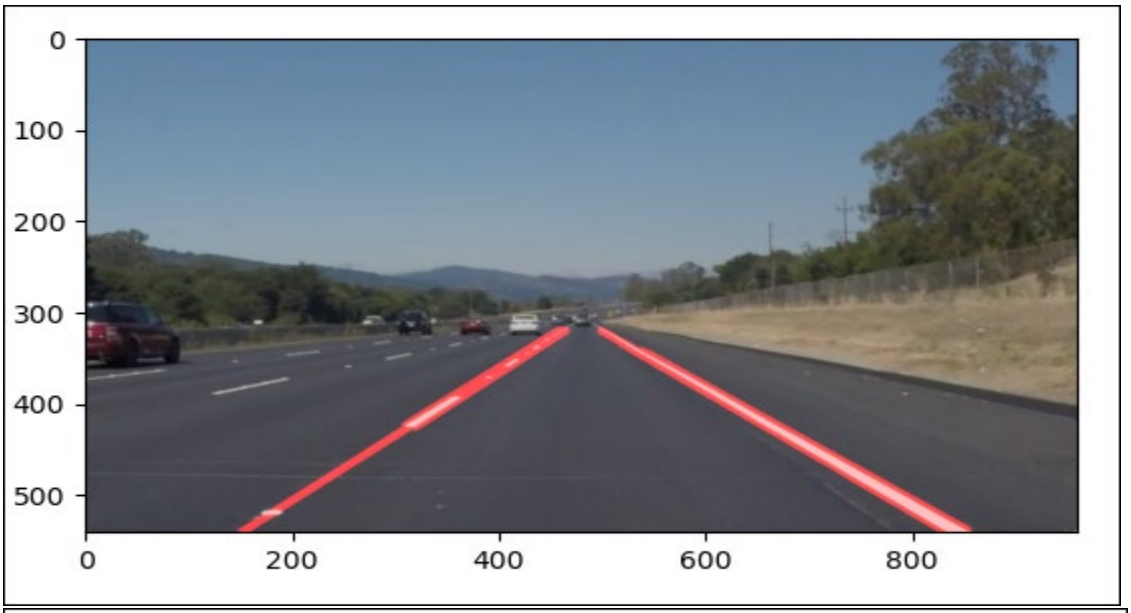

0
100
200
300
400
500
0
200
400
600
800

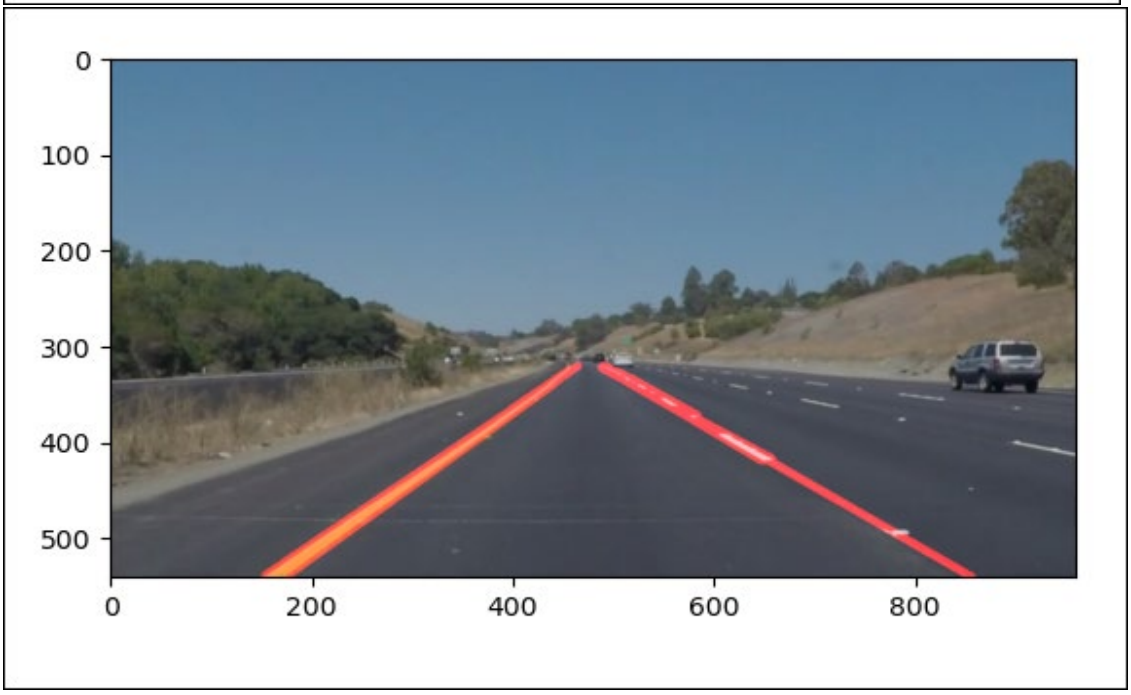

0
100
200
300
400
500
0
200
400
600
800

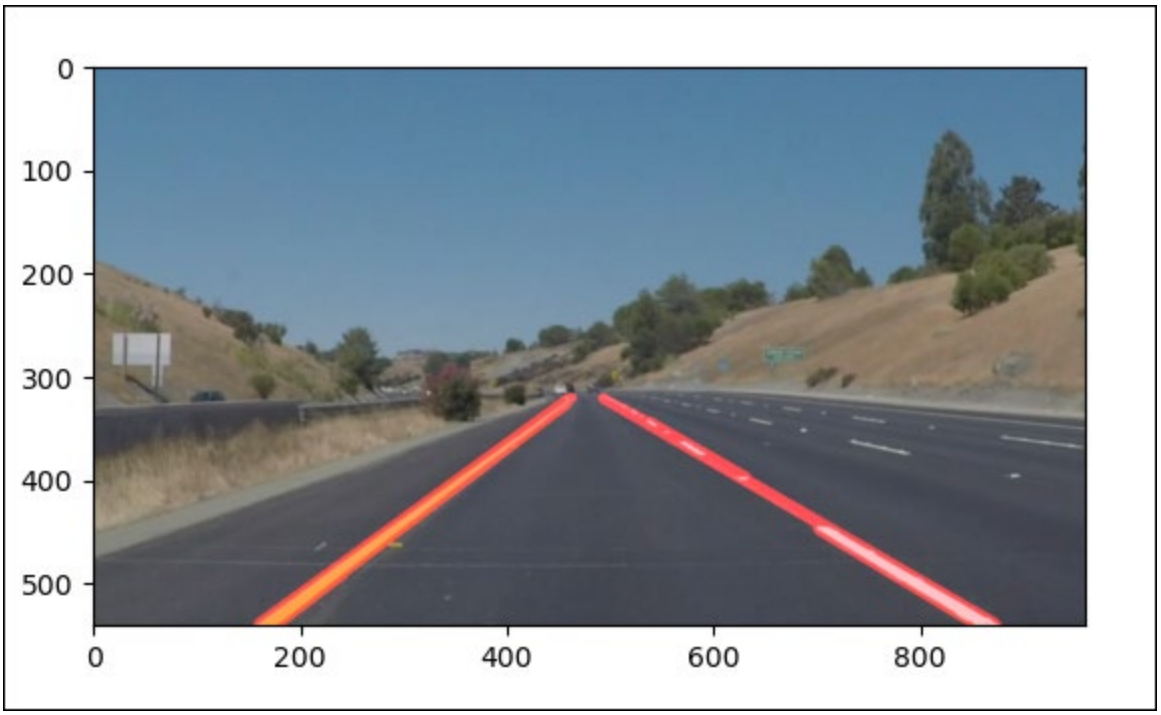

0
100
200
300
400
500
0
200
400
600
800

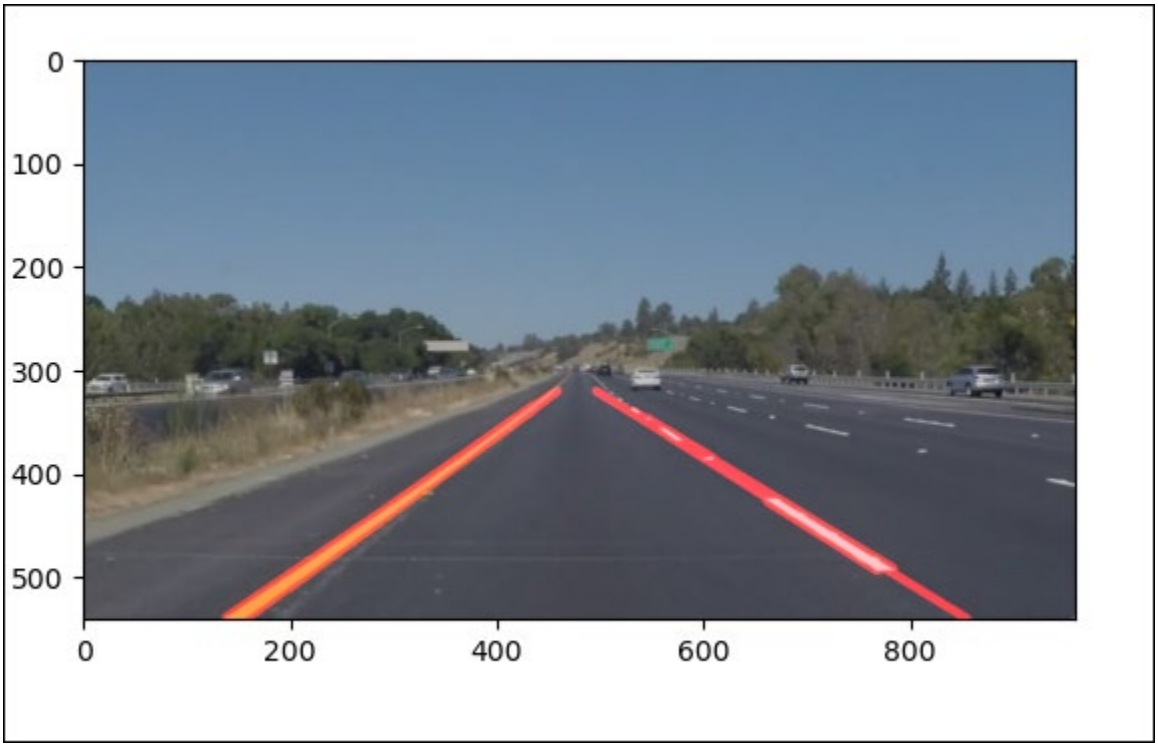

0
100
200
300
400
500
0
200
400
600
800

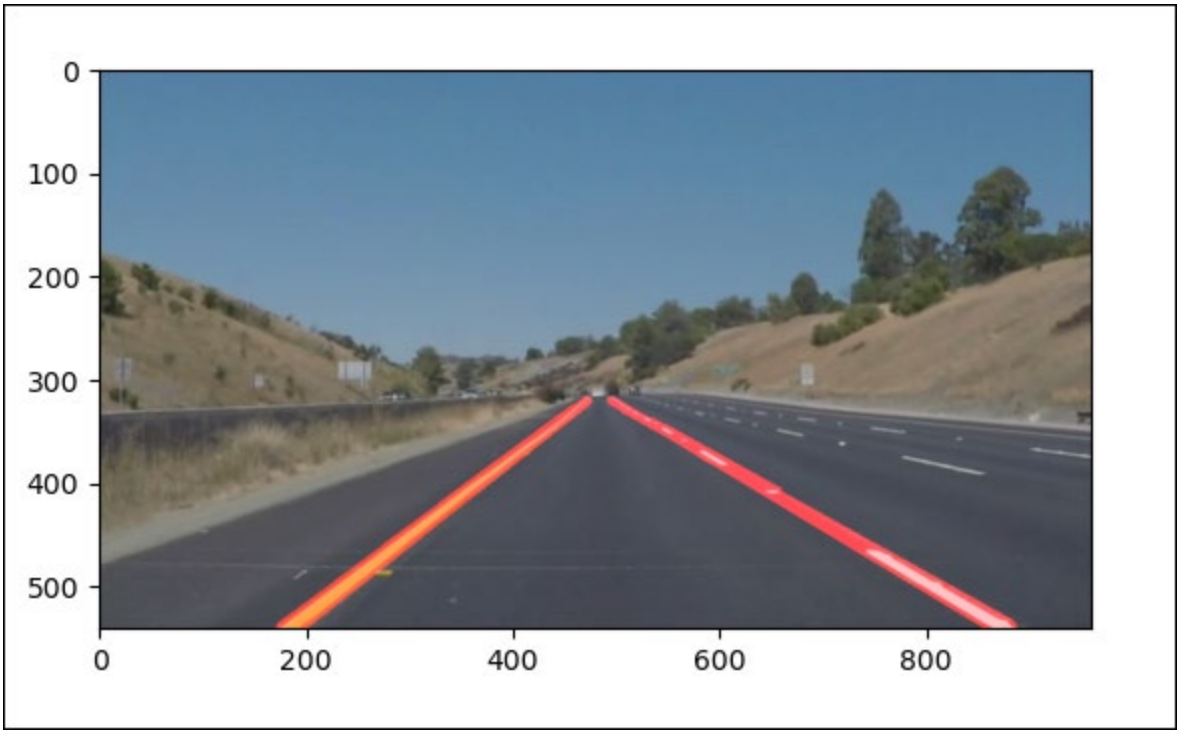
- After that using weight function to draw line and saving the output to a directory


0
100
200
300
400
500
0
200
400
600
800


0
100
200
300
400
500
0
200
400
600
800

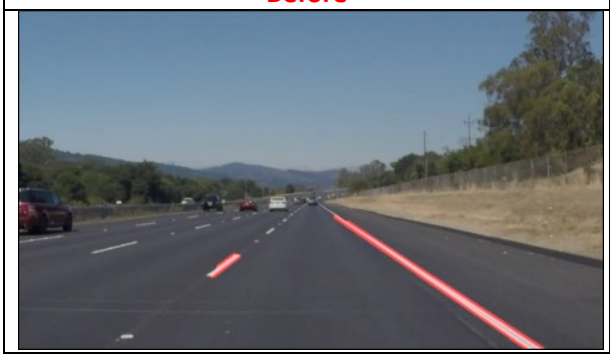

0
100
200
300
400
500
0
200
400
600
800


0
100
200
300
400
500
0
200
400
600
800



Extrapolation and Drawing the lines to get the complete lane:-

- Modifying *draw_lines()* function to extrapolate and consolidate the lines
- The gives function needs to be modified since the output lines need to be continuous and need to be extrapolated.
- Below is the difference of the Image

Before	After
	

- As you see in the above left image (Dotted Line Lane) we need to draw the complete line for the left lane.
- For this we must divide both side of the line segments into two different list and plan to add them left or right side, respectively .
- Looking at the lines in the image it is evident that both lines will have opposite slope to each other.

- We will separate the Left line and put it in a different list and doing similar for the right side.
- There will some slope which will outliers (i.e. m ≥ ±1) and will ignore them
- To make our lines segment more precise we will take average for Slopes and intercepts
- Then will use CV2.line method to draw the complete line for both left and right side

Marking Lines on the Video:-

Use the same pipeline code for the video to draw the line on the lane marking

```
Video 1: solidWhiteRight.mp4
```

In this video the code works well, and we can see complete lines are drawn in the video for both left and right line.

```
Video 2: solidYellowLeft.mp4
```

For this video also the code works well, and we can see complete lines are drawn in the video for both left and right line.

```
Video 3: challenge.mp4
```

After applying the code, the output of the video does not seem correct. The lines are not correctly drawn on the marking lines .

Shortcomings of The Pipeline:-

We have made lot of assumption for this pipeline and lot of shortcomings for the code as below

1. As we can see from the output of the last video that it doesn't work when the road is not a straight line both Canny Edge Detection and Hough Transformation uses the straight line to draw the marking the pipeline fails.
2. For Canny edge finding the minimum and Maximum Parameters values by trial and error is still good , but Hough Transformation we have 5 parameters and to get the correct set of values that give the correct output is time consuming.
3. We are assuming here there is no sharp turns . In case of sharp turn or circular turn the pipelines will not work properly
4. In Case of bad weather condition (i.e. Extreme Rain and Snow on the Road) the marking will be not visible, and the pipeline will fail.

5. In case where the Lane marking is missing the pipeline will not work properly and may not even work since there will be no distinction on the road and line marking.
6. If there is Uneven path the pipeline will not work properly since extrapolation technique would not able to draw the correct lines

Future Improvements:-

1. Using a transformation that can handle both Straight and curved roads and extrapolate the region of driving of the Vehicle.
2. As we are heavily relying on the Camera Images , it needs to verify for the correctness to correctly draw the lines and identify object on the line.
3. Rather Than trial and error for Finding the right value for the parameters for the Transformation it should be automated