P1: Finding Lane Lines on the Road

Reflection

Pipeline Description:

My lane detection pipeline is consisting of different steps and below I have mentioned all these steps in details.

Step 1:

In the very first step, I converted all input images into grayscale images using given grayscale() helper function.

Output:

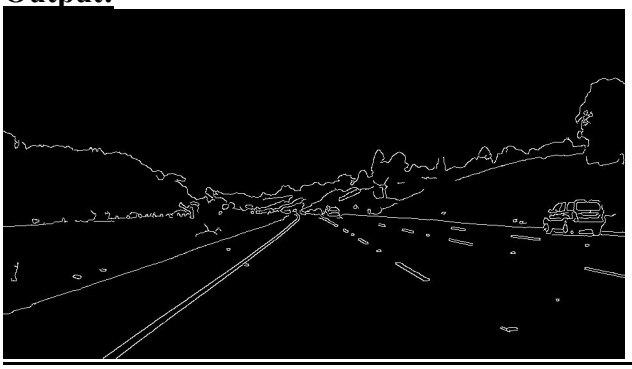
Step 2:

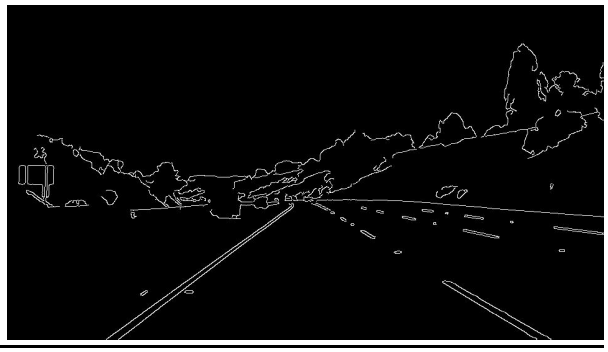
In next step, I applied Gaussian blur function in order to remove noise from grayscale images. I have applied the Gaussian function from OpenCV having a kernel size of 5.

Step 3:

After removing noise from given images, I applied Canny Edge detection to detect all edges in each picture. I have used Canny edge detection from helper function which is having low and high threshold of 50 and 150.

Output:

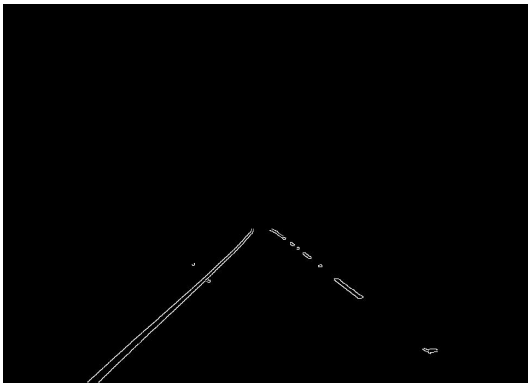
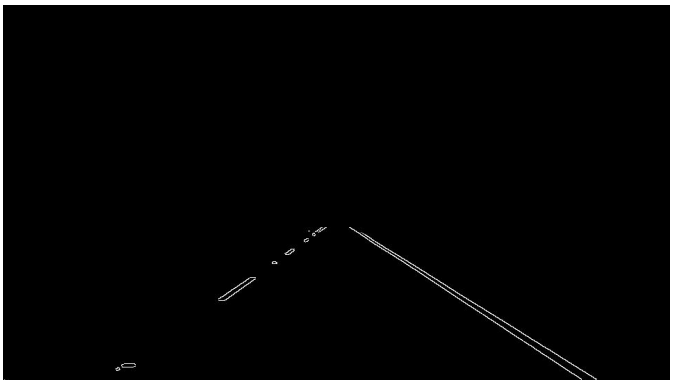




Step 4:

As Canny edge detection was giving me lots of extra edges which I didn't need so I had to apply image masking in order to select only the region of interest from those edges. Using the given helper function, I applied a four-sided polygon mask using well defined vertices.

Output:

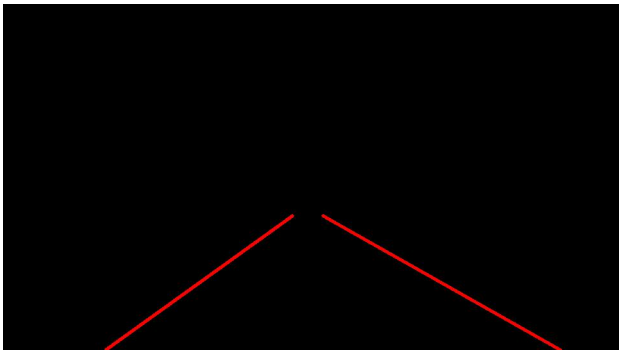
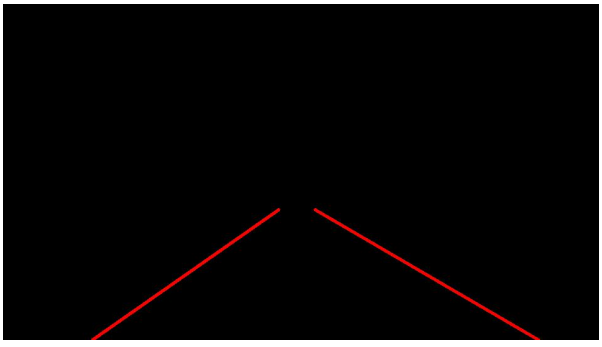



Step 5:

After finding the region of interest, I simply applied Hough transform in order to find all the lines in that given interest region. I used given helper function with below details for each parameter.

```
rho = 1           #distance resolution in pixels of the Hough grid
theta = np.pi/180  #angular resolution in radians of the Hough grid
threshold = 10     #minimum number of votes (intersections in Hough grid cell)
min_line_len = 20   #minimum number of pixels making up a line
max_line_gap = 1    #maximum gap in pixels between connectable line segments
```

Output:

Step 6:

As Hough transform was giving lots of lines so I had to find a way to select the two major lines for either side of lanes and on top of that I had to map out the full extent of the line in order to apply the same model to our mp4 files. As this is the most crucial part of the project so it took me sometimes to find a good implementation of draw_lines() helper function. In order to find the best fitting line for each side of the lanes, my method takes left and right line points based on the value of slope. If slope is + then the points are part of right lane line whereas if slope is – then the points are part of left lane line. And after getting two lists for left and right line points, I am simply using polyfit to find a line that minimize the distances.

Output:


Potential shortcomings with my current pipeline:

The provided challenge video(challenge.mp4) is a challenge. The existing code is not working very well for this video yet as because my model is not being able to fit the road with lots of curves.

Possible improvements to my pipeline:

Dynamic masking and multi section masking in order to detect multi-zones such as

- 10 second ahead,
- 3 second ahead,
- 2 second ahead and
- 1 second ahead.

Add CNN detection for dynamic masking

- Neural Network like CNN has the potential for one program suit all situation as long as we have training set to cover it.