#**Finding Lane Lines on the Road**

Introduction
Appreciate Udacity in providing me an opportunity in learning new technology, deep learning and computer vision for the first time.

The goals / steps of this project are the following:
* Make a pipeline that finds lane lines on the road
* Reflect on your work in a written report

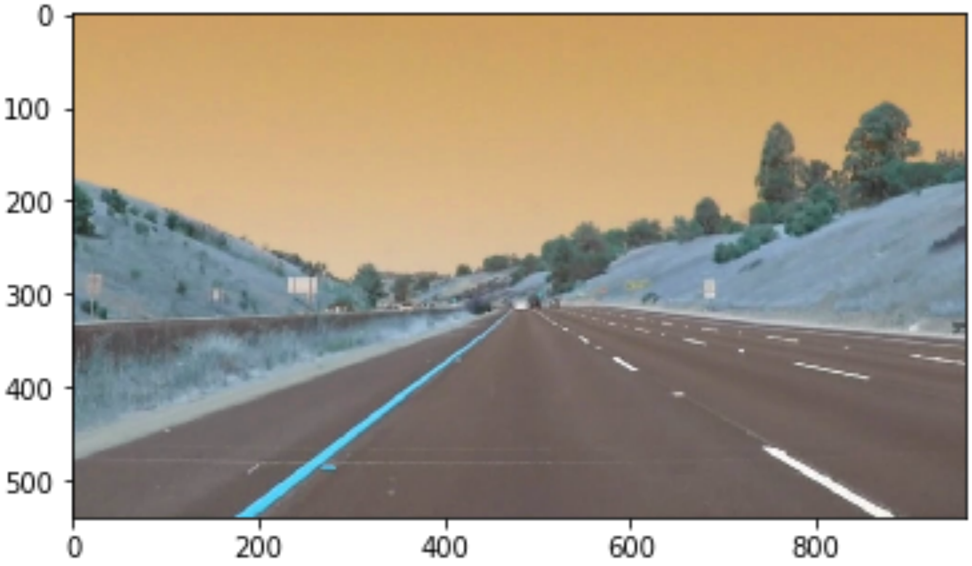
###0. Understanding the Project Pipeline:

1. The first step is to read the test image using cv2.imread API
2. Covert the read image in to a Grey scale image
3. Apply the canny edge detector function to convert the grey scaled image to image to detect the edges
4. Use the Gaussian blur function to reduce the image noise and reduce detail
5. Select the interested region, with the Gaussian image
6. Extrapolated the lines by determining slope, for the left and right side lanes
7. Apply the above logic for mp4 files, one frame at a time

###1. Reflection

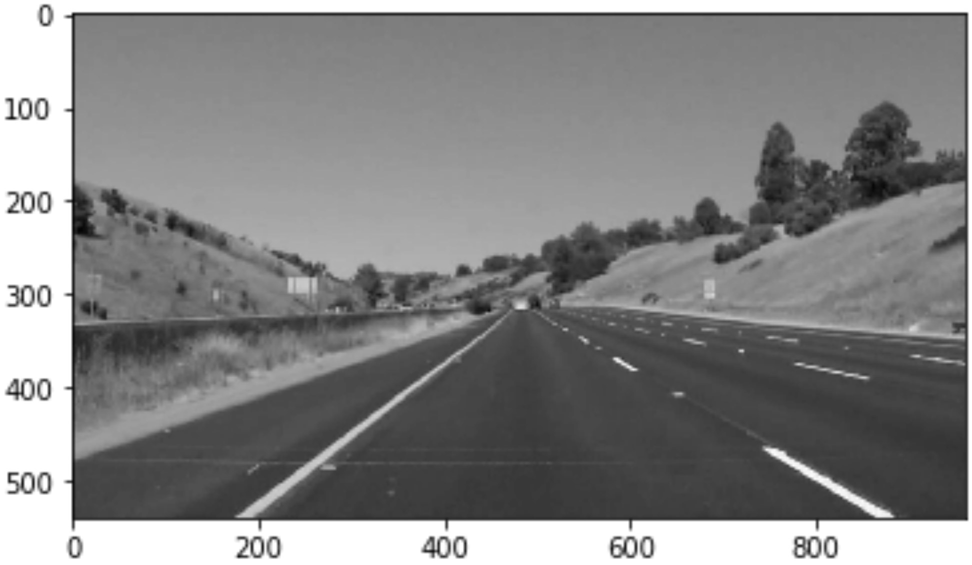
1. The first step is to read the test image using cv2.imread API

```
import os
os.listdir("test_images/")
image = cv2.imread('test_images/whiteCarLaneSwitch.jpg')
plt.imshow(image, cmap='gray')
plt.show()
```



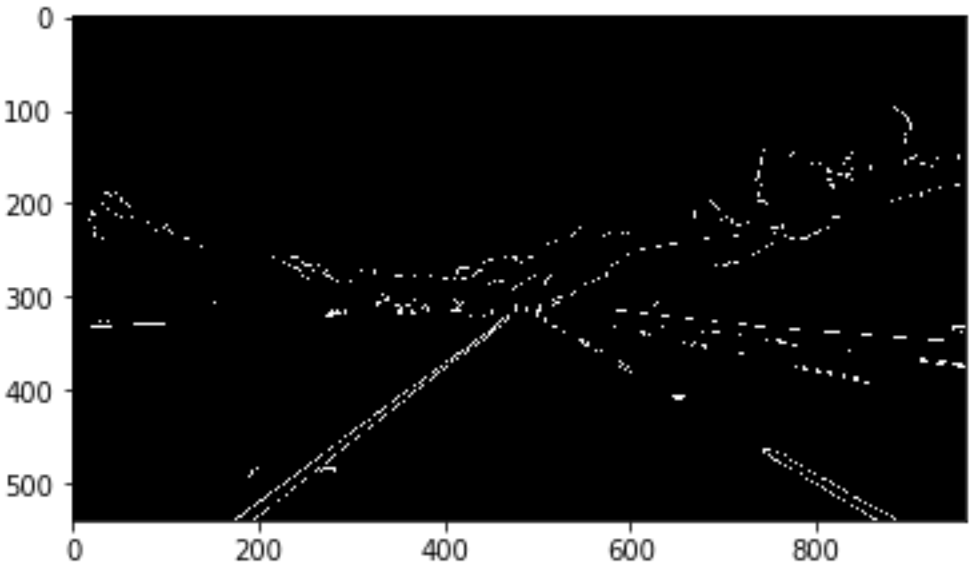
2. Covert the read image in to a Grey scale image

```
gray = grayscale(image)
plt.imshow(gray, cmap='gray')
plt.show()
```



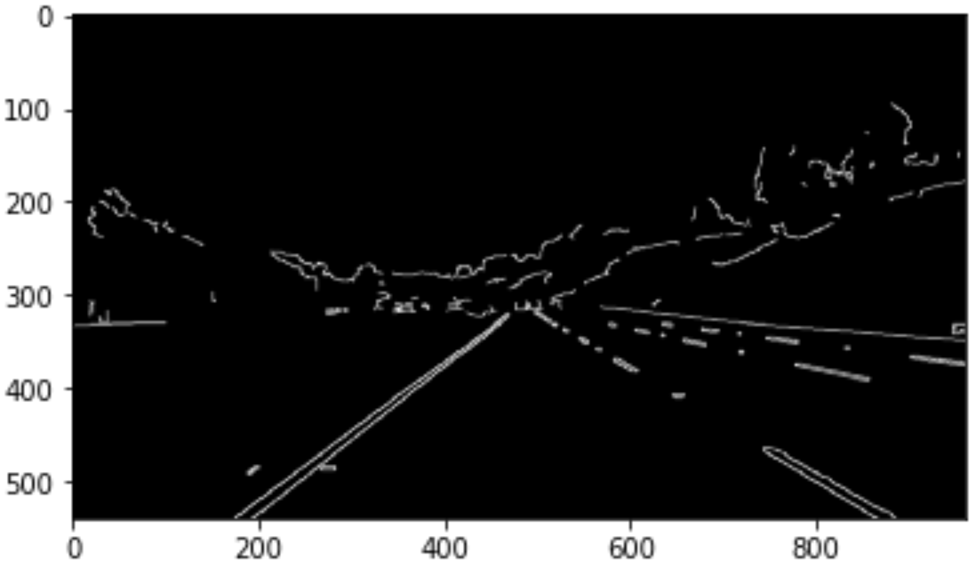
3. Apply the canny edge detector function to convert the grey scaled image to image to detect the edges

```
low_threshold = 150
high_threshold = 255
edges = canny(gray, low_threshold, high_threshold)
plt.imshow(edges, cmap='gray')
plt.show()
```



4. Use the Gaussian blur function to reduce the image noise and reduce detail

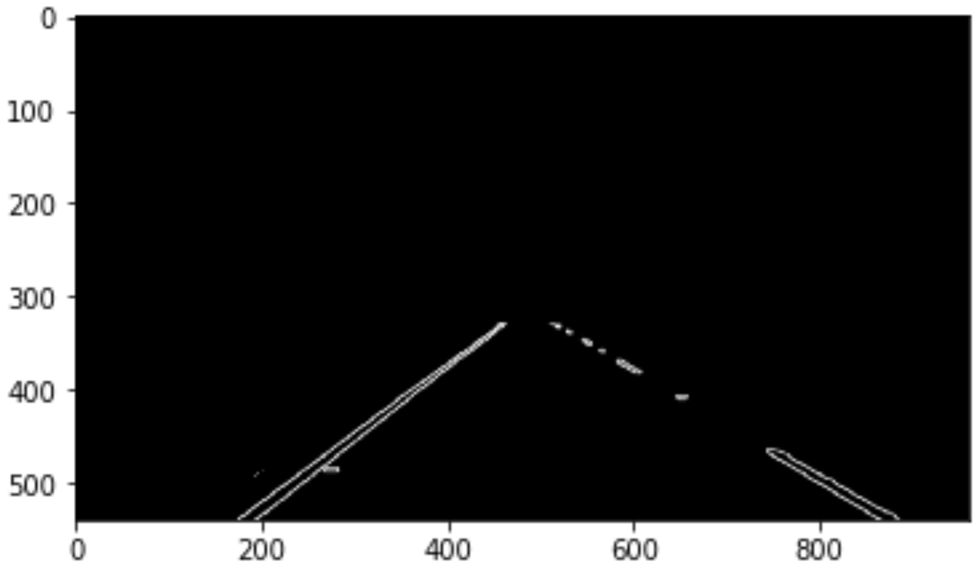
```
kernel_size = 5
blur_gray = gaussian_blur(edges, kernel_size)
plt.imshow(blur_gray, cmap='gray')
plt.show()
```



5. Select the interested region, with the Gaussian image

Region is selected with a Polygon 4 coordinates. This gets the interested region that needs to processed further

```
left_bottom = [120, 540]
right_bottom = [900, 540]
apex = [485, 310]
#apex = [480, 290]
left_top = [440, 330]
right_top = [540, 330]
#vertices = np.array( [[left_bottom, right_bottom, apex]], dtype=np.int32 )
vertices = np.array( [[left_top, right_top, right_bottom, left_bottom]],
dtype=np.int32 )
masked_image = region_of_interest(blur_gray, vertices)
plt.imshow(masked_image, cmap='gray')
plt.show()
```



6. Extrapolated the lines by determining slope, for the left and right side lanes

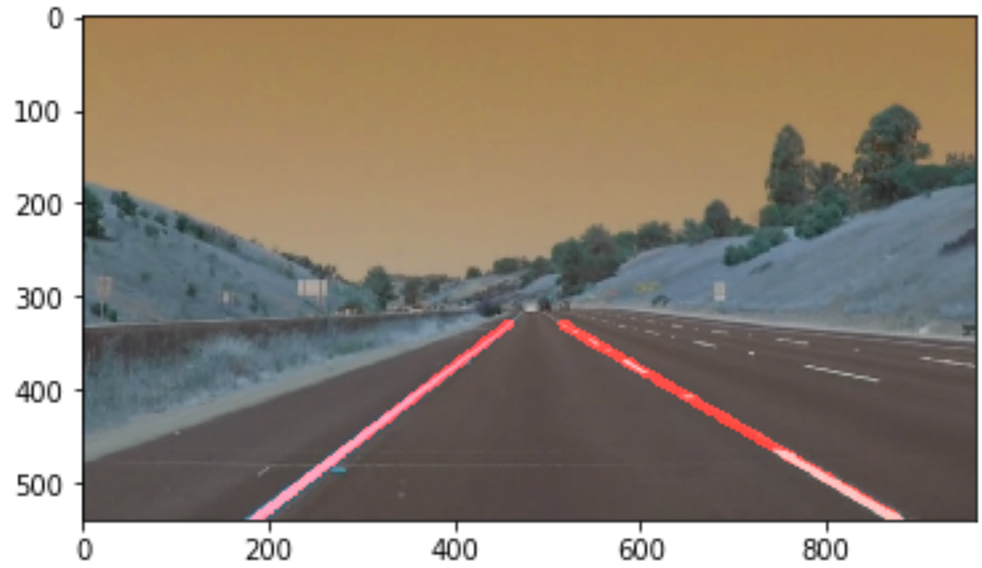
The extrapolation involves multiple steps:

- Draw lines function does the extrapolation, its input parameter is the set of lines and with its (x1, y1) and (x2, y2) coordinates in the image
- The first step is to determine the slope, below document provided better explanation
  - https://peteris.rocks/blog/extrapolate-lines-with-numpy-polyfit/
- Using the above concept, determined the slope and created a left side and right side list
- The goal in this is to determine the 2 coordinates to draw the complete single line
- Go thru the all the left line coordinates and right line coordinates and find the average of the all x1, y1, x2, y2 for each left and right lanes
- Finally determine the Slope, top_x and bottom_x. For the both the right and left side lines and draw the cv2.lines

  ```
  slope = (left_line[3] - left_line[1]) / (left_line[2] - left_line[0])
  ```

```
top_x = int (left_line[0] + (top_y - left_line[1]) / slope)
bottom_x = int (left_line[0] + (img.shape[0] - left_line[1]) / slope)


#Draw left side line
        cv2.line(img, (bottom_x, img.shape[0]), (top_x, top_y), color,
thickness)
```



###2. Identify potential shortcomings with your current pipeline

- In curves there is some variance
- Need fix it for the challenge.mp4, in selecting the precise region
- In testing it showed it to my Son, he was my real qualifier ☺

###3. Suggest possible improvements to your pipeline

- Draw lines can be improved further to have precise match for the lanes
- Should try taking my own video, and try to improve the drawlines and region selection