Finding Lane Lines on the Road Write up

P1: Finding Lane Lines on the Road

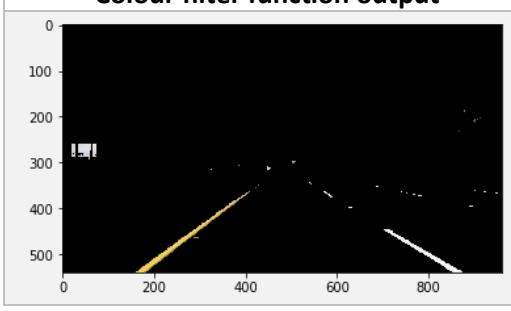
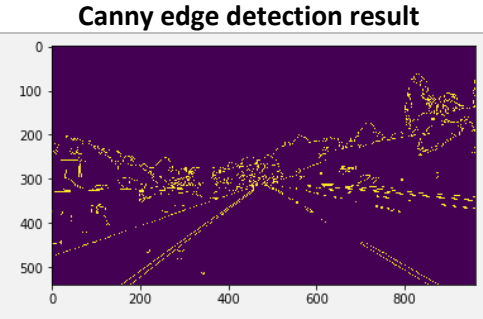
Project goal

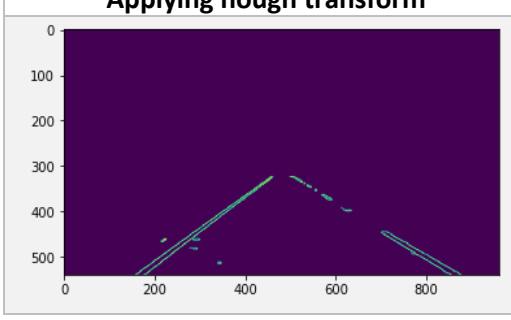
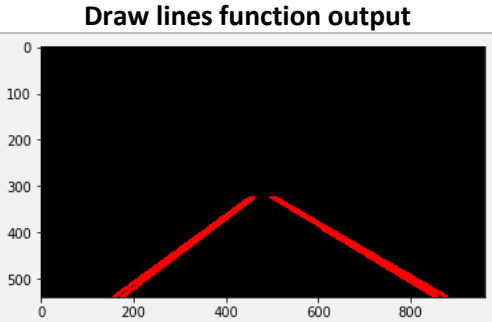
Develop a pipeline which detects lanes on the road on both images and videos

My pipeline Steps

1. Read image or video
2. Apply colour filter colours by thresholding where only white and yellow colours passes the filter to mainly decrease error or false detections as usually lane lines colors are either white or yellow
3. Convert frames to greyscale
4. Apply canny to detect edges
5. Apply Gaussian Filter to smoothen frames and remove noise
6. Apply ROI function to trim part in the frame to be processing or in other words
7. Apply hough transform function to frames resulted from canny
8. Draw lines on detected line using draw_lines() function
9. Combine image with original image to show detected lines
10. Display and save result

Image Test Results

Colour filter function output	Canny edge detection result
	

Applying hough transform	Draw lines function output
	

From my code on Jupyter notebook	From my code using idle on my pc
	

Elaboration on method or approach has been followed in the draw_lines() function.

```
def draw_lines(img, lines, color=[255, 0, 0], thickness=3):
    for line in lines:
        for x1,y1,x2,y2 in line:
            cv2.line(img, (x1, y1), (x2, y2), color, thickness)
```

Hough transform function returns detected lines in **lines** variable

Then by using cv2.line () function draw lines from start and end points of the detected lines using for loop as shown in the code above

Instructions to run files

There are two main files at my workspace to run the program

1. P1.ipynb jupyter notebook where we are assigned to fill our pipeline in
2. P1_Videos.py

P1.ipynb → contains a pipline which process it on images only

You can select image to be processed and saved it at the output folder by hashing and unhashing as shown in the picture below

```
#######################################################################


## Reading Images & Passing them to functions to start processing
#reading in an image
#ximg = mpimg.imread('test_images/whiteCarLaneSwitch.jpg')
#ximg = mpimg.imread('test_images/solidYellowLeft.jpg')
ximg = mpimg.imread('test_images/solidYellowCurve2.jpg')
#ximg = mpimg.imread('test_images/solidYellowCurve.jpg')
#ximg = mpimg.imread('test_images/solidWhiteRight.jpg')
#plt.imshow(ximg)

img = filter_colors(ximg)
img = grayscale(ximg)
img = canny(img,50,150)
```

Note: to obtain a good result on all images, tuning to parameters must take place

Shortcomings in my current pipeline

- Pipe line cannot detect curves
- When changing pictures need to do tuning
- If a white car close to the lane or crossing lane it will cause distortion
- My pipeline cannot play video or save on jupyter notebook

- My pipeline cannot also save video using opencv though I followed open cv documentation for saving videos
- My piple line cannot run all images oneshot in the directory

Possible improvements to pipeline

- Apply bird view to make a the view being processed via pipe line focus on street and avoid vanishing point and road side or sky view
- Draw straight line on the detected lines while don’t show lines that we have detected after applying both draw line functions and Hough transform functions
- Pipeline be able to play video and save in jupyter notebook
- Pipeline be able to also save/write video using opencv through IDLE
- Pipeline be able to run all images one shot from the directory and save them in target directory