Project 1: Finding Lane Lines on the Road

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

Input images:

Are in the folder \test_images

solidWhiteCurve.jpg,
solidWhiteRight.jpg,
solidYellowCurve.jpg,
solidYellowCurve2.jpg,
solidYellowLeft.jpg,
whiteCarLaneSwitch.jpg

Output images:

Are in the folder \test_images

solidWhiteCurvelane.jpg,
solidWhiteRightlane.jpg,
solidYellowCurvelane.jpg,
solidYellowCurve2lane.jpg,
solidYellowLeftlane.jpg,
whiteCarLaneSwitchlane.jpg

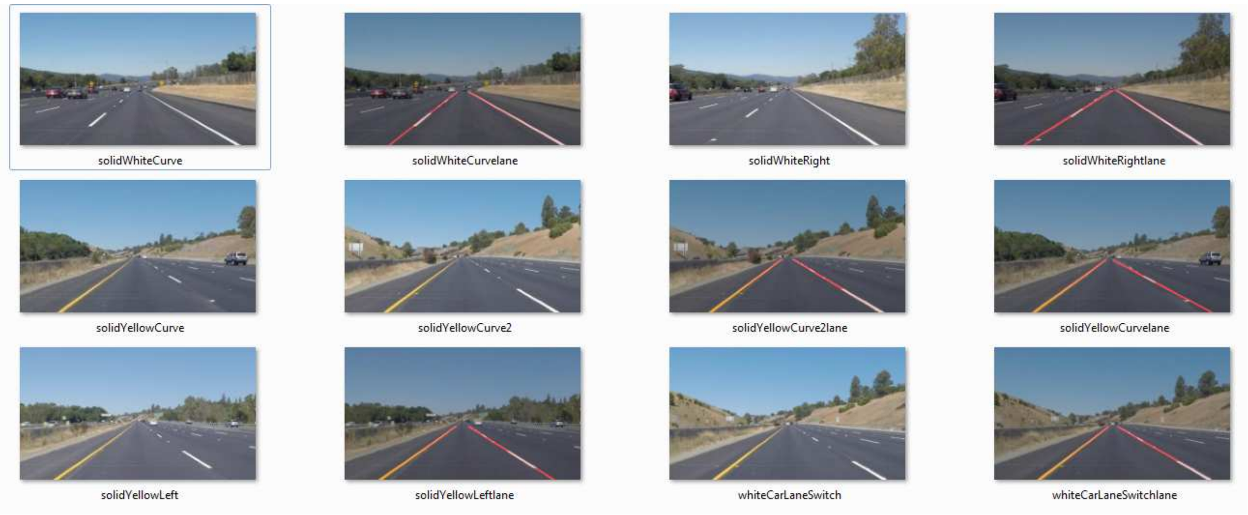


Image pipeline: imagepipeline(image)

Purpose of this pipeline Is to detects lane in an image and draw lines to connect the complete lane

Steps involved achieving the above-mentioned goals:

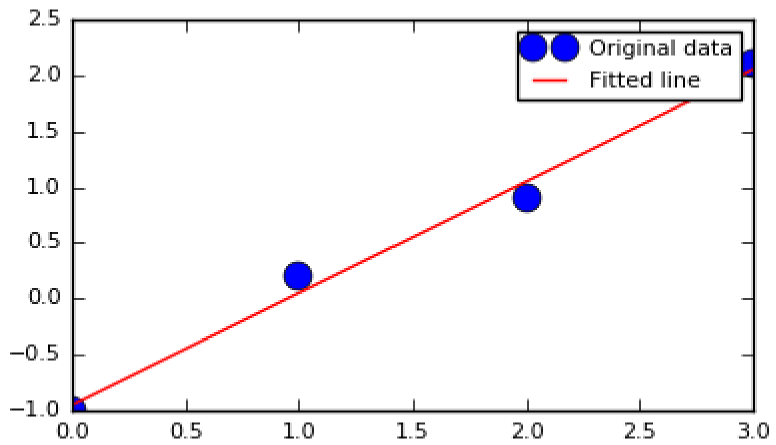
1. Converted the images to grayscale
   cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
2. Reduce image noise by applying Guassian smoothing to image
   cv2.GaussianBlur(gray,(kernel_size, kernel_size),0)
3. Detect edges in the image by running Canny edge detector
   cv2.Canny(blur_gray, low_threshold, high_threshold)
4. Extract the region of the interested in the image, which is projection in front of the vehicle lane it is defined by the quadruple polygon points
   cv2.fillPoly(mask, vertices, ignore_mask_color)
5. Detect line segment which intersects at a common point and draw the line with red color and thickness 2

In order to draw a single line on the left and right lanes modified the draw_lines() function as explained below.

1. First separate the line segments by their slope ((y2-y1)/(x2-x1))
2. Separate the array of (x,y) point into x and y points array
3. Apply liner regression on the arrangement

$$\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ . & 1 \\ . & 1 \\ . & 1 \\ x_n & 1 \end{pmatrix} \qquad \begin{pmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{pmatrix}$$

4. Fit a line through data-points for both left and right lines using linear regression , which uses numpy API : numpy.linalg.lstsq (it is the line that minimizes the sum of the squares of the distance from each data point to the line)



From this technic will find the, line y = mx + b, by calculating values of m and c returned by numpy.linalg.lstsq

5. Since we have the line from previous step, we can find the beginning and ending point of the line
   - Top left point
   - Top right point
   - Bottom left point
   - Bottom right point
6. Draw the line for left and right lane with color red connecting top (x,y) and bottom (x,y), Using open cv line function

```
# Draw the left line
 cv2.line(img, (bottom_left_point[0], bottom_left_point[1]), (top_left_point[0], top_left_point[1]),
              [255, 0, 0], thickness)
 # Draw the right line
             cv2.line(img, (bottom_right_point[0], bottom_right_point[1]), (top_right_point[0],
             top_right_point[1]), [255, 0, 0], thickness)
```

Formulas used for linear regression:

Line = y = mx + b

M = slope and b = y intersect

$$m = \frac{\bar{x}\bar{y} - \overline{xy}}{(\bar{x})^2 - \overline{x^2}} = \frac{\overline{xy} - \bar{x}\bar{y}}{\overline{x^2} - (\bar{x})^2} \qquad b = \bar{y} - m\bar{x}$$

Potential shortcomings with current pipeline

1. **Region Extraction problem:**
   Triangle points are fine-tuned for all the example images and video, but limitations are
   - Detects the White or yellow car at the end of the line driving in middle of the line
   - Line at the end will mix few y axis pixels because it Is not quadruple polygon
   - Line drawing towards farthest y the x is bit inside from the actual Lane because of triangle apex
   - It will not filter out If the image or video with white or yellow objects other than Lane are closer to the current lane at beginning point(y)

2. **External Conditions problem :**
   - Some shadow in the lanes
   - Lane marking is missing for distance less or equal to y axis Region of interest
   - Traffic sign boards on the road edge will be detected also if they are close to end lane

Improvements to your pipeline

1. Draw dynamic quadruple polygon to extract region of interests, which solved problem described in above problem 1.
2. Discard other objects which is yellow and white in color
3. Filter the image to only include yellow and white pixels at first step
4. Use data from Navigation, curvature clothoids information to adjust the line drawing accuracy

Updates after view

1. **Solved Region Extraction problem:**
   Changed from 3 points to quadruple polygon to extract region of interest
2. Fine-tuned threshold, min_line_length, max_line_gap values of HoughLinesP to further extend the line drawing and it also improved the drawing at the center