Project 1 – Finding Lane Lines on the Road

Vitali Mueller

www.vitalimueller.com

Abstract

In this report we will describe the process of finding lane lines on the road. This is the first project of the Self-Driving Car Nanodegree Program in Term 1. In the first week of the course we have learned Image preprocessing techniques to manipulate images, such as select certain regions of the image to preprocess further information and to get rid of unnecessary image content. All practical work is done with Python3 programming language using scientific library such as OpenCV and Numpy.

Keywords: Autonomous Driving, Udacity, Nanodegree, Lane Detection

Project 1 – Finding Lane Lines on the Road

The goal of this project is to find lane lines on the road and draw a color line.

Preprocessing Images / (Videos)

Problem Setup[1]

Following features could be useful in the identification of lane lines on the road:

- Color
- Shape
- Orientation
- Position in the Image

Identifying Lane Lines by Color

To identify Lane Line by Color, we need first to know what color means in case of digital images.

Black and White / Gray image

2D Array, with uint8 it contains values from 0 – 255. Where 0 means black color and 255 white.

Color Image

3D Array, containing 3 channels. Also known as RGB (Red, Green, Blue)

To get white color from an RGB image the value of an particular pixel in 2D space needs to be = [255,255,255] for uint8. For example **image[15, 73] = [255,255,255].**

Pipeline:

1. Step 1: Grayscale
2. Step 2: Gaussian Blur Filter
3. Step 3: Canny Edge Detection
4. Step 4: Set Region of Interest
5. Step 5: Hough Transformation
    - Draw lines

- Use x1,y1,x2,y2 Data points
- To filter out lines with low slope classify lines or data points into:
- Left line, Right line
- Apply extrapolation to find new x,y values to draw lines on the canvas

6. Step 6: Apply “weighted_img” function to draw lines on the image.

Canny Edge Detection

It is one method to use to detect lanes on the road. With edge detection, the goal is to identify the boundaries of an object in the image. Doing this you need:

1. Convert your image into Gray Scale
2. Compute the Gradient

edges = cv2.Canny(gray_color_image, low_threshold, high_threshold)

Rapid change in brightness is where we find the edges

An image is nothing else as an mathematical function f(x,y) = pixel_value. To find out the rapid change and detect edges. We would need to take the derivative of the function:

How to improve the lane lines detection Software?

Knowing exactly where the camera is positioned. One option could be to draw helper lines. For example, one middle line from the driver perspective and other two border lines (left and right from the car). When detecting lane lines, you would need to perform:

1. Distance measurement to compute the distance of the detected lane lines to the border line.

2. Have a threshold value for the distance to filter out lines which are too far away.

Conclusion:

First, it was my first project of this kind. It was very interesting to perform different image processing techniques to find the edges and hence detect lane lines on the road.

I think using region of interest to detect lane lines, is surely one possible way to do it. But is has some disadvantages. It needs to be specifically defined to the camera position. Settings need always to be performed manually to improve correctness of the lane detection system.

In summary, I got myself familiar with OpenCV library. Understanding how to convert and RGB image to grayscale. Applying Gaussian blur filter to remove noise from the image. Applying Canny edge algorithm to detect high gradient changes in the image and therefore the edges. Also doing some linear algebra, applying extrapolation to find new set of points based on the lines which were extracted through Hough transformation.