How can we use Hough transform to detect road?

Author: Mr. Hemant Ramphul

Introduction

The Hough transform is the most efficient and intuitive way to detect straight road lines. If the range of the vehicle's motion and the camera's position are restricted, the line candidate region in the road image can be limited to a narrow range. I propose a novel Hough transform method considering these characteristics. The proposed method efficiently generates the Hough space, reducing the memory usage and generation time. The navigation system of the robot depended entirely on the infrared sensors guiding its path forward, which does the job, but not very good at adjusting to dynamic conditions. Hough transform involves identifying the edges first and then finding identifying the targeted shape.

The Hough Transform

The Hough transform is a features extraction technique used in image analysis, computer vision and digital image processing. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure.

More accurately, the "Hough Transform" is a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form. The classical Hough Transform is most used for the detection of regular curves such as lines, circles, ellipses, etc.

Goal of Hough Transform

The goal is to find the location of lines in images. And this problem could be solved by e.g., Morphology and a linear structuring element, or by correlation. We would need to handle rotation, zoom, distortions etc. Hough transform can detect lines, circles and other structures if their parametric equation is known. It can give robust detection under noise and partial occlusion.

How it works?

In the cartesian plane ***(x and y-axis)***, lines are defined by the formula **y = mx + b**, where **x** and **y** correspond to a specific point on that line and **m** and **b** correspond to the slope and y-intercept.

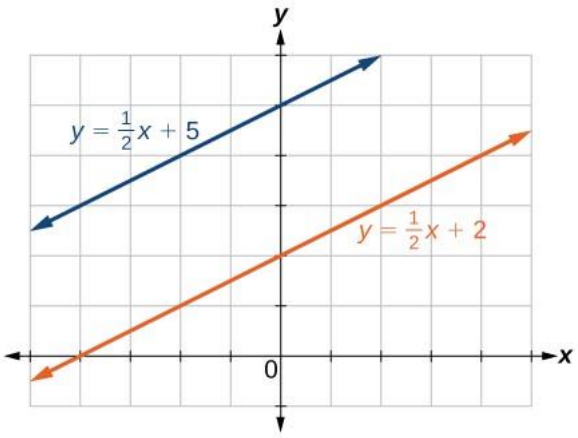


Fig. 1. Lines in Cartesian Coordinate Space

A regular line plotted in the cartesian plane has 2 parameters (***m*** and ***b***), meaning a line is **defined by these values.** Also, it is important to note that lines in the cartesian plane are plotted as a function of their *x* and *y* values, meaning that we are displaying the line with respect to how many ***(x, y)*** pairs make up this specific line (there is an infinite amount of ***x, y*** pairs that makeup any line, hence the reason why lines stretch to infinity).

However, it is possible to plot lines as a function of its m and b values. This is done in a plane called **Hough Space**. To understand the Hough Transform algorithm, we need to understand how Hough Space works.

Explanation of Hough Space

Before we start applying Hough Transform to images, we need to understand what a Hough space is. We can sum up Hough Space into two lines: -

i. **Points** on the Cartesian plane turn into lines in Hough Space
ii. **Lines** in the Cartesian plane turn into points in Hough Space

When we work with images, we can imagine the image being a 2d matrix over some **x** and **y** coordinates, under which a line could be described as **y = mx + b**

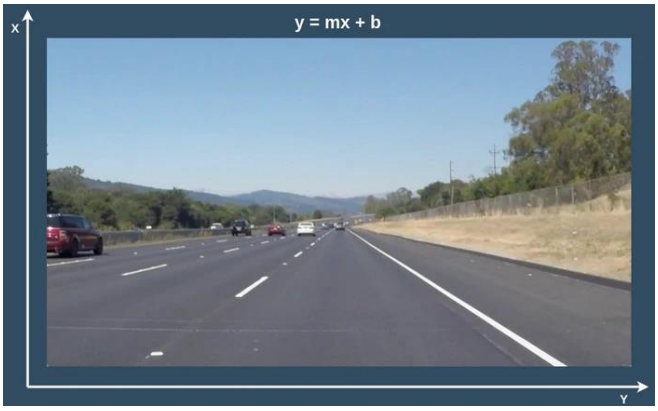


Fig. 2. Parameter space

But in parameter space, which we will call Hough space, I can represent that same line as **m** vs **b** instead, so the characterization of a line on image space, will be a single point at the position **m-b** in Hough space.

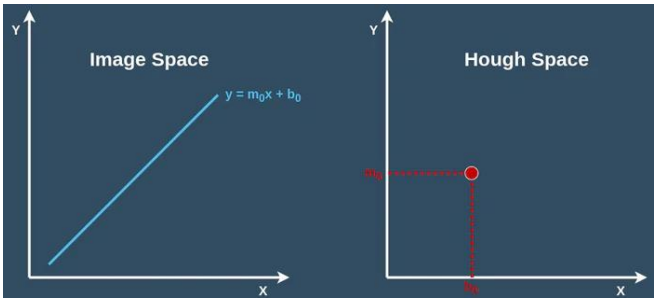


Fig.3. Hough Space

But we have a problem though, with **y = mx + b**, we cannot represent a vertical line, as the slope is infinite. So, we need a better way parametrization, polar coordinates (rho and theta).

Hough space

- **rho**: describes the distance of the line from the origin
- **theta**: describes the angle away from horizontal

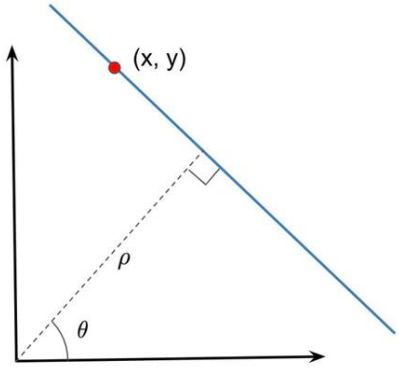


Fig. 4. Line Polar Coordinates

Equation of a line in polar coordinates

The polar form of a line is represented as:

$$\rho = x\cos(\theta) + y\sin(\theta)$$

Here **ρ** represents the perpendicular distance of the line from the origin in pixels, and **θ** is the angle measured in radians

One very important observation though, is what happens when we take multiple points around a line, and we transform into our Hough space.

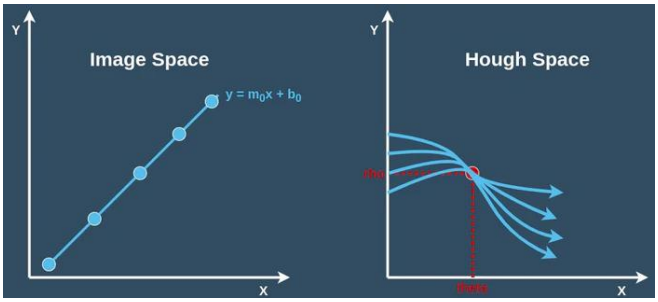


Fig. Dots and Line relation in Hough Space

A single dot on image space translates to a curve on Hough space, with the particularity that points among a line on image space will be represented by multiple curves with a single touchpoint. (1)

And this will be our target, finding the points where a group of curves intersects.

Methodology

Using Python and OpenCV, involves detecting lane lines in an image. OpenCV stands for "Open-Source Computer Vision," and it is a software package that includes several important tools for image analysis.

- The Canny Edge Detection Technique
- Gaussian Blur
- Edge Detection
- Region of Interest
- Hough Transform

Algorithm

The algorithm for detecting straight lines can be divided into the following steps:

1. Determine the range of **ρ** and **θ**. Typically, the range of **θ** is **[0, 180]** degrees and **ρ** is **[-d, d]**, where d is the diagonal length of the edge. Therefore, it's crucial

to quantify the range of **ρ** and **θ**, which means there should only be a finite number of potential values.

2. Create a 2D array called the accumulator with the dimensions **`(num rhos, num thetas)`** to represent the Hough Space and set all its values to zero.

3. Use the original Image to do edge detection.

4. Check each pixel on the edge picture to see if it is an edge pixel. If the pixel is on edge, loop over all possible values of **θ**, compute the corresponding **ρ**, locate the **θ** and **ρ** index in the accumulator, then increase the accumulator base on those index pairs.

5. Iterate over the accumulator's values. Retrieve the **ρ** and **θ** index and get the value of **ρ** and **θ** from the index pair, which may then be transformed back to the form of **`y = mx + b`** if the value is greater than a specified threshold.

Pipeline Steps

1. Conversion of image to grayscale.
2. Applying Gaussian blur on the image.
3. Converting image to an image with only edges using Canny edge detection.
4. Hough transform is used thereafter via Open CV method which helped connecting lines, we are interested in and in eliminating the rest.
5. Last step was to merge both the modified image and the original one to produce the result.

Source Code in Python

Full source code can be download here [H.Ramphul repository]

Result

Original Image

Edges Image

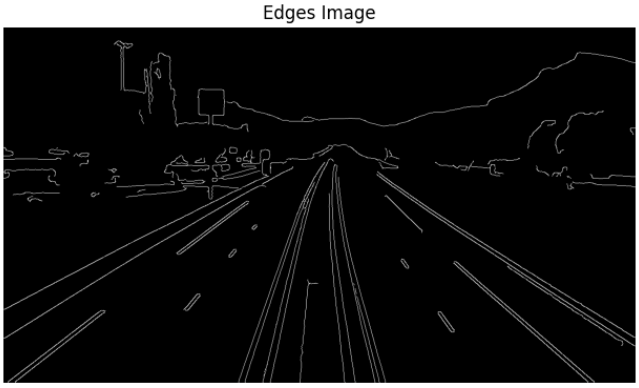


Image with Hough Lines

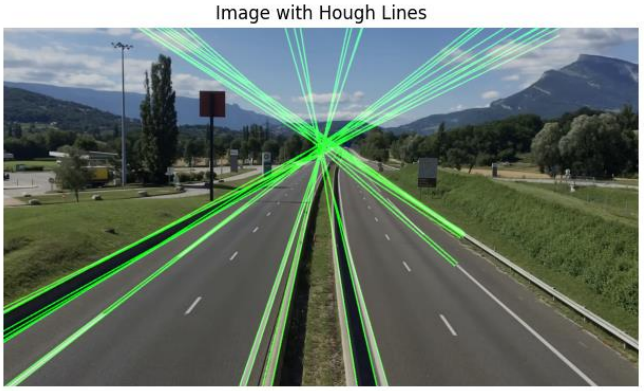
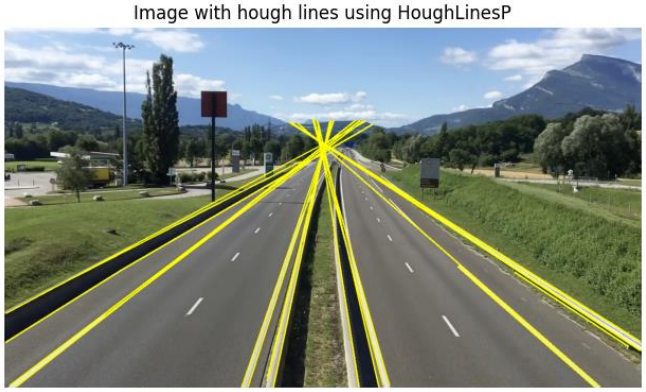


Image with hough lines using HoughLinesP



Drawbacks

i. It can produce deceptive results when objects align by accident.

ii. Rather than finite lines with definite ends, detected lines are infinite lines defined by their ***`(m, c)`*** values.

Conclusion

Hough Transform is an excellent technique for detecting simple shapes in images and has several applications, ranging from medical applications such as x-ray, CT and MRI analysis, to self-driving cars.

To achieve edge detection, we used the OpenCV library and algorithms like the Canny Function. Then we created a zero-intensity mask and mapped our region of interest using the bitwise approach. The Hough

Transform method was then used to detect the straight lines in the image and identify the lane lines. We utilized polar coordinates since Cartesian coordinates could not give an adequate slope for vertical and horizontal lines. Finally, we merged the original image with our zero-intensity image to show lane lines.

References

1. **Martinez, Juan Cruz.** *Understanding & Implementing Shape Detection using Hough Transform with OpenCV & Python.* 2020.

2. **Sinha, Utkarsh.** *The Hough Transform. (n.d.). Retrieved.*

Website

3. A Complete Guide on Hough Transform — **Dulari Bhatt** — Published On June 14, 2022
https://www.analyticsvidhya.com/blog/2022/06/a-complete-guide-on-hough-transform/

4. A Deep Dive into Lane Detection with Hough Transform — **Nushaine Ferdinand** — Published On May 5, 2020
https://towardsdatascience.com/a-deep-dive-into-lane-detection-with-hough-transform-8f90fdd1322f

5. Hough Lines Transform Explained — **Tomasz Kacmajor** — Published On June 5, 2017
https://tomaszkacmajor.pl/index.php/2017/06/05/hough-lines-transform-explained/