ROBUST LANE LINE AND VEHICLE DETECTION

Đào Anh Thành, Lý Trung Kiên, Phạm Tiến Thành, Nguyễn Tuấn Hưng

Instructor: MSc.Bui Quoc Khanh
Faculty of Information Technology, Hanoi University

Abstract:

Automatic Lane Detection is a prerequisite in the development of self-driving cars - a new trend that will surely prevail in the near future. Lane identification has two levels including determining purely by image processing (1) and determining by machine learning (2). Processing with machine learning will produce more accurate results under a variety of environmental conditions than simply processing images. However, that approach requires a huge amount of training model input, along with a high requirement of processing system resources such as CPU, GPU and the other similar types of resource which is not easy to be found by students. Therefore, we choose a simple approach for this problem by districting and processing images which are also the first steps of the level (2) we mentioned above. This comes with both benefits and limitations that we will refer in the paper. Besides, some advanced algorithms are also applied to detect vehicles.

Keywords: Lane line detection, HoughTransform, Edge detection, Retinanet, Traffic sign detection

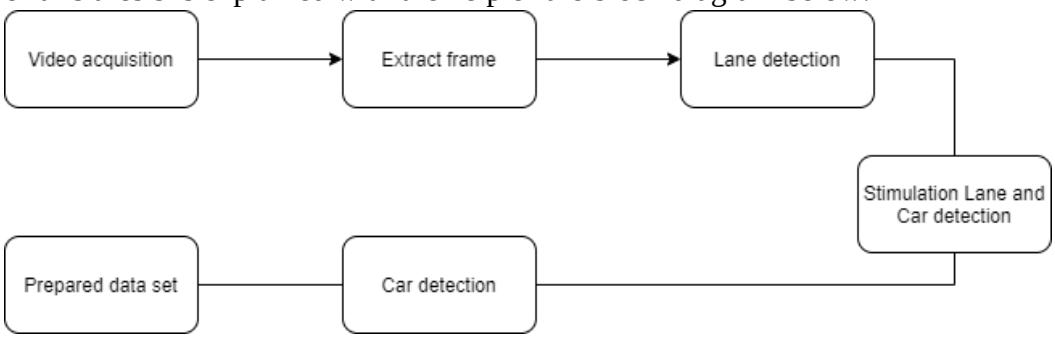
I. INTRODUCTION

1. Motivation

In recent days, vehicles are tending to be more intelligent and convenient. Several smart functions are planned and deployed to improve these machines so that they will be able to assist drivers in both comfort and safety, including night vision assistance, intelligent airbags, crashing avoidance and lane line detection. According to the statistics of the Ministry of Transportation in China, the amount of car accidents caused by deviating from the driving lane is up to about 50%, and in the USA, this number is about 44 % [1]. As a result, lane line detection is a key component for self-driving cars which is used for identifying the path and avoiding collision while driving on the road. Hence, to give an insight on the lane line detection and to improve the efficiency of this system that can help decrease the amount of the accidents caused by lane departure, our group decided to study deeper on this subject, using machine learning base. In this paper, we will analyze the algorithms which was used to build the project including Noise cancelling which was used for eliminating the noise that confounding color moffling that you see, Edge detection that will find the boundaries of objects within images, and Hough line for detecting any shape in images, if you can represent that shape in mathematical form. Moreover, our problems that we have met during deploying the project and the possible solution for them will also be mentioned in this paper. At last, our future plan and possible ways to improve the project will be given to give an overall perspective of this project in the future.

2. Aims and objective

The aim of this project is to provide the fastest way of detecting lane lines and traffic-sign which is based on simple algorithms. Hopefully, it could be a good background for further studies. The clear idea of the important aims and objectives of this thesis is explained with the help of the block diagram below.



3. Relative works

There are many other works and approaches that can achieve the same or even better result. They use many types of neuron network to train machine on how to detect lane line. However, the drawback of it is hardware consuming. Using neuron network such as CNN or retinanet is much effective, but they require a level of hardware that our current machine cannot reach. Therefore, we chosen another methodology, which is quite effective for person who is starter.

II. LANE LINE DETECTION

This is a well-researched area on computer vision. Several algorithms have been developed to distinguish lanes in many environments: shadow, low light and noise input. However, basic steps to do that stay unchanged: extract each frame from input video then process on each image. With method used in this paper, the output reaches some achievements: Adapt specific type of road, detect current road marking in the picture and adapt to different type of light and condition

1. HoughLines Transform

To use HoughLines Transform [2], the processed image should be binary. However, we would like to search for the straight lines on an original, color image since our video is in RGB color space. Therefore, probably the most common solution is to firstly grayscale the image and then to detect edges. Such mask of edges can be then fetched to the Hough Lines method which should output a set of straight lines found on the image.

Figure 1 - RGB-based to grayscale-based

The basis of HoughLines transform bases on the ground which we will discuss deeply on the sections below.

a. Straight line representation

As we all know from early school class, a line can be represented in the Euclidean plane using two parameters: slope and intercept. The line then can be described as: y = ax + b. However, to detect an average line through many scattered points with it is impossible. We therefore unambiguously describe the line using the pair *(ρ, θ)* in the polar system. The first parameter, *ρ,* is the shortest distance from the origin to the line (approaching the line perpendicularly). The second, *θ*, is the angle between x-axis and the distance line. One of the benefits of such representation is that we can describe vertical lines by *ρ* and *θ* which is impossible by using only *(a, b)* parameters in the Cartesian system.

The line now can be represented like that: $\rho = x\cos(\theta) + y\sin(\theta)$

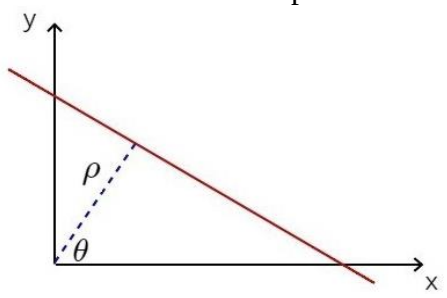


Figure 2 - Line in the Cartesian plane.

b. Mapping image space to Hough space

Hough space also uses 2 parameters to represent a straight line (*ρ, θ*). The figure below will demonstrate how to map a line from Euclidean to Hough space.

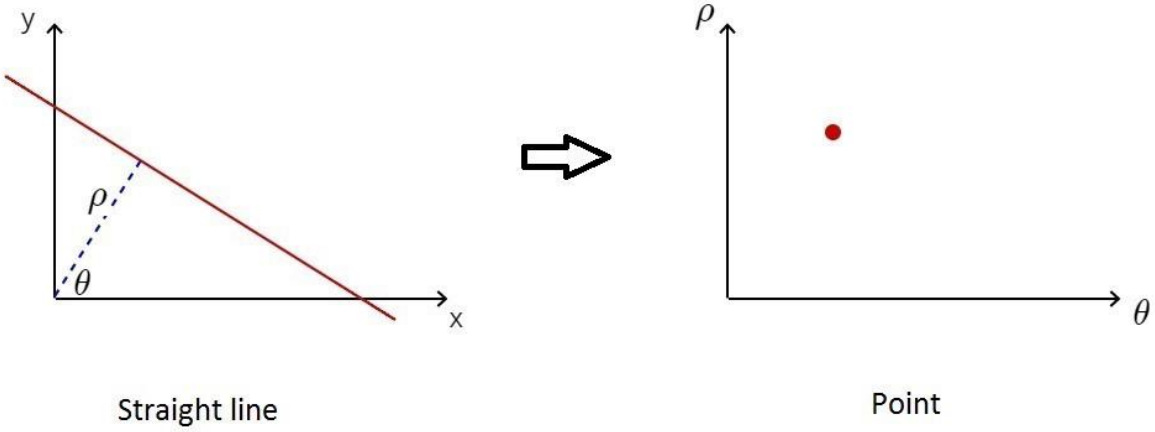


Figure 3 - Mapping Euclidean space to Hough space

Similarly, if there are many lines represented in xOy space, each of them can also represented in Hough space as shown on figure below

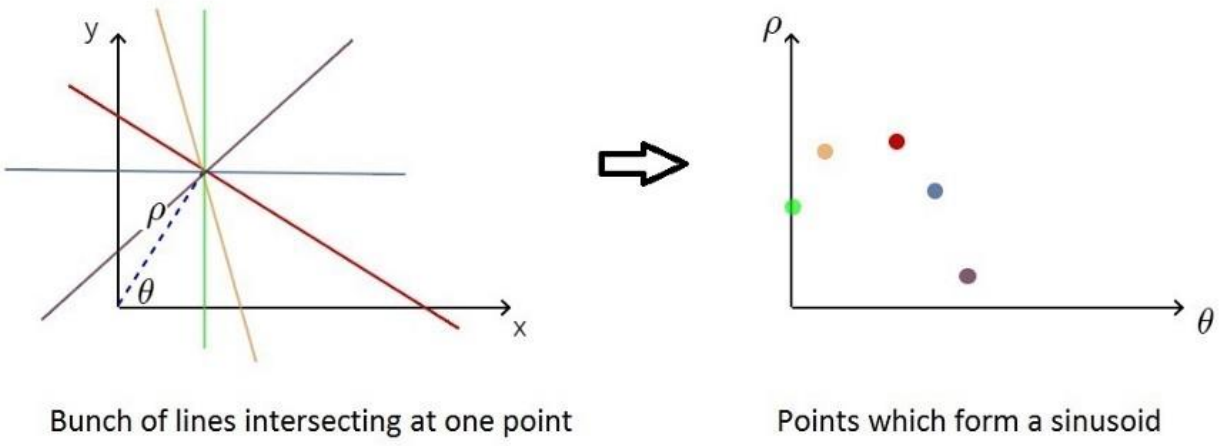


Figure 4 Represent many lines on Hough space

It turns out that these points in *(ρ, θ)* space are forming a sinusoid. Drawing an infinite number of additional lines intersecting at this one point would result in a continuous sinusoid in Hough space. So, maybe, we can say that a **point** in image space results in a **sinusoid** in Hough space? Let us recall the equation $\rho = x \cos(\theta) + y \sin(\theta)$. Indeed, for fixed (x, y) parameters representing point in image space and sliding through all possible values of θ in some range, we obtain ρ values which form a sinusoid.

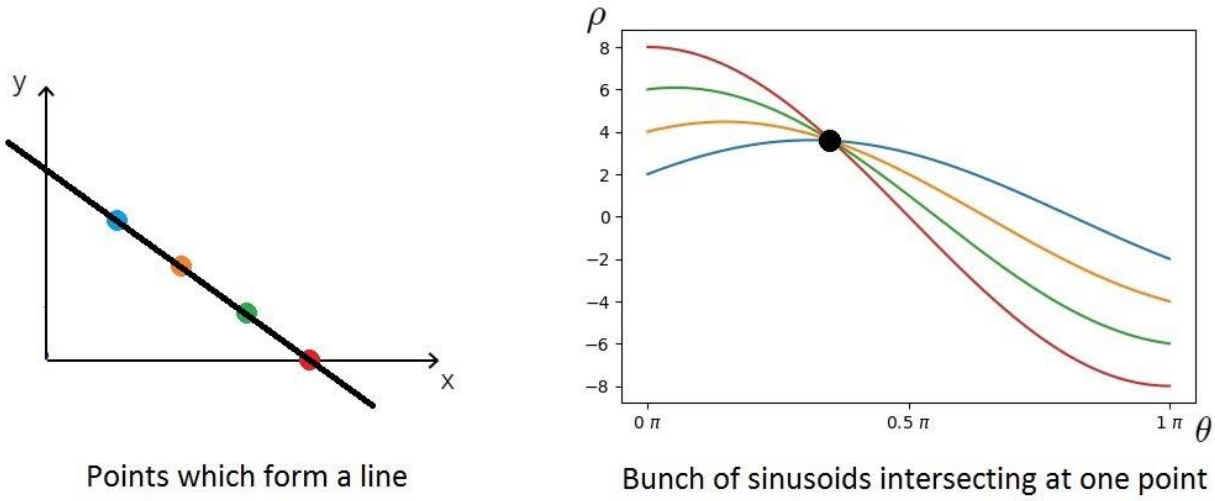


Figure 5 a (left) and b (right) Detect straight line goes through several of dots using Hough space

In fact, in lane lines detection, we need to draw lines based on points which do not lie on a straight line. In other terms, we find the closest axis that goes through all of them. We do it by grouping value from figure 5b: choose a range of value of *(ρ, θ)* that holds a specific number of points to draw a line through, which we also known as thresholds. More explanation presented in *Figure 6*.

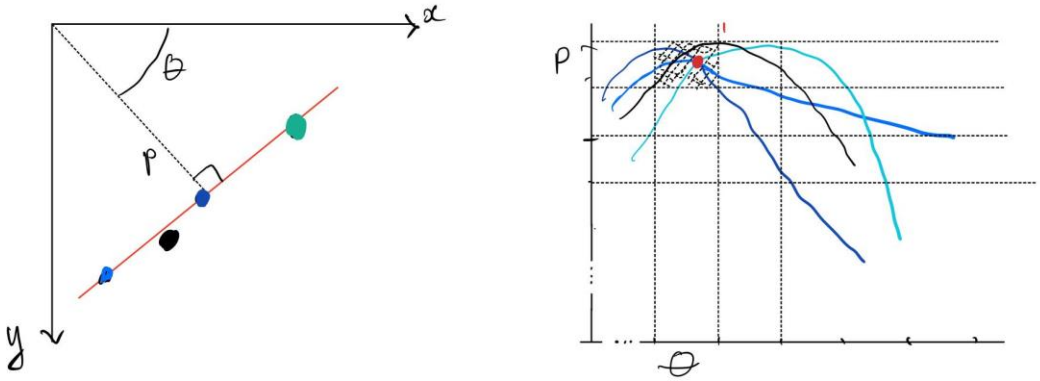


Figure 6 drawing a line through points by choosing a rectangular region in Hough space that holds a specific threshold.

In our work, we choose the minimum threshold to draw line is 100 intersection points in a grid *(ρ, θ)* = (2 pixels, 1 radian), since from our experiment, we believe it is the optimize value for this problem.

2. Pre-process for a single frame

a. Noise reducing

As many other data processing, we do not want random variation of brightness or color information in image affects the accuracy of the output result. Therefore, moving as much as noise from the input is necessary. There are six type of noise [quote] that may appear in image, so applying various methods of noise reduction is recommended. Since our input video is quite clean, we only use *Gaussian blur* to make the difference between lane lines (objects) and road (background) become more significant. Moreover, we are processing on a single dimension image (grayscale) instead of 3d image (RGB), therefore the algorithm above was applied, where x is the distance from the origin in the horizontal axis, and σ is the standard deviation of the Gaussian distribution [3]

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

b. Edge detection

Finding lines using Hough transform requires a collection of point that lie on it. In other word, distinguishing between lanes line (objects) and road (background) is fundamentally. To do that, we use a technique called ‘edge detection’. Especially in this project we are going to apply the Canny edge detection algorithm. “Canny edge detection is a multi-step algorithm that can detect edges with noise suppressed at the same time” [4]. We started applying this theory step by step:

First step is to detect the edge of each lane line in each image (as videos are composed of number of images). Every pixel of the image has the color gradient from 0 - 255, so we need to convert the picture to gradient only.

Figure 7 Image after processed to gray gradient

The next step is to make the unimportant part of the image(noise) become blur (explained in the above section), which aids the lane detection process as the image becomes smoother, the technical phase is “Reducing noise”. The purpose of this is to smooth the image with a Gaussian filter to reduce noise and unwanted details and textures. As all edge detection results are affected by the noise in the image, it is necessary to filter out the noise to avoid false edge detection caused by it, and to smooth the image, a Gaussian filter kernel is convoluted with the image. The equation for a Gaussian filter kernel of size $(2k+1)\times(2k+1)$ is given by

$$H_{ij} = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(i-(k+1))^2+(j-(k+1))^2}{2\sigma^2}\right); 1 \leq i,j \leq (2k+1)$$

After having the processed image, we focus on studying the basic theory of Canny Edge detection algorithm to apply to these images and come up with the conclusion that if exists the portion of the image space that has the gradient descent from maximum to minimum value (255 to 0) or vice versa, that descent represents the transparent from an edge of the lane line to the road. And from that we can also conclude that if portion of the image does not have that descent gradient is also considered not the lane line edge

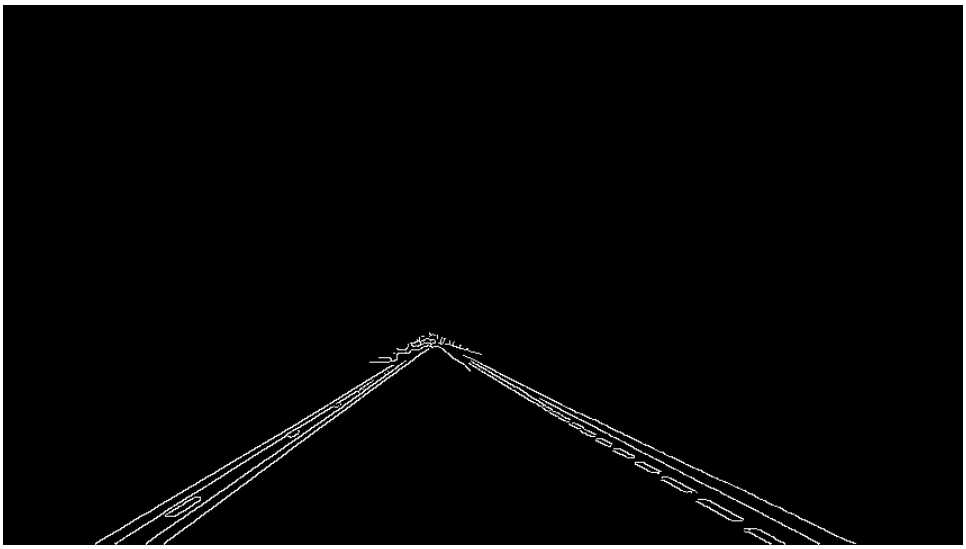


Figure 8 Detected lane-line edge

c. Lane lines detection

After preprocessing, remain steps is just showing result on the screen (we used cv2.line()). However, lines found will not fit the road lane line, as shown in *Figure 9.*

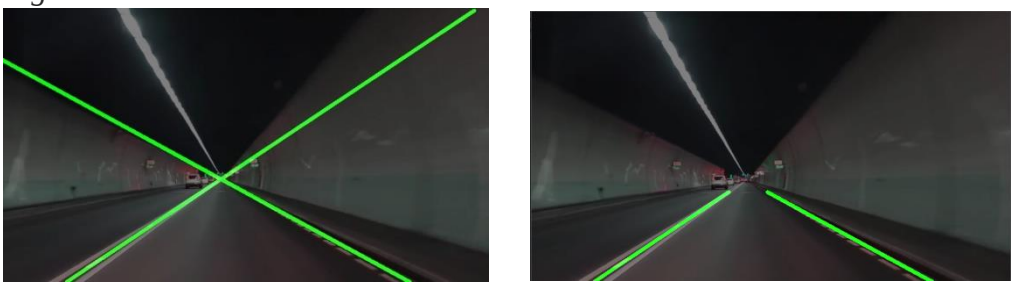


Figure 9 Detected line is overfitting to road lines (a) and after re-calculation (b)

The reason for that is while we only need a segment to fit a specific **segment** of road line, what we detect using HoughLines is a whole straight **line**. To handle that we wrote a function to recalculate displayed segment based on the slope and intercept which have been found using HoughLines.

III. CAR DETECTION

We planned to solve this problem using Convolutional Network method (CNN) since quality trained datasets (*.h5, *.yolo,…) are extremely available on the internet. However, as we mention in section I, our machine does not have enough power to do that. Therefore, we applied another methodology named Haar Cascade. Bared in mind, in this article we will not discuss in how to do that. In-depth analyze and explanation can be found in this article [5].

1. Applying Haar Cascade (Haar-like feature) for detecting car

This technique use “sliding windows” for capturing characteristics of a particular object (in this case is car). It is similar to “convolutional” in CNN, but we have to define characteristics of object manually instead of automatically (CNN). Plus,

since this algorithm only focus on pre-defined features and skips all the other, it fast enough to detect object in real time (processing time on our machine - 25% of i7-8750H CPU and take 600Mb ram in average - is 30ms)

By building our own car-like characteristics, we then obtained the following results, which is quite satisfy our initial goals.

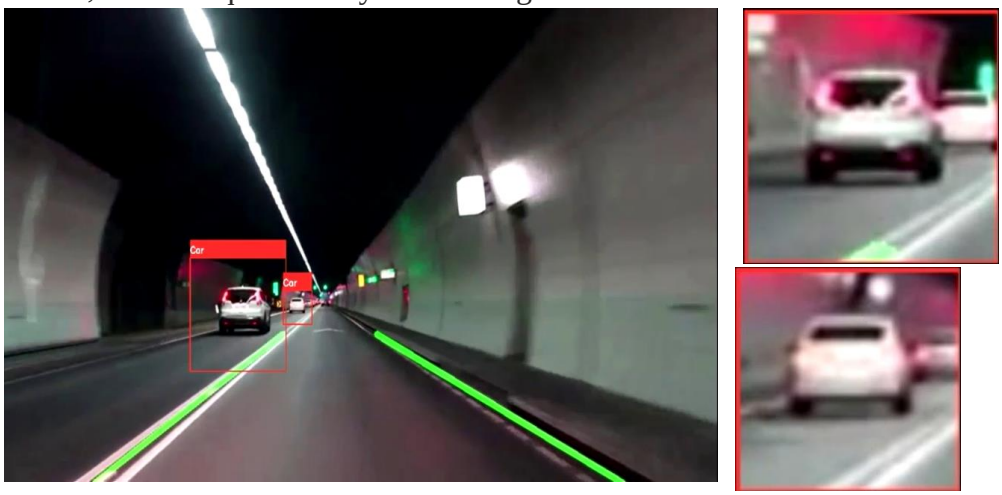


Figure 10 Final output

IV. FUTURE WORK & CONCLUSION

To conclude, Lane Line Detection is one of the key functions that can improve and enhancing the ability of intelligent vehicles in the future. In general, the lane line detection can be used to detect lane line and car through a camera while people are driving. In real world, this algorithm can only be applied exactly under one particular condition. The algorithm is not applicable to other conditions because it is built on the environment of a particular video. Furthermore, car detection accurately detects 100% vehicles on the road, but with noise models having many characteristics which is similar as car such as billboards, traffic sign, etc., it is sometimes mistakenly detected because there is no model for defining and excluding these elements. In future, we are tend to improve those limitations. Lane line detection and car detection will be promoted and optimized so that even devices with low-end hardware can successfully use.

Reference:

[1] **Weiwei Chen, Weixing Wang, Kevin Wang, Zhaoying Li, Huan Li, Sheng Liu** "*Lane departure warning systems and lane line detection methods based on image processing and semantic segmentation: A review*". 2020.

[2] **Thomas Risse** "*Hough transform for line recognition: Complexity of evidence accumulation and cluster detection, Computer Vision, Graphics, and Image Processing"*. Volume 46, Issue 3, 1989. Pages 327-345.

[3] **A. Pathmanabhan and S. Dinesh** *"The Effect of Gaussian Blurring on the Extraction of Peaks and Pits from Digital Elevation Models".* November 28, 2006.

[4] **Ruye Wang** *"Canny Edge Detection".* September 25, 2013.

[5] **Moch Ilham Ramadhani, Agus Eko Minarno , Eko Budi Cahyono** *"Vehicle Classification using Haar Cascade Classifier Method in Traffic Surveillance System".* December 2017