2023 8th International Scientific Conference on Applying New Technology in Green Buildings (ATiGB)

Applications of image processing techniques to determine the lane for autonomous vehicles

*Pham Quoc Thai
Faculty of Transportation Mechanical Engineering, University of Science and Technology – The University of Da Nang, Danang, Vietnam
pqthai@dut.udn.vn

*Duong Van Hoa
Center of Electrical Engineering, Duy Tan University,
Danang, Vietnam
bk17c4a@gmail.com

Nong Trong Tu
Center of Electrical Engineering, Duy Tan University,
Danang, Vietnam
trongtu4@gmail.com

Du Van Ngan
Center of Electrical Engineering, Duy Tan University,
Danang, Vietnam
nganc4bk@gmail.com

*Ngo Tan Thong
Faculty of Mechanical Engineering, University of Technology and Education – The University of Da Nang, Danang, Vietnam
ntthong@ute.udn.vn

Phan Van Binh
Faculty of Mechanical Engineering, Duy Tan University,
Danang, Vietnam
phanvanbinh@dtu.edu.vn

Abstract — Autonomous vehicles are considered a huge human technology foundation for the intelligent vehicle industry. To operate on the road, autonomous vehicles must be equipped with many sensors capable of detecting problems in moving instead of humans. In addition to the control software, many modern technologies and algorithms have been applied today, such as image processing, artificial intelligence, etc. This paper describes a method to build an autonomous vehicle system with real cameras, performing the task of automatic navigation in traffic conditions with lines. Lane detection is essential in determining the center of self-driving vehicles concerning the moving lane.

Keywords—Lane detection algorithm, Autonomous vehicles, Image processing, Artificial intelligence, Sensors.

I. INTRODUCTION

Autonomous vehicles are being widely applied in many different fields, especially playing an essential role in a traffic environment with complex travel conditions. The self-driving system includes image processing and control units [1].

The image processing unit uses sensors, such as a camera, lidars, radars, and a GPS, to acquire information from the environment, including lanes, people, and surrounding objects [2]. Therefore, algorithms are used to process the parameters received from the sensors.

The control unit helps the vehicle move along its limited lane and avoid obstacles on the road. The control part covers electric motors, steering systems, etc. These devices will receive control signals from the central processor through power circuits that control the motors with electricity and perform steering to help the car drive itself in the restricted lane. There have been many different research proposed to solve this problem. Widely used studies such as quadratic curve [3] or Bsnack [4] and some new research [5-6-7] have been applied. However, this problem of identifying self-driving lanes [8] for autonomous vehicles is constantly being researched and developed to come up with the best optimal solutions.

This study describes a practical, simple, and effective method of lane identification using a computer vision and image processing system combined with cameras to obtain information to identify road markings and provide traffic scenes, giving warnings to help the car stay on the right. First, we create a filter for the lane of interest through noise reduction. Noise can produce false edges, so before going any further, it is imperative to perform image smoothing. Gaussian blur is used to complete this process. We will then convert it to an HSV color scale and do color filtering and region of interest. Next, we perform straight-line detection based on canny edges using the Hough Transform algorithm and build bounds for left and right lanes. Finally, based on the left and right lanes, we determine the center lane and calculate the range of steering angles. As a result, we give warning signals and control the car to steer away from the lane and stay in the restricted lane.

II. OVERVIEW OF THE LANERTD ALGORITHM

The most significant advantage of the laneRTD algorithm is taking the signal from the camera to the computer for processing. The laneRTD image processing algorithm enables lane edge detection based on color and the Canny edge detection algorithm.

In this study, we cover the processing of RGB color images obtained from the camera and the process of converting the image to grayscale. For the convenience of later processing, the image is further filtered and smoothed using the Gaussian Blur algorithm [9]. The Canny function was used for further processing for edge detection purposes after image preprocessing [10]. We use the Roi algorithm to achieve high-precision processing while eliminating unnecessary areas. With the application of the Roi algorithm in the processing, limiting a trapezoidal region of interest is possible. The Hough transform is used to construct the line representing the left and right lanes separately. It is possible to calculate and output the midline from these two separate lines. The processing and output of this road have determined the center of the lane, which is the basis for helping the car stay in the right lane.

2023 8th International Scientific Conference on Applying New Technology in Green Buildings (ATiGB) | 979-8-3503-4397-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ATIGB59969.2023.10364523

The advantage of the LaneRTD algorithm is that it streamlines a system of computer vision algorithms, starting with the camera correction algorithm, until the lane highlighting is determined, as well as measuring the curvature of the road. Several edge detection and color recognition techniques use multiple color spaces.

The diagram of the lane detection algorithm is implemented as follows [11]:

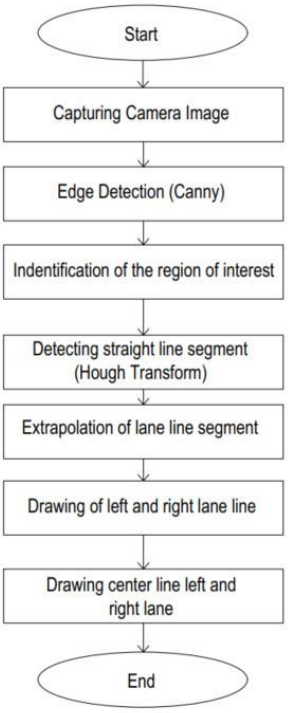


Fig. 1. Flow chart of the lane processing algorithm

1. The Canny edge detection algorithm [10], [12]

The Canny Edge Detector is a multi-stage algorithm for detecting a series of edges in an image, developed by John F. Canny in 1986. Canny put forward a computational theory of edge detection that explains why the technique works.

The Canny Edge Detection algorithm is usually implemented through the following five steps:

Step 1: Noise Reduction

To remove noise factors, the method we give in the study is applying a Gaussian filter to help smooth the image [13]. The image convolution technique is applied with a Gaussian Kernel (3×3, 5×5, 7×7, etc.). The kernel size depends on the desired blurring effect [14].

Equation for a Gaussian filter kernel of size (2k + 1) × (2k + 1):

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

Step 2: Calculate the grayscale gradient of the image.

The grayscale gradient calculation step is usually calculated based on the intensity and direction of the grayscale gradient. The edges correspond to the change in intensity of the pixel's brightness. The easiest way to detect it is to apply filters that highlight this change in intensity in both directions: horizontal (x) and vertical (y).

After the image is smoothed, the derivatives Ix and Iy w.r.t. x and y are calculated. This can be done by convolution I with the Sobel kernels Kx and Ky, respectively:

$$K_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, K_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & 1 \end{pmatrix}$$

Sobel Filter is for both horizontal and vertical. Then, the magnitude G and the angle θ of the gradient are calculated as follows:

$$|G| = \sqrt{I_x^2 + I_y^2},$$

$$\theta(x, y) = arc\tan\left(\frac{I_y}{I_x}\right)$$

G and θ are called the magnitude and direction of the gradient.

Step 3: Apply Non-maximum suppression.

After calculating the grayscale gradient of the image, the gradient intensity level is between 0 and 255, so the edge brightness is not uniform. The thickness of the edge is also uneven and mixed with thick and thin lines often appear. Therefore, it is essential to remove thick edges by applying Non-maximum suppression. The job is to thin the edge by going through the points on the gradient intensity matrix and finding the pixels with the maximum value in the edge directions.

Step 4: Double threshold

This step aims to identify three types of pixels: strong, weak, and outliers:

Strong pixels are intense and directly contribute to edge formation [15].

A weak pixel is a pixel whose intensity value is not enough to be considered strong but is not small enough to be considered an outlier.

Other pixels are considered outliers.

Step 5: Edge Tracking by Hysteresis

Hysteresis tracking converts weak pixels into strong pixels if and only if at least one of the pixels surrounding the pixel in question is strong; the remaining weak pixels are discarded. This kind of removal ensures that the noise is removed from the filter.

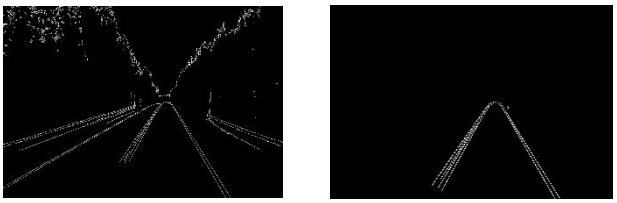
Fig. 2. Image result after processing Canny edges

2. Extract area of interest (ROI)

ROI comes in many shapes, be it square, rectangle, circle, many small areas on the image, or even pixels. ROI is a mask on which the pixel of interest has a value of 1 (white), and the

pixel of no interest has a zero value. The area of interest in the image is usually the background [16].

In this study, we use ROI to remove the upper landscape part of the frame and keep the area of interest as the lower pavement area. First, we create a zero-intensity image with the same size as the frame, then focus only on the bottom half of the screen, which allows the mask to be filled with values of "1" and the other areas filled with "0" values. Finally, do a bitwise operation between the mask and the frame, keeping only the rectangular area of the frame.



a, Image before processing b, Image after processing

Fig. 3. Image result after ROI treatment

3. Hough Transform (for straight lines).

Hough transform is a technique that lies in the shape of an image [17]. In particular, the Hough transform has been used to extract lines, circles, and ellipses (or cut cones). In the case of a straight line, its mathematical determination is equivalent to the Radon transform.

We will first look at finding lines in an image. In a Cartesian parametric representation, the point of collinearity in an image with coordinates (x,y) is relative to the slope m and the intercept c by:

$$y = mx+c$$

Or written as:

$$r = x\cos\theta + y\sin\theta$$

where r is the perpendicular distance from the origin to the line, and θ is the angle between this perpendicular and the horizontal axis measured counterclockwise. Therefore, any line can be represented by these two terms.

There are two ways to determine a straight line, but people choose the second way because only two parameters (r, θ) are needed, which is more convenient for calculation. Since θ is in radians, it has a value in the range [0;π] (or [0;3.14]). When θ equals 0 or 3.14, the line is vertical, and θ equals 1.57 (π/2) is horizontal.

Since each line is defined by two values (ρ, θ), the algorithm creates a 2-dimensional array. The row corresponds to ρ, and the column corresponds to θ. The size of the array depends on your choice, and of course, the larger the array, the more accurate and longer it takes to compute, and the smaller array is faster but not as precise.

When using, the user will pass the values ρ and θ they want. The algorithm will draw ρ x θ lines on the image. For each line, it counts the number of pixels lying on that line, and for each pixel found, it adds a value to the cell corresponding to (ρ, θ) [18].

After processing and converting the image from gray to RGB color scale and displaying the results on the screen, we get the following results:

a, mage before processing b, Image after processing

Fig. 4. The result of image processing and RGB color return

III. IMPLEMENTATION OF THE LANERTD ALGORITHM

Our image processing program is written in Python [19] and uses the OpenCV library for image processing.

1. Image reading test: Acquiring information from a computer (image).

2. Convert the color test image to grayscale: this is done using the OpenCV function "ctvColor".

3. Noise filtering: This is done using the OpenCV function to execute the Gaussian filtering algorithm. Kernel size is 5 has been selected.

4. Edge detection and extraction: This work is done by using the OpenCV function "Canny" to execute the famous Canny algorithm.

5. Area of Interest Recognition: This is done by masking a trapezoidal area in the image with detected edges so that each Hough line (segment) can be identified by its two endpoints. Transform Can be modified to predict representative curves.

6. Develop straight lines: Lines are determined by the Hough algorithm in polar mode using the OpenCV HoughLinesP() function. This is done using the OpenCV function "Canny" for edge detection and extraction.

7. Calculate the straight line between 2 lanes: After processing and drawing two left and right lanes, calculate and draw a straight line between the two lanes. This is the center of the road, where the vehicle must keep a steady state to follow this lane.

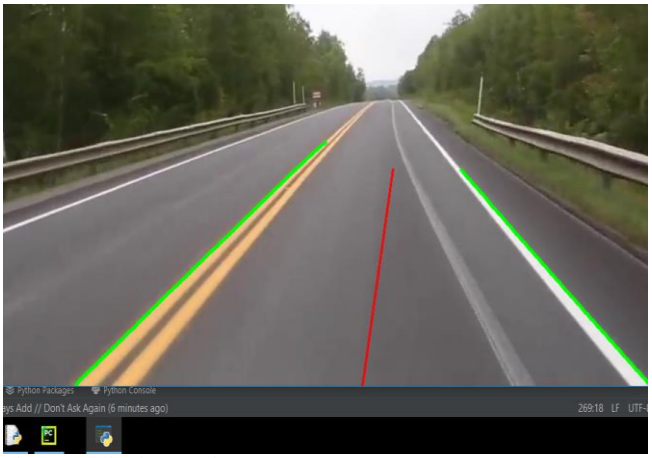


Fig. 5. Result of processing

IV. TESTING AND DISCUSSION

We conducted experimental processing on the robot model. The image was acquired from the camera and sent to the computer embedded in the Raspberry for processing. The

testing process is arranged with blue lines, showing the processing frames, as shown in Figure 6.

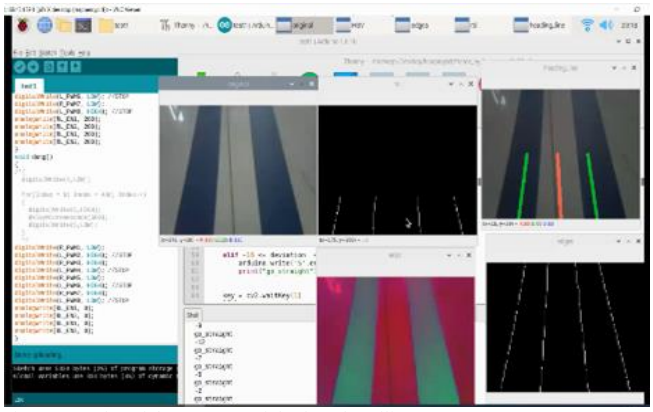


Fig. 6. Results of the test run of the lane handling process

We performed testing using the lane center coordinates, and the processed signals are displayed in Table 1 after comparison with the intended lane center at coordinate angle 0.

TABLE 1. RESULT OF LANE DETECTION

No.	Position edge	Center deviation	Control signals	Note
1	-3	3	Go straight	Left axis deviation
2	-6	6	Go straight	Left axis deviation
3	-5	5	Go straight	Left axis deviation
4	-2	2	Go straight	Left axis deviation
5	10	10	Turn left	Right axis deviation
6	-4	4	Go straight	Left axis deviation
7	15	15	Turn left	Right axis deviation
8	7	2	Go straight	Right axis deviation
9	1	1	Go straight	Right axis deviation
10	-15	15	Turn right	Left axis deviation

Lane edge detection is significant for autonomous vehicle technology. With this paper, we have described important algorithms for handling lane detection and, at the same time, provided a lane center plan, which is considered an essential basis for keeping the vehicle stable in the lane for self-driving cars.

V. CONCLUSION

In this study, we have proposed a lane detection algorithm for self-driving cars in real-world road conditions and dynamic environments. Edge detection is basically the first step in vehicle detection. Using the parameters of the connected components with the edge detection method, we remove noise-detected objects. Once edges are detected, the image is fed to the next module to find the target path for autonomous driving. This method provides information about the vehicle that can be further used to control the vehicle's behavior.

REFERENCES

[1] L. Y. Ma, H. Zhu, and H. Duan, "A Method of Multiple Lane Detection Based on Constraints of Lane Information," in Proc. 2021 China Automation Congress (CAC), pp. 4059–4064, Oct. 2021.

[2] J. Vargas, S. Alsweiss, O. Toker, R. Razdan, and J. Santos, "An Overview of Autonomous Vehicles Sensors and Their Vulnerability to Weather Conditions," Sensors, vol. 21, no. 16, pp. 5397, 2021.

[3] M.Asif, M.R.Arshad, and P.A.Wilson, "AGV Guidance System: An Application of Simple Active Contour for Visual Tracking", International Journal of Computer, vol. 6, no. 2, pp. 664 –667, 2007.

[4] Yue Wang, Eam Khwang Teoh, Dinggang Shen, "Lane detection and tracking using Bsnake", Journal of Image and Vision Computing, vol. 22, no. 1, pp. 269–280, 2004.

[5] Jiang Ruyi, Klette Reinhard, Vaudrey Tobi, Wang Shigang, "Lane detection and tracking using a new lane model and distance transform", Journal of Machine Vision and Application, vol. 22, no. 4, pp. 721–737, 2011.

[6] Yingmao, et.al, "Multiple lane boundary detections using a combination of low-level image features", in Proc. 17th International IEEE Conference on Intelligent Transportation Systems, pp.1682-1687, 2014.

[7] Mohamed Aly, "Real-time Detection of Lane Markers in Urban Streets", International Conference on Intelligent Vehicles Symposium, pp.7-12, 2008.

[8] P. Quoc Thai, H. Duc Tri, B. Van Ga, and P. Van Binh, "Application of Edge Detection Algorithm for Self-Driving Vehicles," in Proc. 2022 7th National Scientific Conference on Applying New Technology in Green Buildings (ATiGB), pp. 225–228, Oct. 2022.

[9] Mark S. Nixon and Alberto S. Aguado, "Feature Extraction and Image Processing", Academic Press, 2008, p. 88.

[10] V. Devane, G. Sahane, H. Khairmode, and G. Datkhile, "Lane Detection Techniques using Image Processing," in Proc. ITM Web Conf., vol. 40, p. 03011, 2021.

[11] W. Farag, "Real-Time Detection of Road Lane-Lines for Autonomous Driving," Recent Patents on Computer Science and Communications, vol. 13, no. 2, pp. 265–274 2019.

[12] W. Rong, Z. Li, W. Zhang, and L. Sun, "An improved Canny edge detection algorithm," in Proc. 2014 IEEE International Conference on Mechatronics and Automation, pp. 577–582, 2014.

[13] T.-A. Nguyen, W.-S. Song, and M.-C. Hong, "Spatially adaptive denoising algorithm for a single image corrupted by Gaussian noise," IEEE Transactions on Consumer Electronics, vol. 56, no. 3, pp. 1610–1615, 2010.

[14] T. D. Pham, "Estimating Parameters of Optimal Average and Adaptive Wiener Filters for Image Restoration with Sequential Gaussian Simulation," IEEE Signal Process. Lett., vol. 22, no. 11, pp. 1950–1954, 2015.

[15] A. A. Assidiq, O. O. Khalifa, R. Islam, and S. Khan, "Real-time lane detection for autonomous vehicles", in Proc. International Conference on Computer and Communication Engineering 2008., pp. 82–88.

[16] B. Ran and H. X. Liu, "Development of Vision-Based Vehicle Detection and Recognition System for Intelligent Vehicles," Transportation Research Record, vol. 1679, no. 1, pp. 130–138.

[17] M. Ragab, "A Survey on Hough Transform, Theory, Techniques and Applications", Accessed: Dec. 08, 2023. [Online].Available: https://www.academia.edu/62448089/A_Survey_on_Hough_Transform_Theory_Techniques_and_Applications.

[18] M. Marzougui, A. Alasiry, Y. Kortli, and J. Baili, "A Lane Tracking Method Based on Progressive Probabilistic Hough Transform," IEEE Access, vol. 8, pp. 84893–84905, 2020.

[19] S. Joy, M. B. S, T. B. Mukesh, M. M. Ahmed, and U. Kiran, "Real Time Road Lane Detection using Computer Vision Techniques in Python," in Proc. 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS), pp. 1228–1232, 2022.