Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

School of Computer Science and Statistics

Analysis of the traditional method for detecting lane lines based on the Hough transform

Yipan Song

A research paper submitted to the University of Dublin,

in partial fulfilment of the requirements for the degree of

Master of Science Interactive Digital Media

2021

Declaration

I have read and I understand the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at: http://www.tcd.ie/calendar

I have also completed the Online Tutorial on avoiding plagiarism 'Ready, Steady, Write', located at http://tcdie.libguides.com/plagiarism/ready-steady-write

I declare that the work described in this research paper is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed: _____

Yipan Song

03/06/2021

Permission to lend and/or copy

I agree that Trinity College Library may lend or copy this research Paper upon request.

Signed: _____

Yipan Song

03/06/2021

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Gerard Lacey, for his patient and excellent suggestions during the writing of this research paper. I would also like to thank my course director, Dr. Mads Haahr, for all the warm help he has given us this year. In addition, I want to express my thanks to my lecturers and classmates. Furthermore, I would like to show my appreciation for my parents, my grandmother and my lovely pet dog.

Abstract

With the development of technology, automatic driving technology has come into being and it has been a critical topic of discussion in recent years. Accelerating the development of technological innovation in smart driving cars is of strategic importance to economic and social development. A prerequisite for several important functions of autonomous driving is that the vehicle must have the judgment and decision-making capability to know its surroundings, its location, and its destination so that it can avoid obstacles and develop a safe driving route. Lane detection is fundamental to the implementation of autonomous driving, as it is crucial to ensure that the car is driving in a defined area and to reduce the chances of traffic accidents, so lane lines detection is currently a key research topic in this area.

Lane line detection methods are divided into traditional methods and deep learning methods. This paper discussed the traditional method based on *Hough* transform. The program in this paper is written in Python based on the open-source computer vision library OpenCV. This paper described in detail the implementation of the traditional method, tested the Caltech dataset using the traditional *Hough* transform method, analysed detected results, discussed the improvements made to it by several scholars, and finally concluded.

The traditional *Hough* transform method is simple to implement and many scholars have improved it to improve the accuracy and speed of operation, but it still has three main shortcomings: It is only applicable to straight lanes, not to curved lanes; the *Hough* transform method is based on the features of the lane lines on the road to obtain lane line information, which is greatly affected by interference information; the algorithm is poorly adapted to the environment, if the parameters are not selected properly, it will easily lead to detection errors.

Table of Contents

List of figures and tables..... 1
List of Abbreviations.....4
Chapter 1. Introduction.....5
1.1 Overview..... 5
1.2 Outline.....7
Chapter 2. Background.....9
2.1 introduction..... 9
2.2 literature Review..... 9
Chapter 3. Methodology and Design.....13
3.1 image preprocessing.....13
3.1.1 Greyscale image conversion.....13
3.1.2 Image Denoising.....15
3.1.3 Edge detection.....25
3.1.4 Extract the ROI.....45
3.2 Lane lane information recognition..... 51
3.3 Least squares based lane line fitting.....57
3.3.1 The principle of least squares.....57
Chapter 4. Implementation, Result and Result Discussion..... 60
4.1 Implementation.....60
4.2 Result.....68
4.3 Result Discussion..... 71
Chapter 5. Improvements and Discussion.....73
Chapter 6. Conclusion and Future direction.....86
6.1 Conclusion.....86
6.2 Future direction..... 87
References..... 88

List of figures and tables

Figure 1: The original image (1)..... 15
Figure 2: The grayscale image..... 15
Figure 3: The template of Median filter..... 17
Figure 4: The image added salt-and-pepper noise.....18
Figure 5: The image with salt-and-pepper noise used median filter (1)..... 18
Figure 6: The image added Gaussian noise (1)..... 19
Figure 7: The image with Gaussian noise used median filter..... 19
Figure 8: The image with Gaussian noise used Gaussian filter..... 19
Figure 9: One-dimensional Gaussian distribution.....20
Figure 10: Two-dimensional Gaussian distribution..... 21
Figure 11: The template of Gaussian filter.....21
Figure 12: The image added Gaussian noise (2)..... 22
Figure 13: The image with Gaussian noise used Gaussian filter (2).....22
Figure 14: The color image added Gaussian noise.....22
Figure 15: The color image with Gaussian noise used Gaussian filter (1).....23
Figure 16: The color image with Gaussian noise used Gaussian filter (2).....23
Figure 17: The image with salt-and-pepper noise used Gaussian filter..... 24
Figure 18: The image with salt-and-pepper noise used median filter (2)..... 24
Figure 19: The Roberts operator..... 26
Figure 20: The image used Roberts operator..... 27
Figure 21: The image used Canny operator..... 27
Figure 22: Prewitt operator.....27
Figure 23: The image used Prewitt operator..... 28
Figure 24: The image with Gaussian noise used Prewitt operator..... 28
Figure 25: Sobel operator..... 29
Figure 26: The image used Sobel operator.....30
Figure 27: The image with Gaussian noise used Sobel operator..... 30
Figure 28: Prewitt operator.....31
Figure 29: The image used Laplacian operator..... 32
Figure 30: The image with Gaussian noise used Laplacian operator.....32
Figure 31: The graph of the LOG operator in space..... 34
Figure 32: The image used LOG operator.....35
Figure 33: The image with Gaussian noise used LOG operator..... 36
Figure 34: The image with salt-and-pepper noise used LOG operator..... 36
Figure 35: The image used DOG operator..... 38
Figure 36: The image with Gaussian noise used DOG operator.....38
Figure 37: The template of Canny operator..... 40
Figure 38: The image used Canny operator..... 41
Figure 39: The image with Gaussian noise used Canny operator (1)..... 41
Figure 40: The image with Gaussian noise used Canny operator (2)..... 42
Figure 41: The image with Gaussian noise used Canny operator (3)..... 42

Figure 42: The image with Gaussian noise used Canny operator (4)..... 43
Figure 43: The image with Gaussian noise used Canny operator (5)..... 43
Table 1: Merit and demerit of operators.....45
Figure 44: The image used Gaussian filter and Canny operator.....45
Figure 45: The detected effect without extracting the ROI and fitting..... 46
Figure 46: The detected effect without extracting the ROI.....47
Figure 47: The ***matplotlib*** figure..... 47
Figure 48: The ROI (1).....48
Figure 49: The image used ROI(1).....48
Figure 50: The image used not suitable ROI and has been fitted.....49
Figure 51: The image used suitable ROI and has been fitted.....49
Figure 52: The ROI (2).....50
Figure 53: The image used ROI (2).....50
Figure 54: The *Hough* transform coordinate system (1)..... 52
Figure 55: The *Hough* transform coordinate system (2)..... 53
Figure 56: The detected image after ***Hough*** transform..... 56
Figure 57: The final detected image.....59
Figure 58: The image without being fitted.....60
Figure 59: The original image (2)..... 61
Figure 60: The image after Canny operator edge detection..... 61
Figure 61: The image after being fitted (1)..... 61
Figure 62: The image after being fitted (2)..... 62
Figure 63: The image after being fitted (3)..... 63
Figure 64: The image after being fitted (4)..... 63
Figure 65: The image after being fitted (5)..... 64
Figure 66: The image after being fitted (6)..... 64
Figure 67: The image after being fitted (7)..... 65
Figure 68: The image after being fitted (8)..... 65
Figure 69: The image after being fitted (9)..... 65
Figure 70: The image after being fitted (10)..... 66
Figure 71: The image after being fitted (11)..... 66
Figure 72: The image after being fitted (12)..... 66
Figure 73: The detected results of cordova1 scene..... 69
Table 2: Accuracy of testing cordova1.....69
Figure 74: The detected results of washington1 scene.....70
Table 3: Accuracy of testing washington1..... 70
Figure 75: The detected results of washington2 scene.....70
Table 3: Accuracy of testing washington2..... 71
Figure 76: The effect of detecting curve lane line (1).....72
Table 4: Accuracy in reference[59].....73
Table 5: Processing time in reference[59].....73
Figure 77: The effect of detecting curve lane line (2).....74
Figure 78: The effect of adjusting the parameters in ***Hough*** transform (1)..... 75
Figure 79: The effect of adjusting the parameters in ***Hough*** transform (2)..... 76

Figure 80: The effect of adjusting the parameters in ***Hough*** transform (3)..... 76
Figure 81: The effect after adjusting the range (1).....77
Figure 82: The effect after adjusting the parameters of range (2).....78
Figure 83: The effect after adjusting the parameters of range (3).....78
Table 6: Error rate comparison in reference[23].....84

List of Abbreviations

ADAS Advanced driver-assistance systems

LDWS Lane departure warning system

ACC Adaptive cruise control

CAS Collision avoidance system

AP Automatic parking

DoG Difference of Gaussian

LoG Laplacian of Gaussian

ROI region of interest

HT *Hough* Transform

Chapter 1. Introduction

1.1 Overview

As a result of social development and economic improvements, urban transport is being optimized and the number of cars is increasing year on year. At the same time, the number of traffic accidents is also on the rise. According to a recent survey of the World Health Organization, approximately 1.35 million people die in road traffic accidents each year. Between 20 and 50 million people suffer from non-fatal harm and many of them are disabled as a result of traffic injuries. In addition, More than half of road traffic deaths are among pedestrians, cyclists, and motorcyclists. Road traffic injuries are the leading cause of death for children and young people aged 5-29. Globally, 93% of road traffic fatalities occur in low- and middle-income countries. With losses of 3% of most countries' GDP, traffic accidents cause substantial financial losses to individuals, families and the nation as a whole. These losses result from the cost of medical treatment, the loss of productiveness of those killed or disabled by the injury, and family members who have to take time out of work or school to take care of the injured person. Surveys have shown that to avoid accidents, drivers should take proactive measures at least 0.5 seconds before a critical situation occurs[1]. How to reduce the number of traffic accidents is a key issue to be addressed.

A new round of global technological revolution and industrial change, with intelligence and networking as important features, is emerging, and the rapid development of artificial intelligence will drive profound changes in human production and lifestyle. With the development of technology, automatic driving technology has come into being, and automatic driving cars have been a hotly debated topic in recent years. Accelerating the development of technological innovation in smart driving cars is of great strategic importance to economic and social development. Autonomous driving technology can be seen as a sophisticated intelligent controlling system that includes several sub-functional modules for

example mechanical control, route mapping, and intelligent sensing. By the interworking of these sub-modules, the vehicle can be driven automatically under the control of a unified system chain [2]. The aim of researching and enhancing automatic driving technology is to enable the development of intelligent urban mobility and to reduce the number of traffic accidents. This technology solves several problems such as urban traffic congestion and accidents, while at the same time providing drivers with a safe, comfortable, and entertaining driving experience. Therefore, the research and implementation of automatic driving technology are of great relevance and have a promising future [3].

In terms of the current development of autonomous driving technology, although there is still a gap from true automatic driving technology, advanced driver-assistance systems (ADAS) has been applied to vehicles, for example, LDWS (Lane departure warning system), ACC (Adaptive cruise control), CAS (Collision avoidance system) and AP (Automatic parking). The main purpose of ADAS is to minimize or avoid driving risks during the driving process, which can effectively prevent or decrease traffic accidents resulting from speeding, forced road changes, drink driving and fatigued driving, and other man-made violations of traffic rules, thus achieving protection for drivers and pedestrians.

When driving on the road, ADAS actively collects information about its state and the surrounding environment through various sensors and processing modules installed in the vehicle, and through the processing of the computing modules, ADAS will alert the driver when there are dangerous factors, such as being too close to the vehicle in front, obstacles in front of the vehicle and the vehicle deviating from the correct driving trajectory, etc., and take safety measures to avoid The sensing module is an integral part of the autonomous driving system

The sensing module is of great significance in autonomous driving systems. It primarily senses the traveling situation while driving, senses vehicles, walkers,

obstructions, and other targets around the vehicle, and provides the sensing results to the route decision-making modules for corresponding route mapping, and finally the machinery controlled modules realize the related machinery controlled operations so that the vehicle can be driven automatically. The environmental sensing system is mainly composed of a series of high precision sensors such as vision sensors, laser radar, and inertial elements, which are used to collect environmental information on road conditions and provide the basis for the control decision system of the automatic driving vehicle, controlling the vehicle's fuel port, braking, steering, and other actuators.

Lane detection plays an important role in the sensing system level of autonomous driving. Lane detection and tracking are considered to be the basic module of ADAS, which directly influences the path planning of autonomous driving or driver assistance. As the first thing to ensure during driving is that the vehicle is driving in a safe area so that it is less likely to collide with other vehicles and pedestrians and other obstacles, lane line detection can guide the direction and area of travel for the vehicle to drive safely on the road. The LDWS (Lane Departure Warning System) in the Advanced Driver Assistance System mainly uses the on-board computer to analyze information from the on-board sensors to locate the vehicle's position regarding the current lane and provide adequate warning to the driver when the driver is unconsciously drifting out of the lane before the vehicle does so. This gives the driver sufficient time to react and significantly reduces accidents caused by lane drift.

1.2 Outline

The paper consists of six chapters, including Introduction, Background, Methodology and Design, Implementation and Result, Improvement and Discussion, and Conclusion and Future direction.

Chapter 1 introduces the significance and purpose of the chosen topic and gives an

overview of the general structure of the paper.

Chapter 2 presents the different improvements made by different scholars to the traditional lane line detection techniques in recent years.

Chapter 3 discusses the traditional lane line detection methods in detail and provides a detailed analysis of the algorithmic principles as well as the advantages and disadvantages of each step.

Chapter 4 tests the Caltech dataset using the traditional *Hough* transform method and analyses the resulting accuracy against the results of other scholars who have tested on the same data using other methods.

Chapter 5 compares and discusses the improvements made by several scholars to the traditional method and the results of the improvements.

Chapter 6 concludes the whole text and presents some perspectives on the field.

Chapter 2. Background

2.1 introduction

The steps of the traditional lane line detection method consist broadly of image pre-processing and lane line detection. The pre-processing of the image includes the conversion of the color image into a greyscale image, noise reduction, edge feature extraction, and extraction of regions of interest. Lane line detection includes feature extraction and model fitting. The *Hough* transform is a mapping from two-dimensional space to parametric space, and it is known from the principle of mathematical duality that the covariance of points in two-dimensional space corresponds to the covariance of curves in parametric space, so it can be used as a basis to extract the relevant parameters of a straight line [12]. However, the *Hough* transform method has many drawbacks. Many scholars have improved it to improve the accuracy and speed of operation.

2.2 literature Review

P. Maya and C. Tharini[59] proposed a detection method which limits the range of the angle θ in the *Hough* transform, which significantly enhances the efficiency of the algorithm, saves detection time and does not lose the accuracy of the algorithm, and they also derive the angular ranges of the left, middle and right lane lines respectively through experimental analysis. However, the article does not perform lane line detection for curved lane lines, and this method is limited to straight lane lines only, which is not very generalizable.

L. Jiang, J. Li, and W. Ai[6] proposed an improved *Hough* transform method to address the shortcomings of the *Hough* transform algorithm in terms of poor real-time performance. They restricted the pole angle size, pole angle direction, and pole diameter, respectively, and twice used the Soble operator to obtain edges of a

single-pixel width, calculated and counted the number of edge points in each direction, and used the directions with the two highest counts of edge points as the approximate lane line directions. The least-squares method is noise sensitive and less robust, which they improved with the R-LSDR algorithm. This method simultaneously removes the sample points with the maximum error in the forward and reverse directions and then updates the equation of the regression line until the percentage of the remaining points to the total amount of sample points reaches a threshold value R. Kalman filter is applied to lane line tracking. Data from the experiments demonstrate that the improved lane detection algorithm enhances the detection accuracy over the conventional method. However, this algorithm is complex and computationally intensive, the time cost of this method is high.

Y. Zhang and J. Yang[7] proposed a road image pre-processing method with the inverse perspective transformation as the core and used the *Hough* transform to detect the lane lines. Due to the perspective, the original parallel lines become intersecting lines, and the interval of dashed lanes becomes small at the distal end of the image, which is not conducive to the detection and discrimination of lines, therefore, the effect of perspective must be removed to obtain accurate road parameters. The inverse perspective transformation is to convert the single image obtained from the monocular camera, with some known parameters, from two-dimensional space to three-dimensional space, and then from three-dimensional space to two-dimensional space, the conversion process using known parameters to remove the perspective effect, to get a two-dimensional image without perspective effect, the converted image is a top view of the road. Their method improves the detection accuracy. However, this method ignores many realistic environmental factors, such as rain and snow. And this method works well for lane lines on a flat road, but if the input lane line image has a certain slope, the lane lines will cross in the transformed image with this method so that when finding the lane lines, errors will occur. Therefore this method has some limitations.

R. K. Satzoda et al[8] proposed to optimize the *Hough* transform by using the *Hough* attribute of addition of *Hough* transform in multiple hierarchies to produce a new *Hough* transform architecture with high running efficiency. However, this method only can detect straight lane lines, not curved lane lines, which is not too practical in practice.

B. Bai et al[9] proposed a method to extract lane lines in images using double-threshold segmentation method, to be specific, they extracted the white and yellow information matching the lane line characteristics in the images by using double-threshold, then established a dynamic search band in the ROI to detect the lane line pixel information, and finally completed the recognition of lane lines by *Hough* transformation. Data show that this algorithm can eliminate the interference well, retain the required pixel information, and recognize the lane lines effectively. However, this algorithm is for the lane lines in the daytime when the they are relatively clear, and it is not applicable to the night environment, which is more affected by environmental factors.

Hsiao et al [18] used the lane line vanishing point as the upper boundary of the ROI in lane line detection, this method extracted the ROI area well. However, in order to determine the location of the vanishing point, strict calibration of the camera was required, as well as strict requirements for the camera mounting position [19].

H. Li et al [11] proposed a two-stage HT to detect straight lines. The experiments show that the algorithm has high accuracy.However, the computation burden of the algorithm is a little heavy, the program runs slowly, so the time cost is high.

Y. Cai and Li. Gao et al[13] proposed an improved lane line detection method with the *Hough* transforms to address the shortcomings of traditional lane line detection algorithms which are susceptible to interference from the external environment. After image pre-processing, feature points containing lane line edges are extracted in

behavioral units, and feature points are combined according to Euclidean distances to establish links between feature points in the longitudinal direction. The feature point extraction reduces the computational effort of the *Hough* transform, the feature point clustering reduces the interference of erroneous feature points, and the straight line equation can be derived without the need to compute all feature points. The algorithm is less computationally intensive than the traditional *Hough* transform, saves memory and computing time, significantly improves interference immunity, and reduces false and missed detection rates, improving correctness and real-time performance. However, it is important that the appropriate feature points are selected in this method, if the selected feature points do not meet the requirements then this the detection result will be affected.

J. Kim et al[20] proposed an algorithm that uses lane shape and color features to identify lanes. This method reduces the influence of distracting information on the road and is fast. However, it relies on lane line features and requires clear lane line colors, so that it is not suitable for the situation that the lane lines are worn or have become obscure, etc.

Chapter 3. Methodology and Design

3.1 image preprocessing

Lane lines are captured by cameras in automatic driving cars and require digital image processing techniques to extract the image information, which means that it should translate the information it carries into digital form to convey its meaning. So this step is an important start. Digital image processing is also known as computer image processing. Digital image processing technology used a number of technical means to convert the information expressed in an image into digital information, a process that is mainly carried out using computer technology. When it was first applied in 1920, it was the processing of images related to the return of spacecraft. Its application has now been extended to agricultural technology, military science, industrial monitoring, and other industries that have a very important impact on people's lives. Digital image processing technology usually includes a digital image collector, image processing computer, camera, image display terminal and other equipment, the effective combination of these devices can think of the whole process of digital image processing. In the process of practical application, digital image processing technology can be combined with electronic technology, intelligent technology, and other high technology included in the intelligent transportation system, to achieve all-round and multi-angle monitoring and management of urban traffic, thus reducing the demand for manpower and material resources for urban traffic management and improving the management effect at the same time.

3.1.1 Greyscale image conversion

This step is to converse the color image to the gray image.

The traditional fixed-weight linear projection greyscale algorithms include the Intensity, Luminance, and Luma algorithm:

(1) Intensity algorithm

The Intensity algorithm converts all three components of a color image to a greyscale image with a weight of 1/3:

$$Gray = (R + G + B) / 3$$

However, greyscale images suffer from serious distortion, because pixels with the same sum of R, G, and B components are mapped to the same grey level, the algorithm results in a greyscale image with only 766 possible colors, compared to the 2563 possible colors of the original image[15].

(2) Luminance algorithm

The Luminance algorithm is the classical weighted average method, which is based on a psychological formula:

$$Gray = 0.299 \times R + 0.587 \times G + 0.114 \times B$$

(3) Luma algorithm

$$Gray = (R \times 2.2 \times 0.2973 + G \times 0.6273 \times 2.2 + B \times 2.2 \times 0.0753)^{1/2.2}$$

This paper used the *cv2.cvtColor* function in the OpenCV library, this function works for color space conversion, which needs to use two parameters: the *"src"* input the original image, this paper used the image from Google search, the RGB image to greyscale image conversion code is *cv2.COLOR_RGB2GRAY.*

The reason for this step is that to detect lane lines, the most critical factor is the gradient. Nowadays a lot of feature extraction is essentially the acquisition and analysis of gradient information, and calculating the gradient requires the use of greyscale images. Because color images are inherently affected by many factors, such as lighting, the key information is hard to extract. Also, after converting a color image

to a greyscale image, the computer becomes less computationally intensive because each pixel in a color image has three color channels and a single-pixel point can have over 16 million (255*255*255) color variations. A greyscale image has only one channel, and its range of variations for a one-pixel point is 255 (The images are from google search).

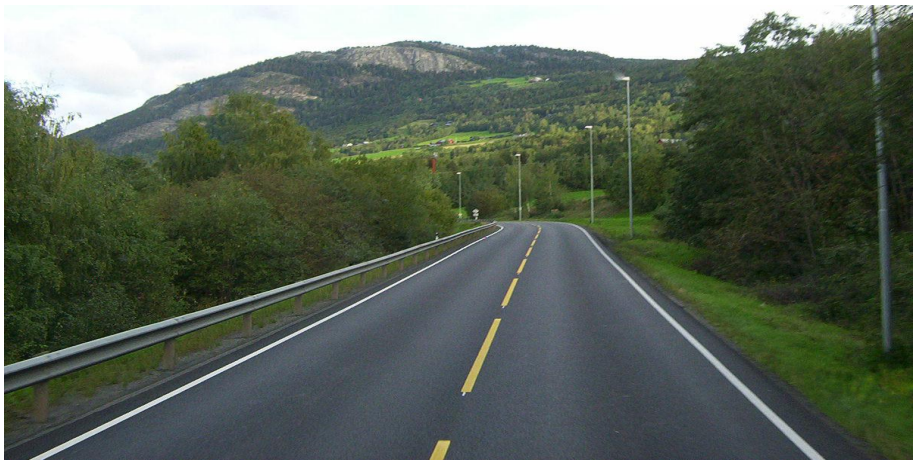

Figure 1: The original image (1)

Figure 2: The grayscale image

3.1.2 Image Denoising

Image denoising is the process of reducing noise in digital images. During automated driving, camera photos of the road are often affected during digitization and transmission by noise interference between the camera and the external environment, for example, known as noisy images or noisy images. Noise is an important cause of

interference with the correct detection of lane lines. An image can have a wide range of noise in practice, which can be generated in transmission or in processing such as quantization.

There are four types of filtering to reduce the noise as follows:

(1) Mean filter

Mean filtering is also known as linear filtering, where every pixel of the image output is the mean (all pixels have the same weight) of the pixels that correspond to the image input from the kernel window, and it is a normalized box filter.

The main method is neighbor averaging. The fundamental principle of linear filtering is to substitute the value of each pixel of the original image, the pixel point currently to be processed (x, y), with the average value, select a template that consists of a number of pixels in its immediate neighborhood, find the average value of all the pixels of the template and allocate this mean value to the current pixel point (x, y) as the gray level g(x, y) of the image to be processed at that point.

$$g(x, y) = \frac{\sum f(x, y)}{m}$$

m: the total number of pixels in this template including the current pixel

The drawback of mean filtering is that it does not protect the image details well and destroys the detailed parts of the image while the image is denoised, thus making the image blurred and not removing the noise points well.

(2) Median filter

Median filter is a non-linear smoothing technique that starts by constructing a

two-dimensional template, which is usually an n×n region (since n×n odd-numbered areas are more convenient to take the centroid and odd-numbered numbers are more convenient to take the median, so n is often an odd number), but of course, it can also be a different shape, such as a line or a circle. The image is processed from front to back, taking all pixel values in the area of the 2D template The center of the two-dimensional template is replaced by the median of all pixel values in the area of the two-dimensional template, then moved backward by one-pixel unit, and finally repeated until all the centroids of the image are covered. This operation is repeated until all the pixel values of the covered centroids have been replaced by the median value[24].

The output of the 2D median filter is:

$$g(x,y)=\text{med}\{f(x-k,y-l) \quad (k,l \in W)\}$$

W: a two-dimensional template, usually a 3×3, 5×5 region, or different shapes such as lines, circles, crosses, circles, etc.

The matrix is used to represent a greyscale image, and the 2D template is taken as a 3 x 3 area.

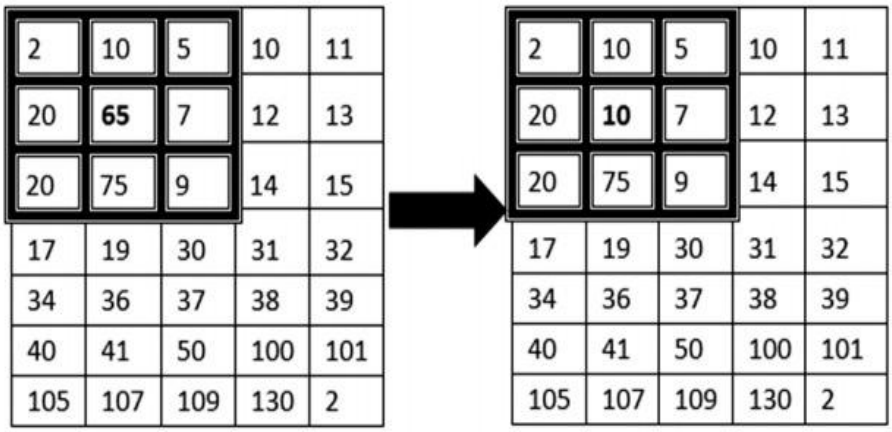

Figure 3: The template of Median filter

Median filter is very good for removing point noise, especially pepper noise.

The following image has been added the salt-and-pepper noise, the SNR is 0.6:

Figure 4: The image added salt-and-pepper noise

The following image has been used median filter to reduce the noise (kernel size is 5×5):

Figure 5: The image with salt-and-pepper noise used median filter (1)

The disadvantage of median filtering is that it is not effective in removing noise in large areas and it tends to cause discontinuities in the image, and the effect that reducing the Gaussian noise is not better than Gaussian filter.

The following image has been added Gaussian noise:

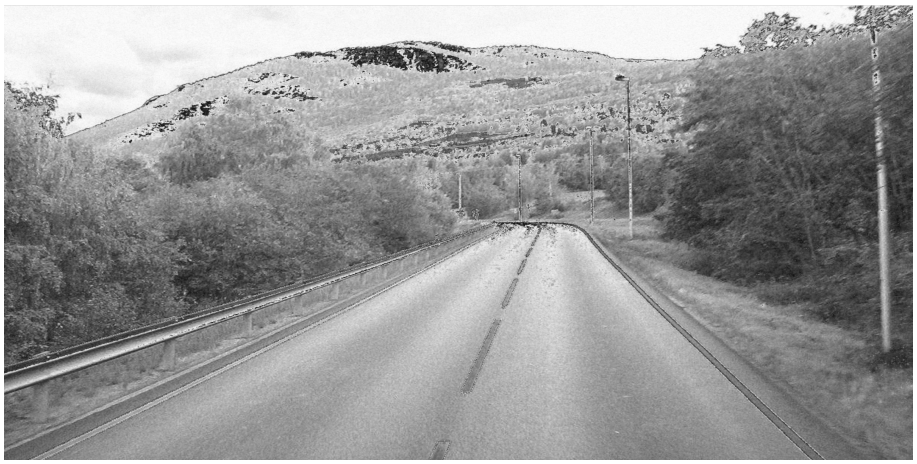

Figure 6: The image added Gaussian noise (1)

The following image has been used median filter (kernel size is 5×5):

Figure 7: The image with Gaussian noise used median filter

The following image has been used Gaussian filter (kernel size is 5×5):

Figure 8: The image with Gaussian noise used Gaussian filter

(3) Gaussian filter

The Gaussian filter is a linear smoothing filter suitable for removing Gaussian noise and used extensively in the noise removal process for image processing. It is a result of a process of weighted averaging over the whole image, where the value of every pixel is derived from a weighted average of its value and the values of the remaining pixels in its neighborhood. Gaussian filtering operates by scanning every pixel in the image with a template (or convolution, mask) and substituting the pixel value at the center of the template with the value of the weighted average grey level of the neighboring pixels determined from the template.

One-dimensional Gaussian distribution

$$G(x, y) = (1/2\pi\sigma)e^{-x^2/2\sigma^3}$$

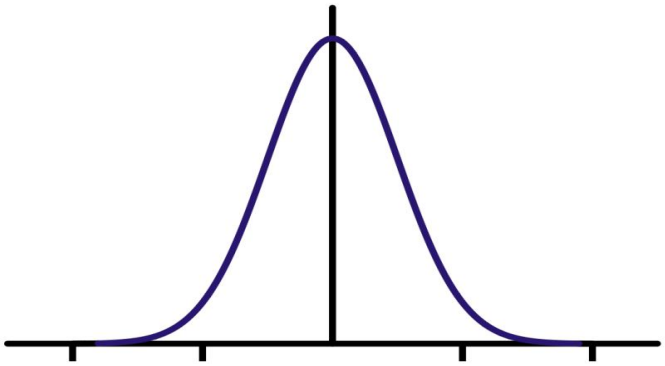

Figure 9: One-dimensional Gaussian distribution

Two-dimensional Gaussian distribution

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

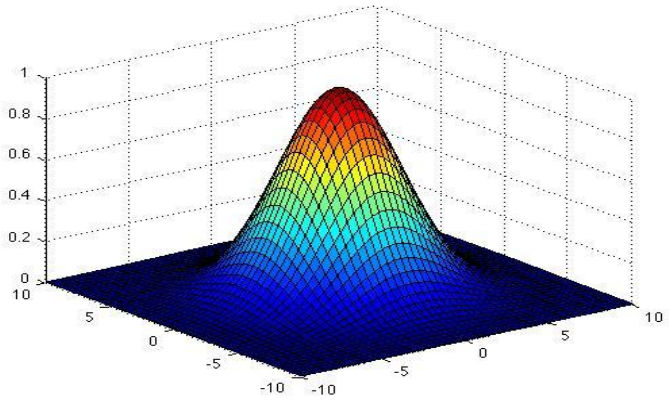

Figure 10: Two-dimensional Gaussian distribution

Since the frequency response of a Gaussian filter is independent of the pixel size of the image and depends only on the distance between pixels in the image, the Gaussian filter can be filtered quickly through the fast Fourier transform. As a result, the computational and operational effort of a two The computational effort and operation time of the 2D Gaussian filter are relatively constant and are not affected by the mask window size. The two commonly used Gaussian templates are shown below. On the left is the commonly used 3*3 Gaussian template and on the right is the commonly used 5*5 Gaussian template：

$$\frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad \frac{1}{273} \times \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Figure 11: The template of Gaussian filter

Gaussian noise is one of the most dominant types of image captured by the equipment and can produce impurities that are difficult to remove, and its removal is a priority in image processing[24]. So Gaussian filter is generally used in conventional lane line detection to remove noise. The following image has been added Gaussian noise:

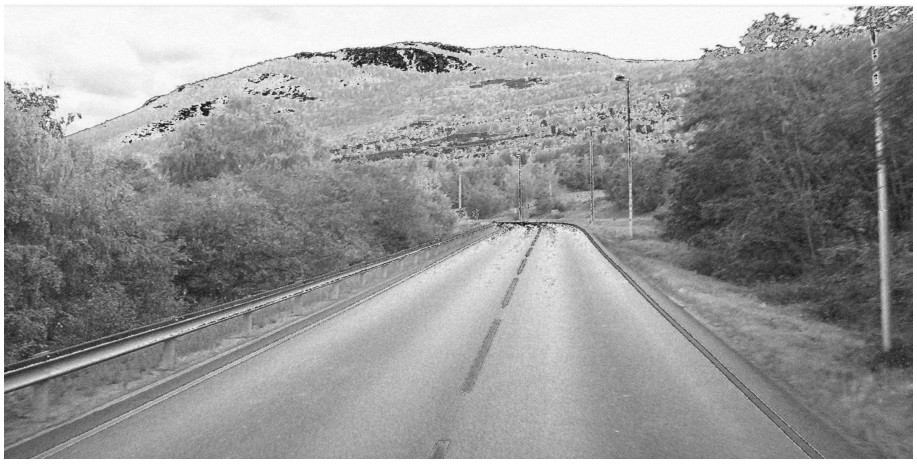

Figure 12: The image added Gaussian noise (2)

The following image has been used Gaussian filter (kernel size is 9×9):

Figure 13: The image with Gaussian noise used Gaussian filter (2)

The contrast in color images will look more obvious:

Figure 14: The color image added Gaussian noise

The following image had been added Gaussian filter, the kernel size is 9×9:

Figure 15: The color image with Gaussian noise used Gaussian filter (1)

The following image had been added Gaussian filter, the kernel size is 15×15:

Figure 16: The color image with Gaussian noise used Gaussian filter (2)

The size of the kernel is bigger, the intensity of blur is greater correspondingly. But it is not true the larger size of the kernel is better, because it will blur the critical edge too and cause the loss of the important information of the image. This time we choose the kernel of size 5×5.

But Gaussian filtering also has its drawbacks, it is very unsatisfactory for pepper noise removal, the effect is much worse than Median filter.

The first figure below shows the removal effect of salt-and-pepper noise by Gaussian filter, the second figure shows the removal effect of salt-and-pepper noise with median filter (Both figures are added the salt-and-pepper noise of SNR 0.6)：

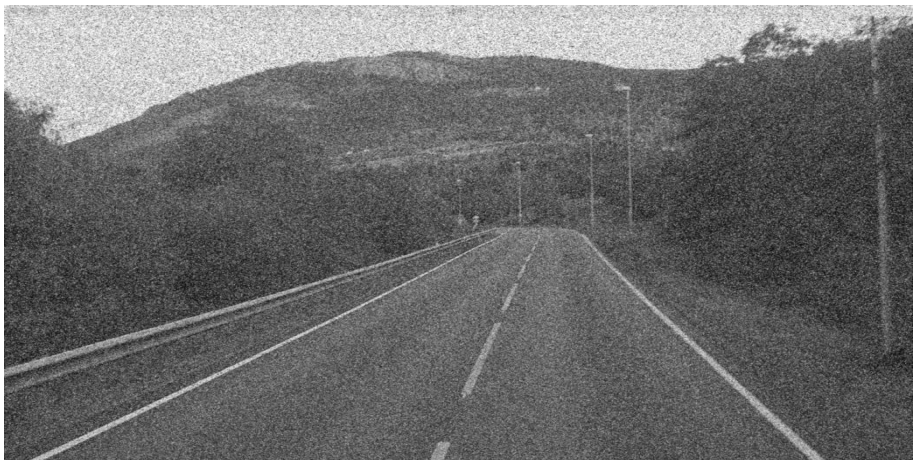

Figure 17: The image with salt-and-pepper noise used Gaussian filter

Figure 18: The image with salt-and-pepper noise used median filter (2)

3.1.3 Edge detection

The purpose of edge detection is to identify points in a digital image where there is a significant change in brightness.

Edges in an image have both direction and magnitude properties. The greyscale changes along the direction of the edge are flat, while perpendicular to the direction of the edge, the greyscale changes dramatically.

The theory for image edge detection is to use the certain kind of algorithm to extract the boundaries between target and target, target and image background, which can be divided into the conventional edge detection algorithm and the modern edge detection algorithm. Conventional edge detection algorithms are mainly applied to 2D images of a specified type. Because pixel points are straightway involved in the differentiation process, conventional algorithms are more susceptible to noise points, in particular the higher-order differentiation above the third order, which tends to produce blurred and broken edges. Modern edge detection algorithms are an expansion of conventional algorithms with good noise immunity and accurate localization accuracy and are particularly suitable for detection where high detail of edge features is required, and sub-pixel detecting accuracy can be achieved in such applications as dimensional analyses and visual localization. The results of edge detection can be evaluated according to three traditional metrics proposed by Canny, which are the single-edge response, localization accuracy and signal-to-noise ratio[25]-[30].

First-order differential operators

First-order differential operators are generally accomplished by convolution operations with the aid of null differential operators. The main first-order differential operators include the Roberts, the Sobel operators[31],[32], the isotropic Sobel

operator, the Prewitt operator, the Kirsch operator, and the Robinson operator based on it[25].

(1) Roberts operator

The Roberts algorithm detects the edges of an image based on the difference between the gray values of four adjacent diagonally positioned pixels in the image localization, without any special smoothing operation, with the disadvantage that its calculation is based on diagonal pixel points, ignoring the vertical and horizontal directions, which can easily lead to missed detection, less accurate positioning of edges, coarser lines of extracted edges, so it detects image edges that are not very smooth and usually detects wide edges that require further refinement, adding to the workload[35], another point that can't be ignored is that it is sensitivity to noise. It is suitable for image segmentation where the edges are obvious and the noise is low. As can be seen, the Roberts operator is more suitable for steep, low-noise images[30].

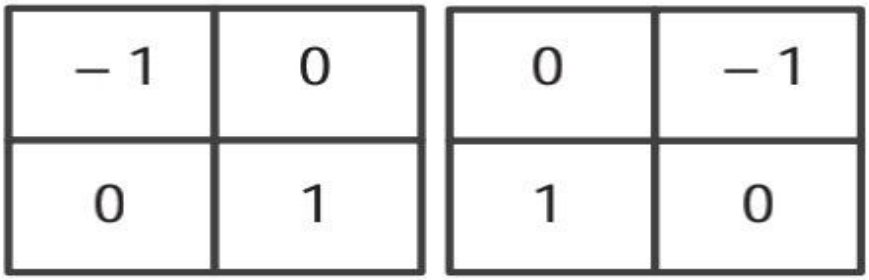

Figure 19: The Roberts operator

The following first image is used Robert operator to detect the edge, the following second image is used Canny operator. By comparing the two images, we can see that the edges of the lines detected in the first image are rougher.

Figure 20: The image used Roberts operator

Figure 21: The image used Canny operator

(2) Prewitt operator

The Prewitt operator has some suppression of noise effect, but it should be artificially selected thresholds, the detected edge lines are sparse, high computational effort and edge localization accuracy are not high[33]. The following figure[25] shows the 3 x 3 convolutional template (Horizontal, vertical and diagonal directions respectively):

−1	−1	−1	−1	0	1	0	1	1
0	0	0	−1	0	1	−1	0	1
1	1	1	−1	0	1	−1	−1	0

Figure 22: Prewitt operator

The following figure shows the effect that using the Prewitt operator:

Figure 23: The image used Prewitt operator

The following figure which has been added Gaussian noise with 0.001 variances, the result is affected by noise:

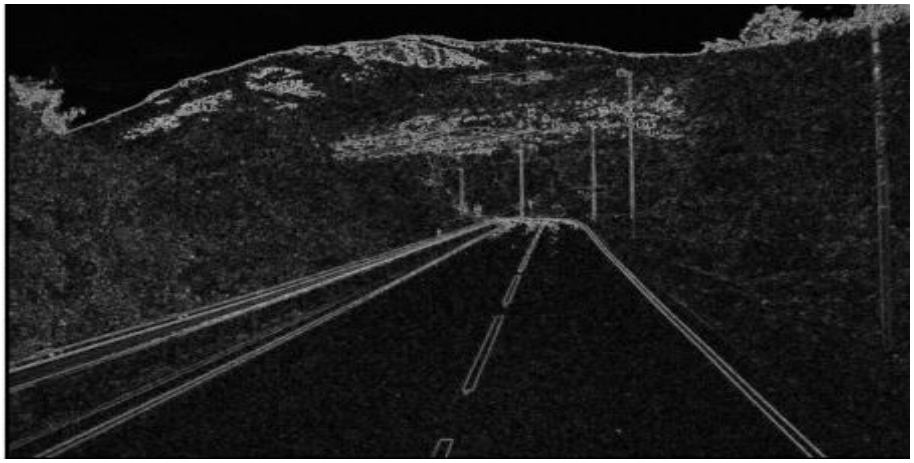

Figure 24: The image with Gaussian noise used Prewitt operator

(3) Sobel operator

The Sobel operator is a discrete difference operator that calculates a result showing the changing sharpness of an image at a given pixel point. It is implemented by

sliding two different convolution kernels over the image and doing a convolution operation to detect the horizontal and vertical edges of the image respectively.

For a point in the image, a 3 × 3 region is formed by taking eight surrounding pixel points at its center, multiplying them with each point of the convolution kernel in Figure 1, and summing them to the new value of the point. For points with incomplete surrounding elements around the edges of the image, the missing part is treated as 0. For points (i, j) with[34] :

$$\begin{cases} s_i = \sum_{m=-1n=-1}^{1} I(m,n)h(i-m,j-n) \\ s_j = \sum_{m=-1n=-1}^{1} J(m,n)h(i-m,j-n) \end{cases}$$

$$S(i,j) = \sqrt{s_i^2 + s_j^2}$$

I(m, n) and J(m, n) are the 3 × 3 horizontal and vertical detection templates of the Sobel operator, respectively;

h(i, j) is the image processed by the Sobel operator;

Si and Sj are the degrees of the image in the horizontal and vertical directions respectively.

However, the Sobel algorithm also has some shortcomings, it does not distinguish between the subject and the background of the image and is not very accurate in locating edges[35].

The following figure shows the 3 x 3 convolutional template of Sobel operator (Horizontal, vertical and diagonal directions respectively):

−1	−2	−1	−1	0	1	0	1	2
0	0	0	−2	0	2	−1	0	1
1	2	1	−1	0	1	−2	−1	0

Figure 25: Sobel operator[25]

The following figures show the effect that using the Sobel operator:

Figure 26: The image used Sobel operator

The following figure has been added Gaussian noise with 0.001 variances, in which we can see that the result of Sobel operation detection is affected by noise:

Figure 27: The image with Gaussian noise used Sobel operator

The results of the Prewitt and Sobel operators are very sensitive to noise. And as concluded from the paper by Chao Zhou[47], the Prewitt operator is better if the noise at each point is the same. If the noise near the edges is 2 times greater than the noise along the edges then the Sobel operator is preferable. So the choice between the Prewitt and Sobel operators depends mainly on the structure of the noise.

Second-order differential operators

For the case where first-order differential detection algorithms may fail to detect edges when processing images with uniform greyscale, second-order differential operations are effective in finding image details. Common second-order differential operators include Laplacian, LOG, DOG, and Canny operators[25].

(1) Laplacian operator

This operator works better for no noisy images, as the noisy case may respond more strongly to noise points and thus ignore the edge points, resulting in less effective edge detection. Therefore If the image is noisy, this operator cannot be used directly but must be filtered first. The greatest advantage of the Laplacian algorithm over other algorithms is that the Laplacian algorithm is directionless and has the same effect on the detection of edges in all directions, so the detection of edges is more accurate[35].

The following figure shows the 3 x 3 convolutional template of Laplacian operator, it is isotropic, sensitive to changes in greyscale, and poor noise immunity：

0	– 1	0
– 1	4	– 1
0	– 1	0

– 1	– 1	– 1
– 1	8	– 1
– 1	– 1	– 1

1	4	1
4	– 20	4
1	4	1

Figure 28: Prewitt operator

The following first figure shows the effect of Laplacian operator using the original image, the following second figure shows the effect of Laplacian operator using the image which has been added Gaussian noise:

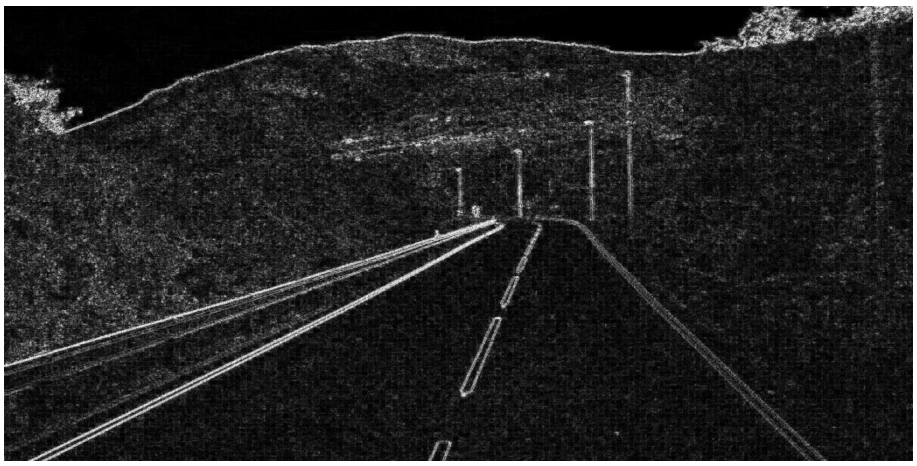

Figure 29: The image used Laplacian operator

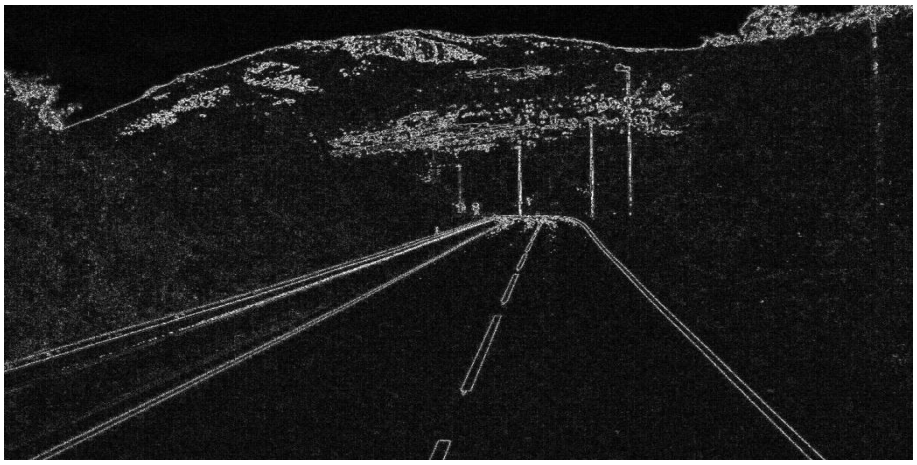

Figure 30: The image with Gaussian noise used Laplacian operator

It can be seen that it is susceptible to be affected by Gaussian noise. Therefore the Laplacian operator is generally used for edge localization and determining where a pixel point is located in a bright or dark region, and it is not directly used for edge detection.

(2) LOG operator

The LOG algorithm proposed by Marr[23],[24] is one of the better edge detection algorithms available. It has been widely used.

It is derived from Marr's vision theory of edge extraction, in which the original image is first optimally smoothed to achieve maximum noise suppression, and the smoothed image is then edge-smoothed.

The LoG algorithm combines a Laplacian sharpening filter and a Gaussian smoothing filter, which first smooths the image to remove noise, and then performs edge detection, which has a certain suppression effect on noise.

When extracting edge information with the LoG algorithm, the image is first smoothed, then the Laplace operator is applied to perform edge detection on the smoothed image, and finally, the edge image is obtained, the specific procedure is shown below.

Assume that the original image is f (x, y), smoothing it to obtain the image g (x, y):

$$g(x,y) = h(x,y) \times f(x,y)$$

h(x, y) is a smoothing function:

$$h(x,y) = \frac{1}{2\pi\delta^2} \mathrm{e}^{-\frac{x^2+y^2}{2\delta^2}}$$

The smoothed image g(x, y) is blurred compared to the original image f(x, y), with the standard deviation δ determining the degree of blurring. The Laplace operator is again used for edge detection of g(x, y) detection:

$$\nabla^2 g(x,y) = \nabla^2 [h(x,y) \times f(x,y)] =$$
$$\nabla^2 h(x,y) \times f(x,y),$$

$\nabla^2 g(x,y)$ is the Laplace operator,

$$\nabla^2 h(x,y) = \frac{\partial^2 h(x,y)}{\partial^2 x} + \frac{\partial^2 h(x,y)}{\partial^2 y} = \frac{1}{\pi\delta^4}\left(\frac{x^2+y^2}{2\delta^2} - 1\right) e^{-\frac{x^2+y^2}{2\delta^2}}$$

The graph of the LOG operator in space is shown in the following figure:

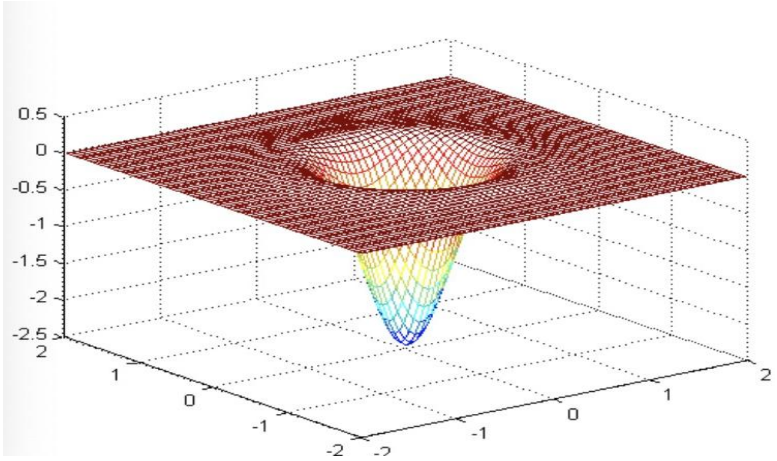

Figure 31: The graph of the LOG operator in space

As can be seen from the diagram, the LOG operator is a bandpass filter because of its resemblance to a Mexican straw hat. It is also known as the Mexican straw hat operator. The study shows that the LOG operator is more compatible with human visual properties[47].

However, in practice the LOG operator has some shortcomings:

a) When the traditional LOG edge detection algorithm used Gaussian function filtering, although it suppresses the noise, it also corrupts some of the low-intensity edges.

b) The algorithm is unable to eliminate the pretzel noise in the image.

c) The noise smoothing ability of the LOG operator contradicts the edge localization positioning ability of the LOG operator contradicts.

d) Scale factor cannot be adaptively adjusted.

e) The size of the template has a strong influence on the detection results.

f) Zero crossing results cannot distinguish the size of the pixel contrast.

g) In practice, the LOG operator is still noise sensitive and the noise smoothing capability contradicts the edge localization capability.

The following first figure is the effect of LOG operator using the original image, the following second figure is the effect of LOG operator using the image which has been added Gaussian noise with 0.001 variances, the third figure shows the removal effect of LOG operator using on the image which has been added salt-and-pepper noise (SNR is 0.6):

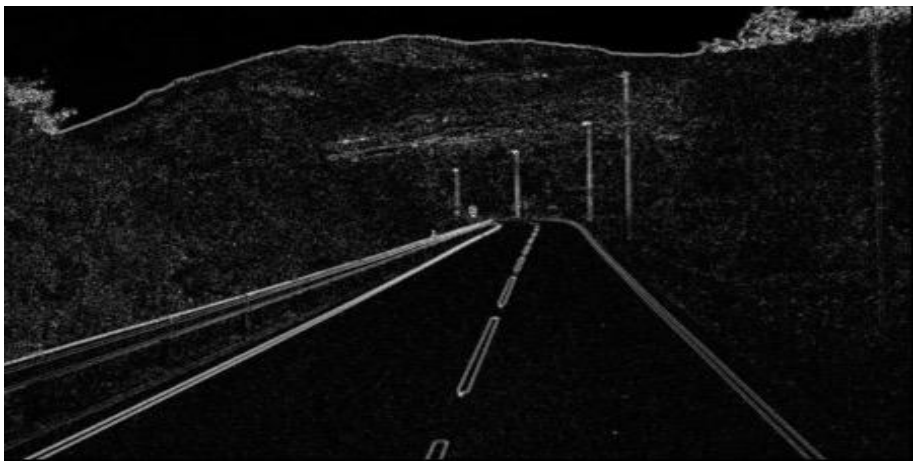

Figure 32: The image used LOG operator

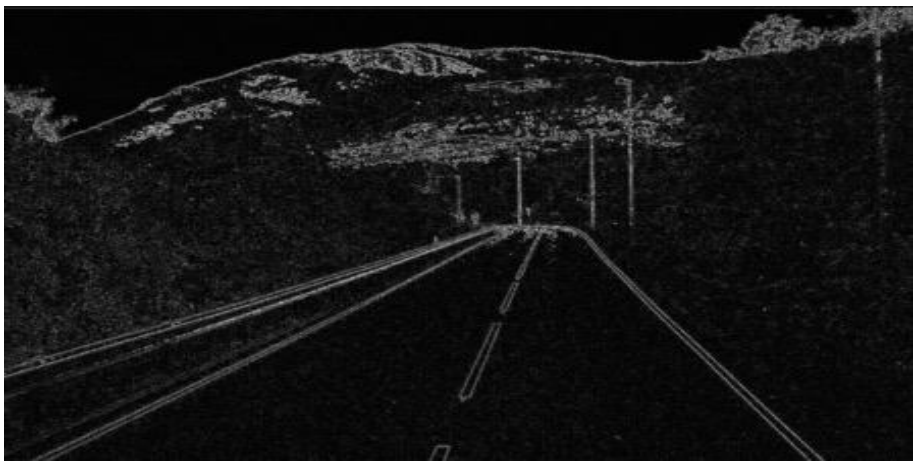

Figure 33: The image with Gaussian noise used LOG operator

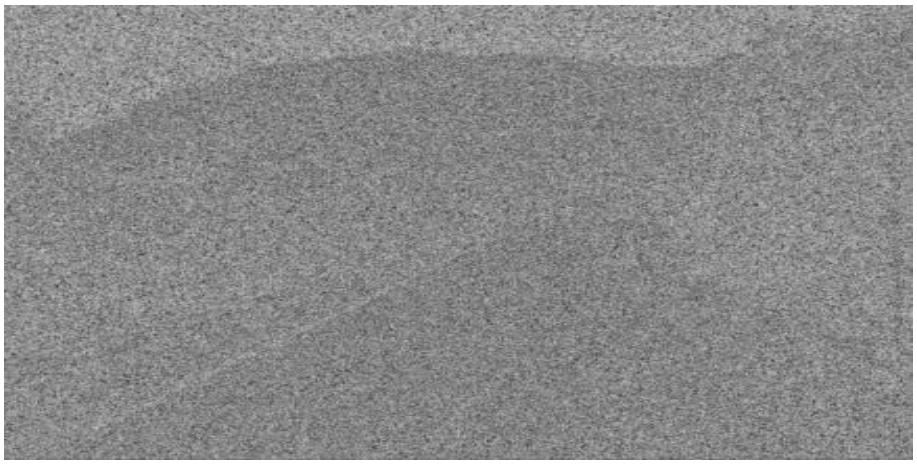

Figure 34: The image with salt-and-pepper noise used LOG operator

As we can see from the plots above, although it is more resistant to Gaussian noise than the previous operator, the lines are still not very good and the results are extremely poor for images with salt-and-pepper noise added.

(3) DOG operator

The Difference of Gaussian operator obtains the DoG response map by The response image of DoG is obtained by subtracting the images of two adjacent Gaussian scale-spaces image.

The difference of Gaussians (DOG) operator is an approximation to the Gaussian Laplace operator (LOG). The LOG operator is a second-order differential operator The LOG operator is a second-order differential operator with good edge response, but the construction of the operator requires a computationally intensive Laplace transform of the two-dimensional Gaussian function. The DOG operator is a good approximation to the LOG operator and is computationally intensive. The DOG operator is a good approximation to the LOG operator and is less computationally intensive while maintaining a good The DOG algorithm is a good approximation to the LOG algorithm and is less computationally intensive while maintaining a good edge response[40],[41].

DoG (Difference of Gaussian) is the difference of a Gaussian function, which is a normally distributed function. In digital images, the formation of image edges is usually due to sudden changes in the grey value of neighboring pixels in the image. By filtering the image with different parameters, an image with different grey values of the edges can be obtained, which can be subtracted to obtain the response image of the DoG operator.

The response image of the DoG algorithm can be obtained by subtracting the response image of the DoG algorithm and then obtaining the edge of the image [42],[43].
DoG operator can be expressed as:

$$\mathrm{DoG} \triangleq G_{\sigma_1}(x,y) - G_{\sigma_2}(x,y) = \frac{1}{\sqrt{2\pi}}\left(\frac{1}{\sigma_1}\mathrm{e}^{-\left(x^2+y^2\right)/2{\sigma_1}^2} - \frac{1}{\sigma_2}\mathrm{e}^{-\left(x^2+y^2\right)/2{\sigma_2}^2}\right)$$

Jilei Liu[44] used the DoG filter to detect human activity trajectories on the ice surface, and the average target object extraction accuracy could reach 82.29%, but the filter parameters needed to be manually tuned[25].

The following figure is used DoG operator, which can be seen that the detected edges are complete and continuous:

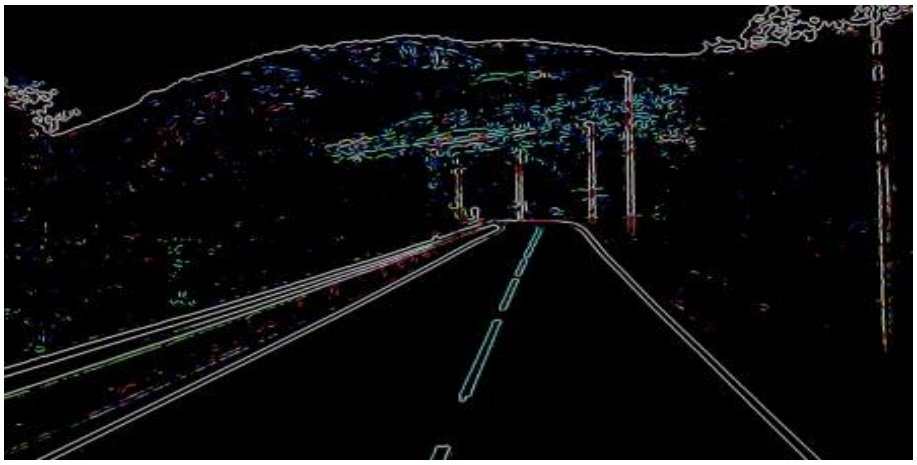

Figure 35: The image used DOG operator

The following figure has been added Gaussian noise with 0.001 variances, in which we can see that DoG operation is also sensitive to noise, but not too serious:

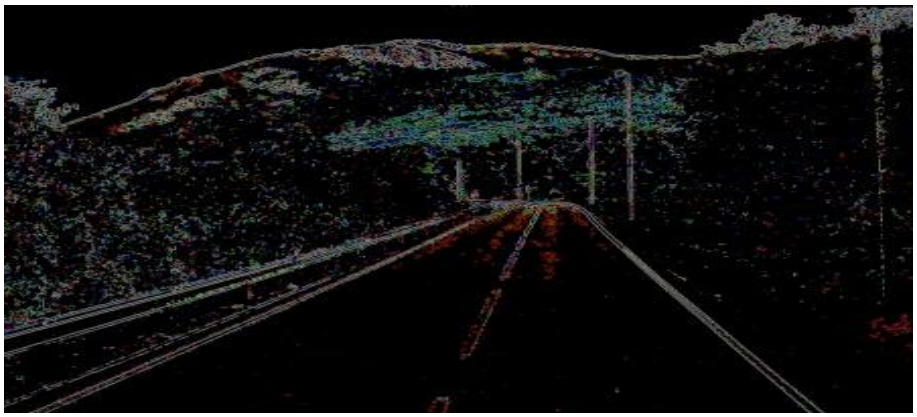

Figure 36: The image with Gaussian noise used DOG operator

The target object edge detection in the image with suitable filter parameters is better than that of the LoG operator. At the same time, the background image is more affected by the filtering[25], and if it is over-smoothed, the edge detection result is worse than that of the LoG operator[45].

(4) Canny operator

The Canny edge detection operator was proposed in 1986 in Canny's master's thesis at MIT[46]. The Canny operator is a multi-stage optimization operator with filtering,

enhancement as well as detection, which has good edge detection. It has been used more and more widely.

The Canny operator, which is similar to the LoG edge detection method, the Canny edge detection algorithm maximizes the identification of the true edges of the image and removes noise interference[49]. Canny edge detection is a common edge detection method for lane line detection in conventional lane line detection.

This part of this paper used Canny edge detection to extract lane line edges.

There are three criteria to judge the performance of edge detection:

a) The signal-to-noise ratio criterion:

The main aim of the S/N standard is for the accuracy of edge detection to be enhanced to reduce the rate of false or missed edge errors.

b) The best positioning criterion:

It aims to make the marked edges as close as possible to the true edges of the images, thereby enhancing positioning accuracy.

c) The single-edge response criterion:

It is used to secure a single pixel responses on a single edge, in order to maximise positioning accuracy and minimise spurious edge responses.

$A(i-1, j+1)$	$A(i, j+1)$	$A(i+1, j+1)$
$A(i-1, j)$	$A(i, j)$	$A(i+1, j)$
$A(i-1, j-1)$	$A(i, j-1)$	$A(i+1, j-1)$

Figure 37: The template of Canny operator

The traditional method used a Gaussian filter to smooth the noise of the image, due to the fixed area of pixels with large differences in grey values, the use of Gaussian filtering tends to make the image The edges of the image are blurred, resulting in a lot of details missing from the edge image, and the Canny The selection of the threshold parameters of the Canny operator does not rely on the feature information of the image edges. The threshold parameters of the Canny algorithm are not based on the feature information of the image edges but need to be set artificially, so they are poorly adaptable and easy to cause This makes it difficult to adapt to the situation and may lead to missed detection or pseudo-edges.

The following figure is used Canny operator (The high and low thresholds are 50 and 150 respectively), in which the edges have been detected clearly (this figure has not used extra blur or denoising, in traditional lane line detection, we should reduce the noise once before the edge detection):

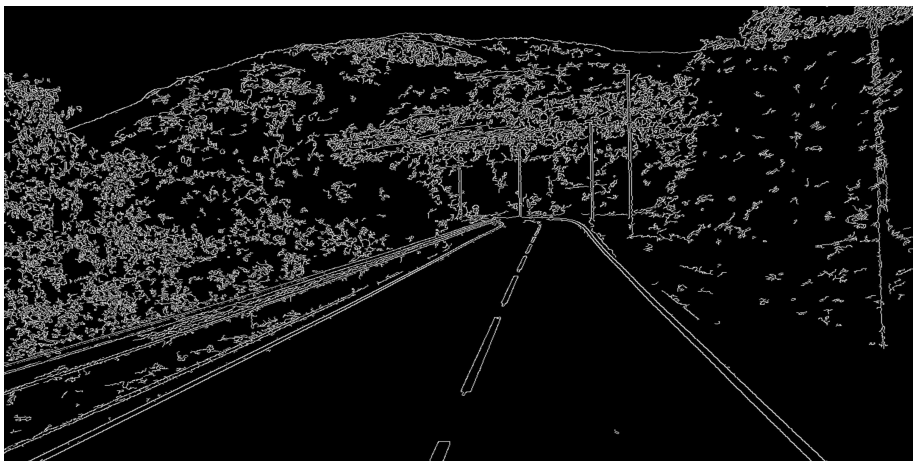

Figure 38: The image used Canny operator

The following figure has been added Gaussian noise with 0.001 variances, in which we can see that Canny operation is very sensitive to noise:

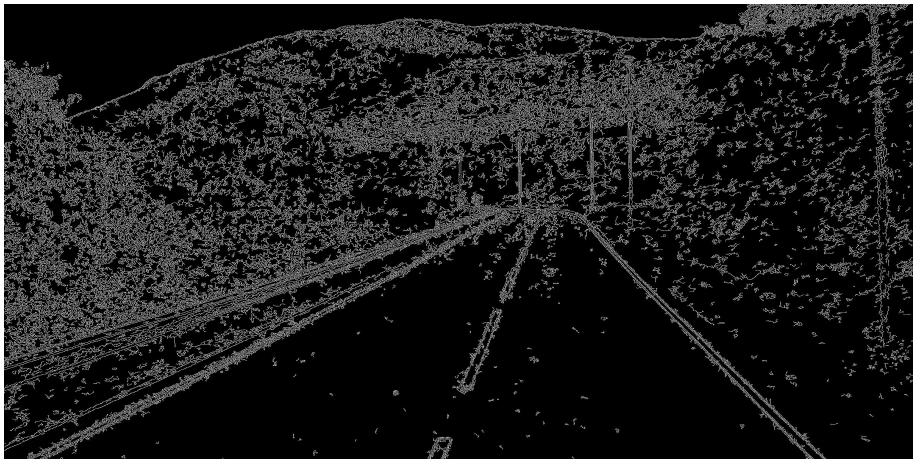

Figure 39: The image with Gaussian noise used Canny operator (1)

The high and low thresholds in the Canny algorithm are determined artificially and cannot be combined with the image itself. The high and low thresholds in the Canny algorithm are determined artificially and cannot be combined with the specificity of the image itself to avoid human interference. Different images are segmented using the same thresholds. This is why there are significant differences in detection results.
The following figure is used Canny operator (The high and low thresholds are 100 and 150 respectively):

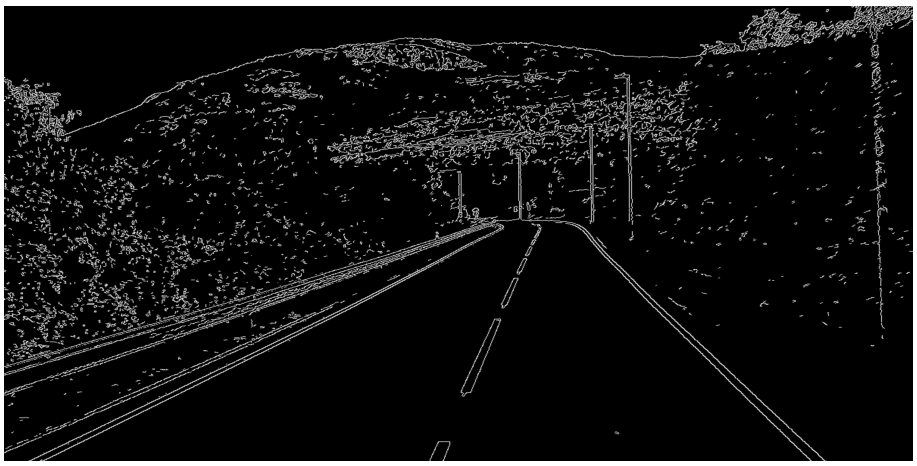

Figure 40: The image with Gaussian noise used Canny operator (2)

The following figure is used Canny operator (The high and low thresholds are 100 and 200 respectively):

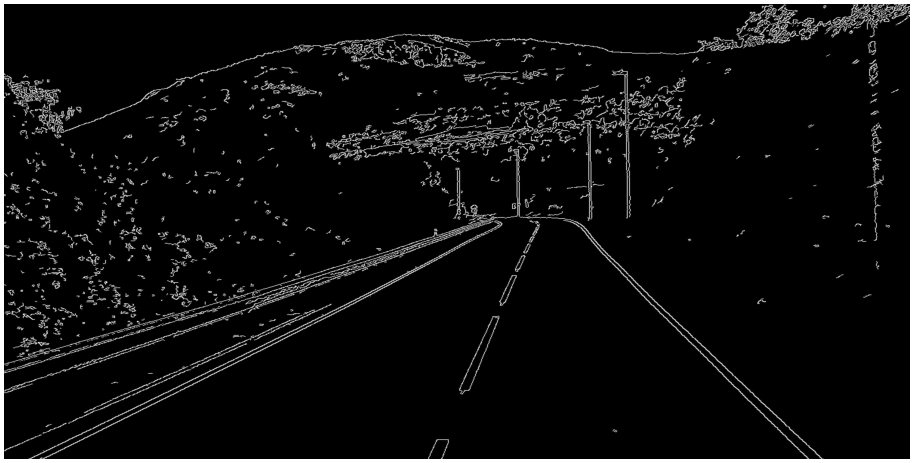

Figure 41: The image with Gaussian noise used Canny operator (3)

The following figure is used Canny operator (The high and low thresholds are 150 and 250 respectively):

Figure 42: The image with Gaussian noise used Canny operator (4)

The following figure is used Canny operator (The high and low thresholds are 200 and 250 respectively):

Figure 43: The image with Gaussian noise used Canny operator (5)

As we can see from the images above, as the threshold increases, the edges of the image also tend to be visible, as many stray points are filtered out.

The images captured by cameras in real-world autonomous driving are susceptible to interference from illumination, stray light, and various other noises, recording images with varying degrees of high and low contrast and noise. Under these circumstances, it is difficult to determine the high and low threshold values for edge detection by the Canny algorithm; on the other hand, using the same threshold ratio for different

images may result in the loss of edge information or the appearance of pseudo-edges, etc[51].

Method	Merit	Demerit
Roberts	Good detection effect of low noise images with the sensitive first-order response. Detects horizontal and vertical edges better than diagonal edges.	More sensitive to noise The detected edges are rough and easy to miss some edges.
Sobel	Has a certain degree of noise suppression, better edge detection in horizontal and vertical directions than in other directions, better processing of grey-scale gradients, and higher positioning accuracy.	Easy to cause a partial loss of edge information The threshold parameter of the operator is not adaptive. The ability to suppress noise is not strong, and at the same time some detailed information will be lost so that the edges have some blurring
Prewitt	Better processing of grey-scale gradient images with noise suppression	High computational effort and poor edge positioning accuracy
Laplacian	Accurate localization of step-edge points	Sensitive to noise, sometimes the edges found are Pseudo-edges
LoG	High edge positioning accuracy	Discontinuous detection edges, prone to pseudo-edges, and very sensitive to noise. The size of the Gaussian coefficient σ directly affects the performance of the LOG operator edge detection algorithm, so it is difficult to make a reasonable trade-off between edge and noise in practice.

DoG	Good noise suppression, continuous edge detection, rich detail and high edge positioning accuracy.	The filter parameters need to be tuned manually, while the background image is influenced by the filtering, and if If it is over-smoothed, the edge detection result is worse than the LoG operator.
Canny	Good edge detection, fine detected edges, and good detection accuracy	Poor resistance to noise interference, more false edges and blurred edges, light source noise and custom thresholds can produce edge detection instability

Table 1: Merit and demerit of operators

Every edge detection method has its shortcomings. This paper used Gaussian filter to blur images and then use Canny operator to detect the edges, the size of Gaussian kernel is 5×5 and the high and low thresholds of Canny operator are 50 and 150 respectively (As shown in the following figure):

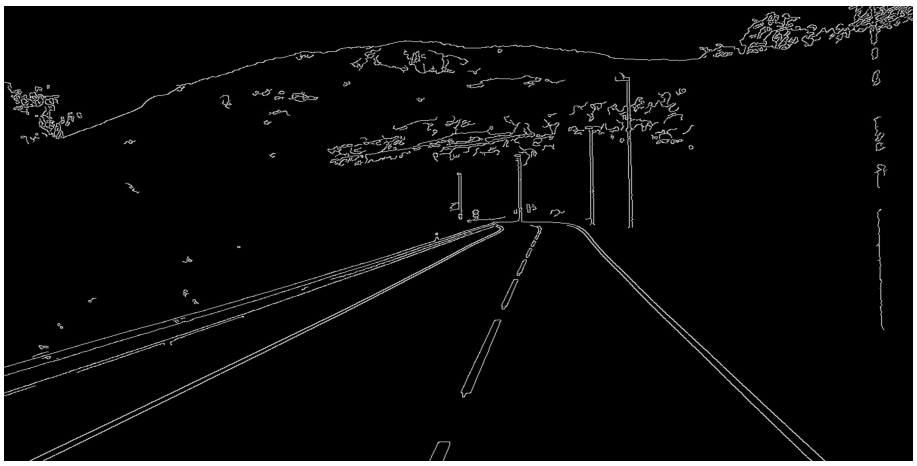

Figure 44: The image used Gaussian filter and Canny operator

3.1.4 Extract the ROI

By analyzing the lane images captured by the vehicle-mounted camera, it was found that the captured map had data information including housed, trees, guardrails, and the sky ahead on both sides of the lane, which basically did not contain lane

information. If the images were processed directly, this would increase the complexity of the detection algorithm and reduce the efficiency of lane detection, and the extraneous information could interfere with the accurate extraction of lane lines. Therefore, before further processing of road images of complex driving environments, the road images need to be extracted according to the current needs, only the regions we need to focus on, ignoring the parts containing useless interference information, and then proceed to the subsequent processing, we call this process region of interest (ROI) extraction[52].

If an image detected directly without extracting the ROI, there are so many useless lines to be marked, just like the following image, the lines of this following image were not fitted :

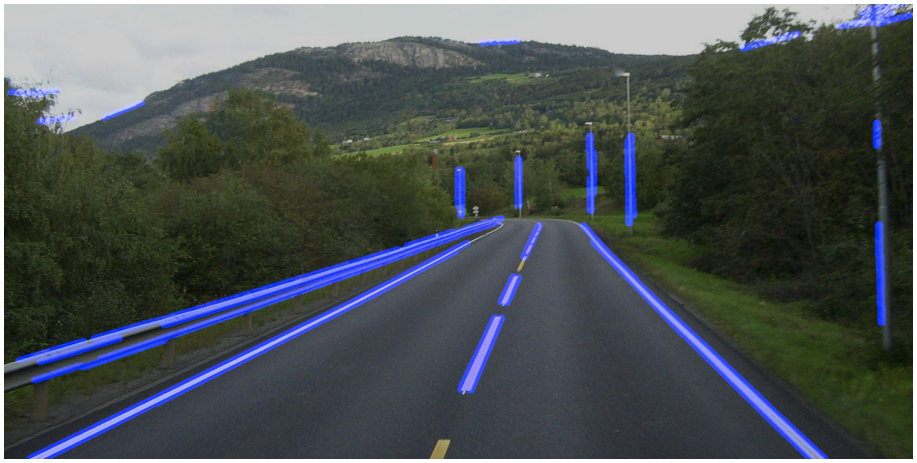

Figure 45: The detected effect without extracting the ROI and fitting

The lines of the following image have been fitted, but also has not extracted the ROI:

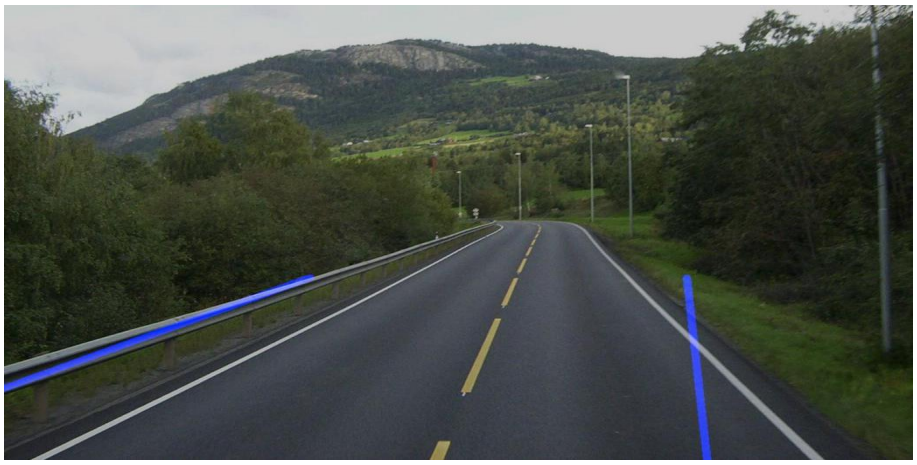

Figure 46: The detected effect without extracting the ROI

So as we can see, whether the lines are fitted or not, if we haven't extracted the ROI, the effect of the detection result is really bad, and at the same, to big size image also makes the computer calculate more.

This paper used matplotlib to derive the coordinates of the region of interest in an image.

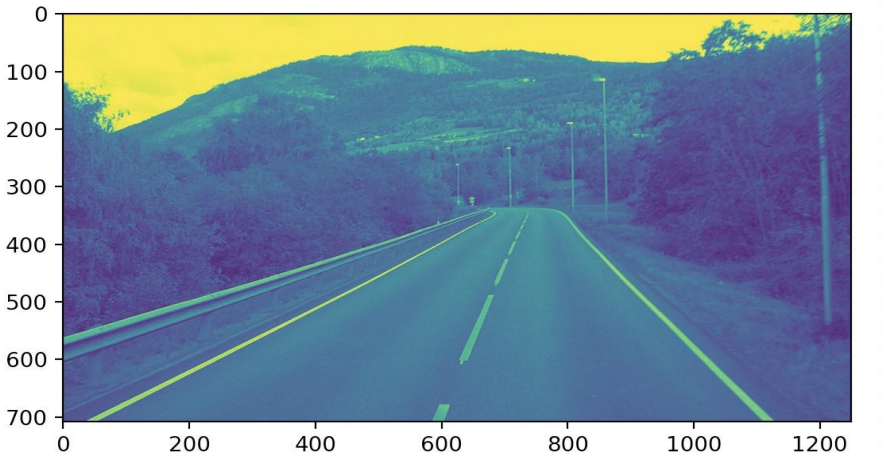

Figure 47: The *matplotlib* figure

Defining appropriate coordinates of ROI is also important. In this situation, this paper chose the coordinates of (0, height), (735, 330), (1200, height) to define a triangular region, which is more suitable for this photo, then we draw the triangle with the three

points on the whole black image with is have the same size as our original image to make the mask, then use the mask to extract the ROI. The following step is processing this ROI.

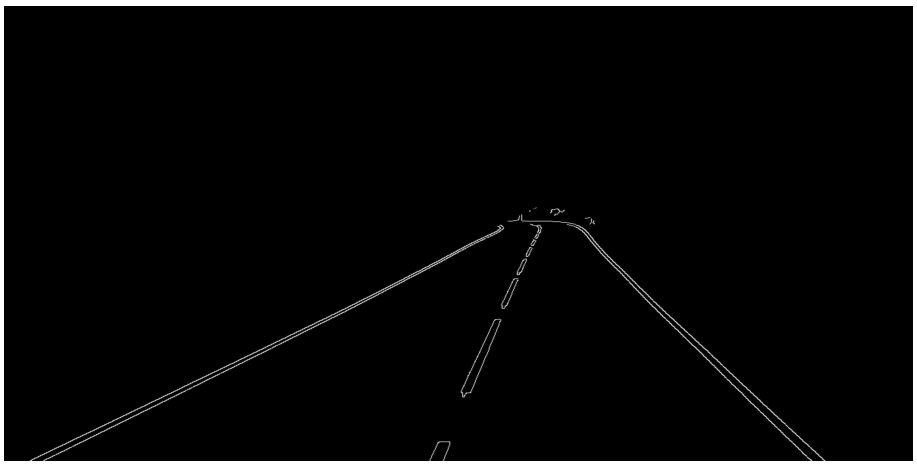

Figure 48: The ROI (1)

If the coordinates can not be defined suitably, the accuracy of detection is not good, just like the following image, its ROI coordinates set as (0, height), (735, 200), (1200, height), which is a little bit larger then the suitable ROI size (This time the lines have not been fitted):

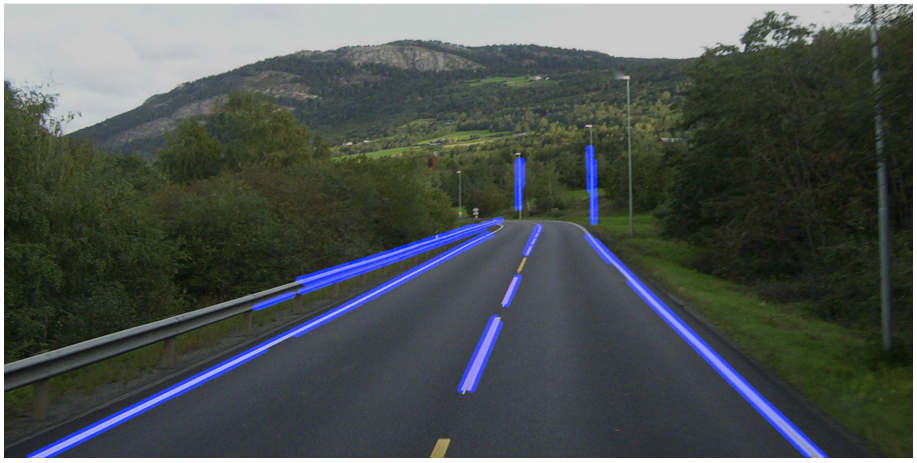

Figure 49: The image used ROI(1)

This time the lines have been fitted:

Figure 50: The image used not suitable ROI and has been fitted

And if we choose the suitable ROI coordinates, we can mark the lines:

Figure 51: The image used suitable ROI and has been fitted

However, when a automatic driving car is driving on a real road, even though the camera's field size of view remains the same, the lane lines will not always be in a fixed position in the photos it takes, the vehicle will turn or change its driving position, and this is when we cannot use just this precise coordinates to determine the area of interest. Actually, the ROI is usually defined as the lower half of the image, for the reason the lane line information is mainly located in the lower half of the image, or it

also always set as below the vanishing point of the lane line, but they are not always accurate either.

The following ROI is of the coordinates of (0, height), (0, height/2), (width, height/2), (width, height), which is the lower half of the images:

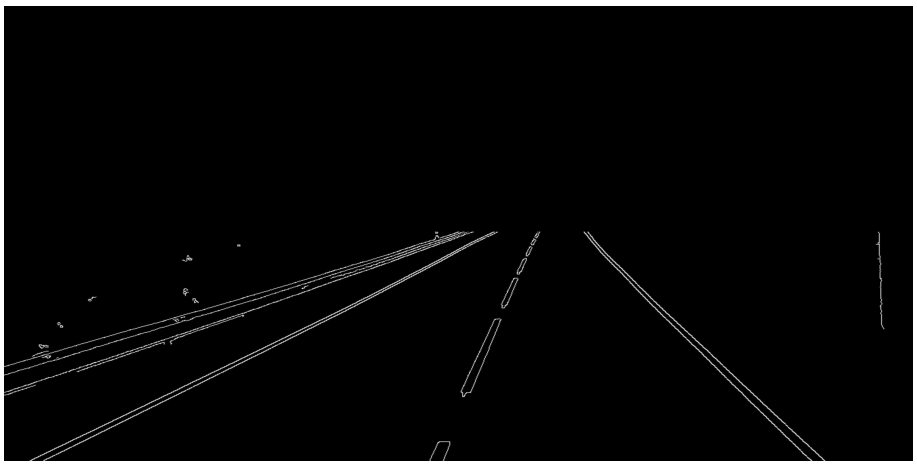

Figure 52: The ROI (2)

The following image is the result of detection:

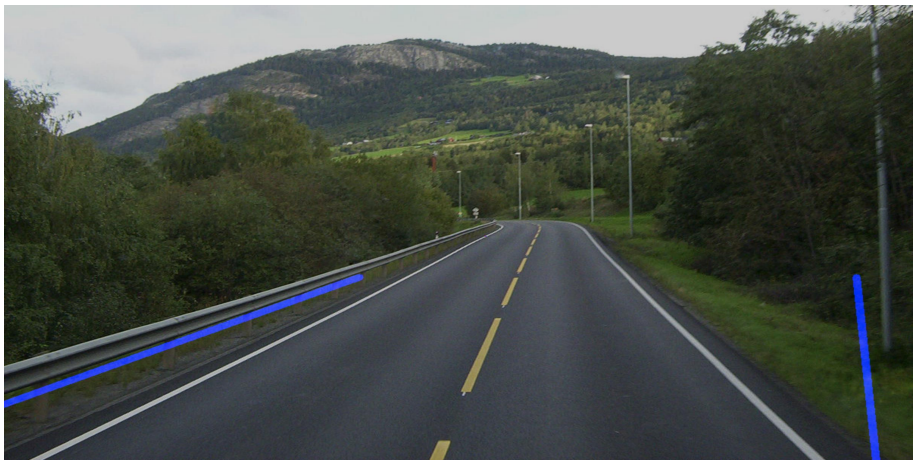

Figure 53: The image used ROI (2)

As we can see, the result is not ideal. So in the traditional lanes line detection method, defining the ROI is critical and a little difficult.

3.2 Lane lane information recognition

In image processing, it is sometimes necessary to search for specific shapes in an image, such as circles, triangles, squares, straight lines, etc. The feasibility of searching directly using the image dot matrix is too low and very complex and time-consuming, so a simpler method is to map the image pixels into the parameter space according to a certain algorithm.

In more detail, this means that a curve or line of a certain shape in the raw image is converted to a point in the transformation space, forming a peak point. This transformation transforms the problem of detecting a curve or line of a certain shape in an image into one of finding a peak in the transformation space, i.e. the problem of detecting the overall feature into one of detecting a local feature. The problem is considerably less complex and less efficient, and good results are obtained.

Whether based on color or pixel gradients, lane lines are essentially recognized on the basis of their characteristics. The most important feature of lane lines is that they are vertical and thin parts of the image. This paper used the *Hough* transform for lane line detection.

As a typical feature detection method, the *Hough* transform has been widely applied in image analysis, computer vision, and digital image-processing[53]. the *Hough* transform is applied to identify and find features of objects, such as lines, circles, etc. The *Hough* transform provides a way of mapping information about each pixel in an image into the parameter space and then making judgments about specific shapes in the parameter space. It takes advantage of the duality between the image space and the *Hough* parameter space point-a-line to convert the detection problem in the image space to a problem in the parameter space. A simple accumulation statistic in the parameter space, followed by a method of finding the peak of the accumulator in the *Hough* parameter space, allows for good detection of straight lines[54].

The principle of *Hough* Transform:

In a right-angle coordinate reference system, a line can be expressed by this equation:

$$y = ax + b \tag{1}$$

In this case, a represents the slope and b represents the intercept, the above equation can be translated into the following equation:

$$b = -xa + y \tag{2}$$

We use equation(2) to represent a line in another space, consisting of the coordinate points [a,b], with x denoting the slope and y the intercept. The point [x,y] represented in equation (1) corresponds to a curve in equation (2). Since the lines represented in equation (1) are composed of an infinite number of points, they correspond to an infinite number of curves in equation (2), which must have a common point of intersection. Once this point is found, the geometric parameters of the line can be determined[54].

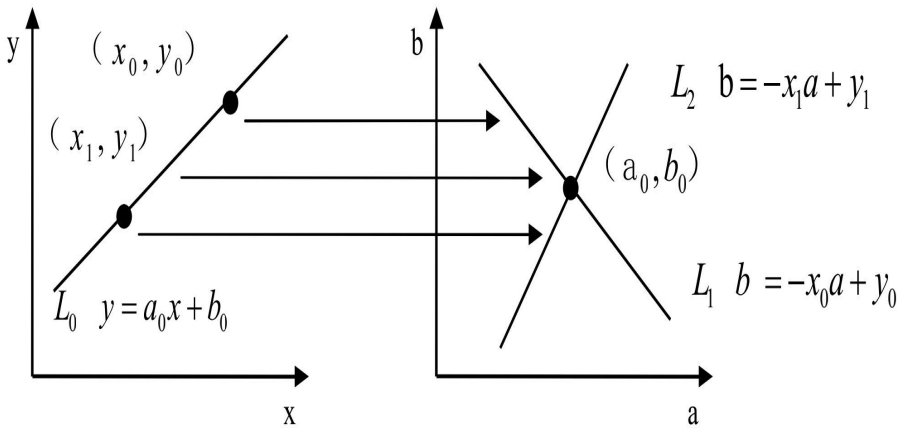

Figure 54: The *Hough* transform coordinate system (1)[56]

Using this idea, it is possible to transform the lines of the coordinate system in which the image is located to the points of the parametric coordinate system and thus solve for the geometric quantities.

However, the lane line to be detected may appear vertical on the screen. When using a linear equation based on a primary function, the slope becomes infinite. In order to better represent the lane lines, this paper used an implementation based on parametric equations, which means that describing the original image in polar coordinates:

$$\rho = x\cos\theta + \mathrm{ysin}\,\theta$$

In the equation, ρ represents the length of the vertical line from the origin to a line, and θ represents the angle formed by the vertical line in the positive direction of the x-axis. In this way, a point in the coordinate system where the original image is located corresponds to the sine of the polar coordinate system. Since lines are made up of an infinite number of points, there must be several sine lines at the same time when changing to another coordinate system, and these lines must have a common point of intersection. The point represents the geometric parameter of the line, and the *Hough* transformation applies this property of the point line [55],[56].

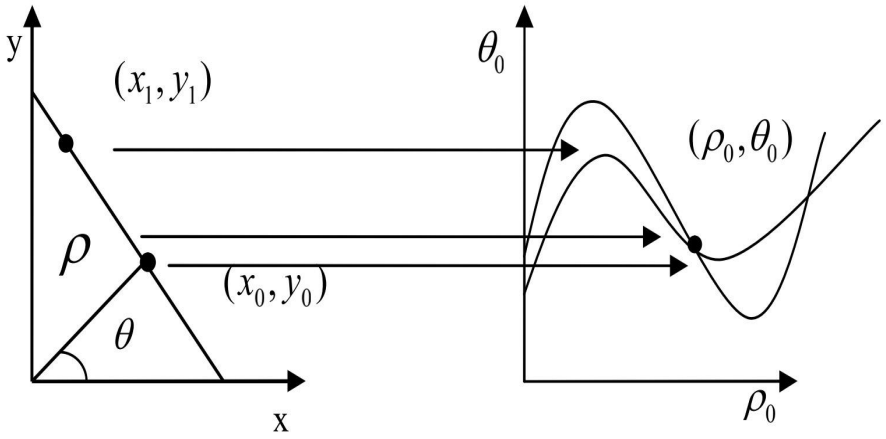

Figure 55: The *Hough* transform coordinate system (2)[56]

When we examine all points in the image space, many corresponding curves will appear in the corresponding parameter space. When these curves in the parameter space have intersecting points, which means that the lines over these points at a certain ρ and a certain value of θ are coincident, and it is not difficult to deduce that these points exist in a straight line in the image space, in conjunction with the conclusion drawn from the previous problem that one point in the Cartesian coordinate system corresponds to one curve in the parametric equation (any line over this point) of the image space.

The ρ and θ values can be discretized into a finite number of equally spaced discrete values. The discretized parameter space is no longer continuous and will be discretized into a grid of equal size.

The main implementation process is that after transforming the coordinate values of each pixel in image space into parameter space, the resulting value will fall within a grid, causing the accumulation counter of that grid to be added by one. Once all the pixel points have been *Hough* transformed, the cell grid can be checked. Prior to this a limit should also be placed on the number of points on a straight line, i.e. a threshold should be set so that when the count value of a cell grid is greater than the set straight line threshold, it is considered a straight line.

The essence of the *Hough* transform is to cluster pixels in the image space that have some correlation and then find a way to link them together in an analytic form with cumulative correspondence in the parameter space. This method works well for parameter spaces below two dimensions, but the transform is less effective when the parameter space exceeds two dimensions[57].

This paper use *cv2.HoughLinesP(image, rho, theta, threshold, lines=None, minLineLength, maxLineGap)* function to complete the *Hough* transform, to be specific:

a) The *image* parameter indicates the output image for edge detection, which is a single-channel 8-bit binary image.

b) The *rho* parameter indicates the parameter polar diameter r Resolution in pixel values, this paper used 2 pixels.

c) The *theta* parameter indicates the parameter pole angle theta resolution in radians, this paper sets it to *"np.pi/180"*.

d) The *threshold* parameter indicates the minimum number of curve intersections required to detect a straight line, this paper sets it to 100.

e) The lines parameter represents the container that stores the parameter pairs *(x_{start}, y_{start}, x_{end}, y_{end})* of the detected line, i.e. the coordinates of the two endpoints of the line segment.

f) The *minLineLength* parameter indicates the minimum number of points that can form a line; lines with less than that will be discarded; this paper sets the parameter to 40.

g) The *maxLineGap* parameter indicates the maximum distance of bright points that can be considered to be on a line and is set to 5 for this paper

The following is the detected result after using the above steps:

Figure 56: The detected image after *Hough* transform

Although the lane lines are detected on this image using the *Hough* transform, the *Hough* transform method itself still has several drawbacks:

a) When counting statistically, not all count points correspond to the same line and may not be co-linear, which can make the detection less accurate.

b) The process of *Hough* transformation requires the conversion of the coordinate system for each point in the image and is computationally intensive.

c) When the value of the polar coordinate space counter reaches the threshold requirement, the system will simply assume that a straight line has been formed in the original image space, when the line may not exist in the original image space, possibly due to a discontinuity in the curve, and the straight line judgment is inaccurate.

d) When applying the *Hough* transform, quantization in polar coordinate space into several grids is required. If the quantization is too small, the higher the accuracy, but at the same time it may cause the system to repeat the calculation of a straight line; if the quantization is too large, the processing is more tedious, and the system will consider that the points counted have constituted a straight line after meeting the

threshold requirement, while the edge points may be included at this time, and the effect is not satisfactory. A deeper quantization makes the calculation more complex and less effective in real-time, and conversely, less effective. It is therefore important to weigh up the degree of quantization to ensure accuracy and effectiveness[56].

3.3 Least squares based lane line fitting

Although the *Hough* transform algorithm ensures the accuracy of lane line detection to a certain extent, it still has some limitations of its own, it will be limited by the image noise, due to the interference of image noise, the accuracy of the *Hough* transform detection becomes limited, will detect some forged edges and existing science and technology can not completely avoid the interference of image noise. Therefore, to further improve the accuracy of lane line detection, we use the least-squares method to fit the lane lines for further detection[58].

3.3.1 The principle of least squares

The basic idea of least squares is that given a set of experimental data, which are often ordered pairs, the best functional match for these data is found according to the principle of minimizing the sum of squares of errors.

The mathematical principle of least squares is: given a set of data (x_i, y_i) (i = 1, 2, ..., n), let its experimental equation be F(x), which contains some coefficients to be determined a_n. Substitute (x_i, y_i) into the equation to find the difference y_i - F(x_i), in order to consider the overall error

The difference can be taken as a sum of squares. The reason for squaring is to take into account that the positive and negative errors can cancel each other out by adding them directly, so the error is noted as:

$$e = \sum (y_i - F(x_i))^{\wedge} 2$$

By finding the minimal value of e it is possible to find an, and thus the best-fit function for the set of data, which minimizes the sum of squares of the errors.

To be specific, Assuming that there are P known data points, a straight line fit is to determine a straight line for these points. Since the detected data is not very accurate and there will be errors or incorrect data, this straight line does not need to go through all the data points, as long as it is very close to these points, and this straight line is the best-fitted line. Let x and y satisfy the following linear relationship.

$$\mathrm{y} = G(\mathrm{x}) = ax + b$$

The expression for the correlation coefficient r for the least-squares parameter is calculated as follows:

$$r = \frac{\sum (x_i - \bar{x}) \sum (y_i - \bar{y})}{\left[\left(\sum (x_i - \bar{x})^2\right)\left(\sum (y_i - \bar{y})^2\right)\right]^{\frac{1}{2}}}$$

$$\bar{x} = \frac{\sum x_i}{P}$$

$$\bar{y} = \frac{\sum y_i}{P}$$

In the equation, r represents the degree of linearity of fit, representing the functional relationship established between the two variables, and takes values in the range [-1, 1]. r takes a positive maximum value and a negative minimum value to mean that x and y are positively or negatively correlated, respectively. the closer the absolute value of r is to 1, the better the linear relationship established between the two variables, and close to zero means that the linearity is not good, if it is too poor, the linear fit has no practical meaning.

The least-squares method is used to fit the lane lines to reduce errors and achieve higher accuracy and more precise lane lines, and the least-squares method is simple to calculate and very small, requiring only one traversal of the data points and a relatively fast response time.

This paper used *np.polyfit()* in OpenCV. In this part, this paper divided the fitted lane lines into left and right lane lines based on whether the slope of the line is greater than zero. If the slope is less than zero, it is classified as a left lane line, and if the slope is greater than zero, it is classified as a right lane line. The origin of the coordinate system in OpenCV is in the top left corner of the image. Because the coordinate in OpenCV is different from the normal one, the X-axis of the coordinate system in OpenCV is the horizontal line above the image rectangle, from left to right, and the Y-axis is the vertical line to the left of the image rectangle, from top to bottom, so the slope of the left-hand lane line is less than zero, and vice versa.

After fitting the lane lines, drawing the lines on the black mask, finally using the function of *cv2.addWeighted()* to combine the lane lines and the original image. This paper set the weight of the original image as 0.8 and set the weight of the lane lines as 1 to clearly show the lane lines.

The following image is the final result:

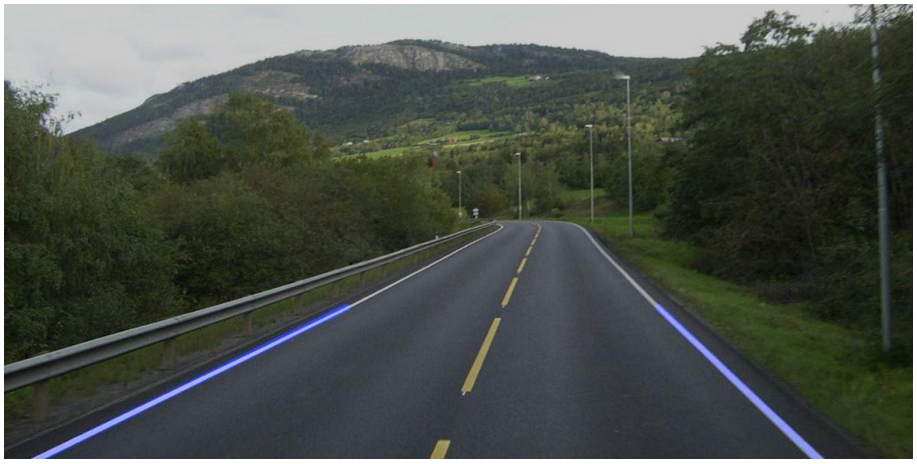

Figure 57: The final detected image

Chapter 4. Implementation, Result and Result Discussion

4.1 Implementation

Although in the last chapter the lane lines have been detected successfully, just judging the left and right lane lines through if the slope is less than 0 is not accurate.

The following image is detected by *Hough* transform, which has not been fitted:

Figure 58: The image without being fitted

From the above image, we can see that the detected lines without being fitted, the detected result is really bad, which is affected seriously by noise, even if some of the line segments are correctly marked on the lane lines, but they are discontinuous and rough, so it's necessary to fit the lines.

The following image is the original image and the image after being detected by Canny operator (it has already been blurred by Gaussian filter):

Figure 59: The original image (2)

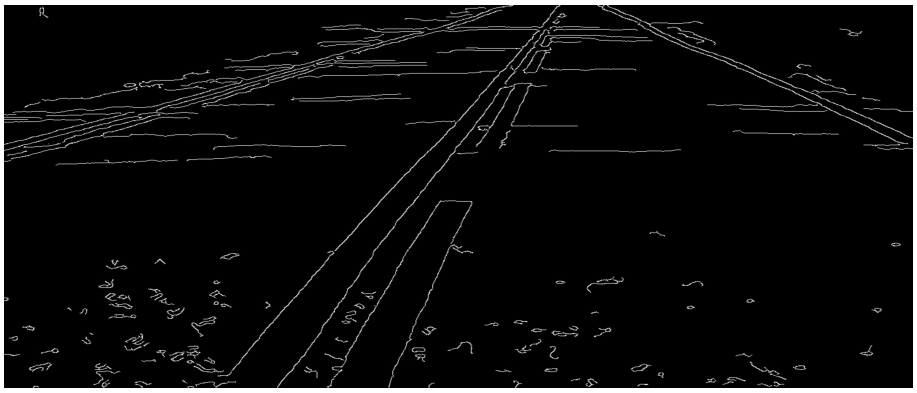

Figure 60: The image after Canny operator edge detection

Through the above image, we can see that this image is seriously affected by the shadow, which causes the inaccurate detected lines after *Hough* transform.

The following image is fitted by the above method, which only is classified by whether the slope is less than zero:

Figure 61: The image after being fitted (1)

Through the above image, we can see that the detected result is still not accurate, because there are many vertical fake lines detected in this image. So it is necessary to eliminate some incorrect points which must not be on the lane lines. We can limit the slope to achieve this. To be specific, since the points on the different lines detected according to the *Hough* transform are fitted separately in this paper, and then calculate the average of the slope and the average of the intercept, and the final lane line detection results are drawn according to these two averages, this time, we left only the points which meet the required slope requirement to be fitted and then performed the average calculation, which means that we can choose a range for the slope, if the points slopes are not in this range, we get rid of them.

This time the range of the slopes of the points left behind is set as the absolute values of them are greater than 0.05, and then calculate the average of the slope and the intercept of these points.

The following is the result (In order to let the marked lines more clear, the color of lines are set as red):

Figure 62: The image after being fitted (2)

We can see that by modifying the slope range, the lane lines are detected more accurately than before, but the lane lines detected on the left still do not quite match the actual lane lines.

This time we set the ranges as greater than 0.1, 0.2, 0.3, 0.4 respectively, the following images are the result after being changed:

Figure 63: The image after being fitted (3)

Figure 64: The image after being fitted (4)

Figure 65: The image after being fitted (5)

Figure 66: The image after being fitted (6)

As for the above images, we can see that it is appropriate to set the lower limit of the slope between 0.3 and 0.4.

The following images are another lane line photo that the range of the lower limit of the slope is set as 0, 0.05, 0.1, 0.2, 0.3, 0.4 respectively:

Figure 67: The image after being fitted (7)

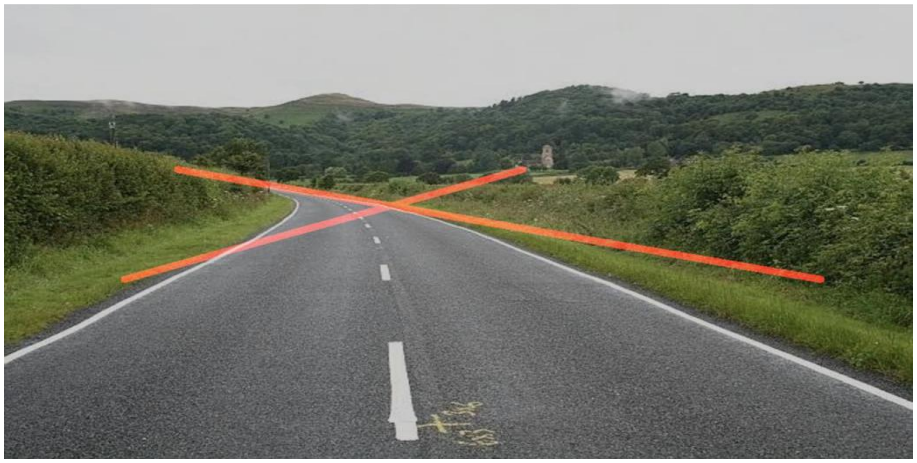

Figure 68: The image after being fitted (8)

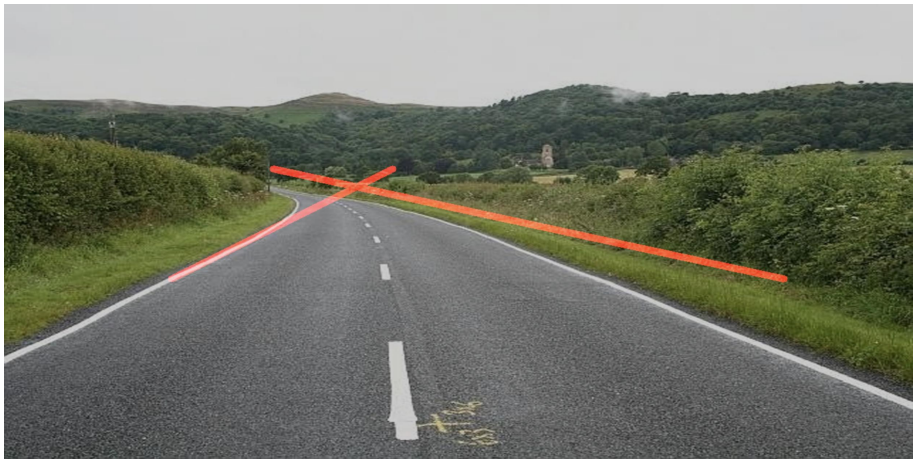

Figure 69: The image after being fitted (9)

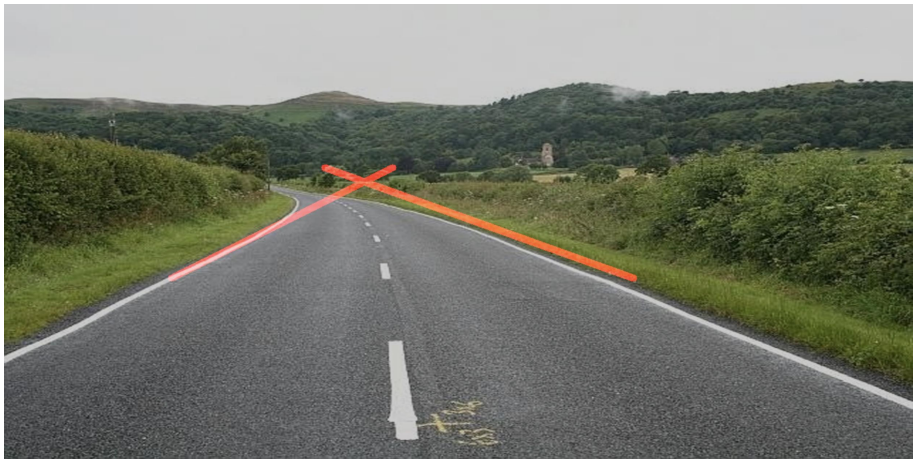

Figure 70: The image after being fitted (10)

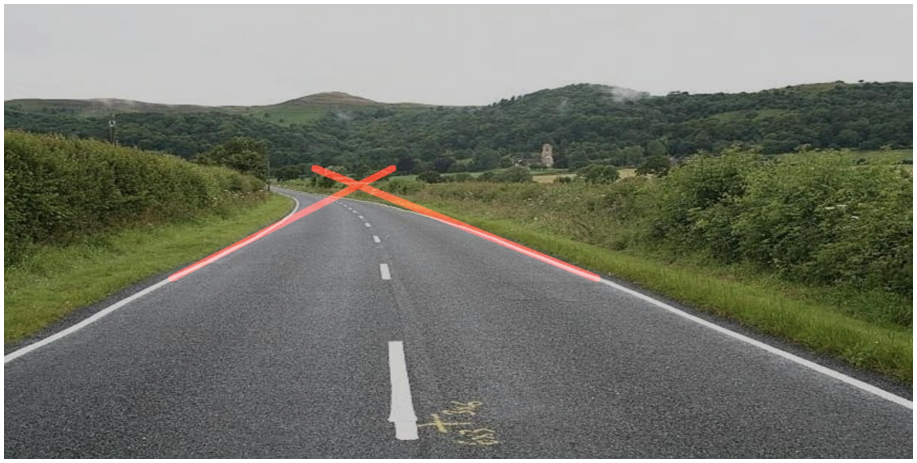

Figure 71: The image after being fitted (11)

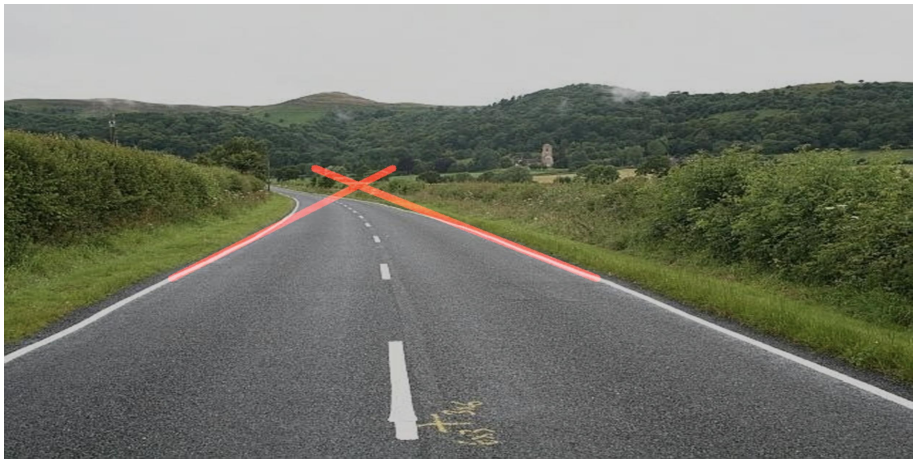

Figure 72: The image after being fitted (12)

Through the above images, we can see that the lower limit of the slope set as 0.3-0.4 is also suitable for this image. In this experiment, this paper chose 0.3 as the minimum slop.

The program in this paper is written in Python based on the open-source computer vision library OpenCV. The computer platform used for this experiment is Intel Core i7, 2.6 GHz 6-Core, with 32 GB of memory.

This paper chose the cordova1, washington1, washington2 scenes in Caltech dataset, which must be mentioned is that the coordinates of ROI of these three scenes are set manually according to the vanishing points of the roads, respectively:

1) In the cordova1 scene, the coordinates of ROI are (0,height), (width/2,3*height/10), (width,height);

2) In the washington1 scene, the coordinates of ROI are set as (width/4,height), (width/2,2*height/7), (width,height);

3) In the washington2 scene, the coordinates of ROI are set as (0,height), (width/2,5*height/14), (width,height).

All of the ROIs are set as triangles.

On the whole, this experiment for detecting lane lines is just based on the traditional *Hough* transform method, which used Gaussian filter and Canny operator to pre-process the images, and after extracting the ROI, the function *cv2.HoughLinesP(src, 2, np.pi/180, 100, np.array([]), minLineLength=40,maxLineGap=5)* was used to complete the *Hough* transform, finally the lines are fitted by the *np.polyfit()* function, and the minimum slop was set as 0.3.

The aim of this experiment is to analysis and explain the shortcomings of traditional *Hough* transform method, so this paper took the detection result of this experiment and compared it with the detected results in the reference[21] and reference[22].

To be specific, reference[21] is an improved *Hough* transform lane line detection algorithm based on edge information coupling. After obtaining edge information through edge detection, it counted the number of edge pixels in each line in the search region and used the line with the most edge pixels as the dividing line for the ROI to determine the ROI. Then, to suppress the effect of cluttered backgrounds such as non-lane line edges, a specific gradient direction was chosen to refine the RoI. Finally, the direction interval and threshold were used to *Hough* transform to improve it and applied to the edge pixels to extract the lane lines. Reference[22] proposed a lane line detection method based on deep learning, which learned lane line features using a CNN model, passed the lane line features obtained from the CNN to a BLSTM RNN (Bidirectional Long Short Term Memory RNN) model, extracted lane line feature points according to the bi-directional memory feature of BLSTM RNN, the lane line feature points were extracted using the multi-frame lane line information and the lane line type results were output. Finally, the dynamic programming algorithm was used to plan the optimal path for the lane line feature points and the least squares method was used to fit the lane lines.

All these three experiments were completed on the Caltech dataset.

4.2 Result

The following images are part of the detected result of cordova1:

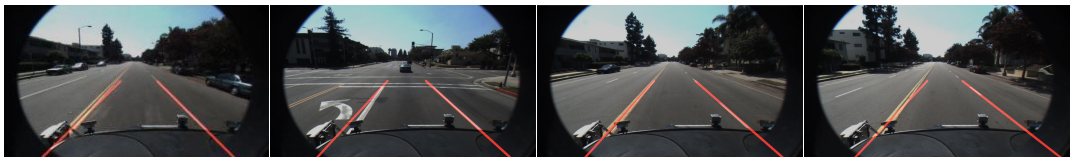

Figure 73: The detected results of cordova1 scene

In this scene, a total of 250 frames were detected in this experiment, in which the number of frames detected incorrectly was 39.

The table below shows a comparison of the results[22][21]:

Method	Accuracy(%)
This paper	84.40
Reference[22]	92.00
Reference[21]	98.02

Table 2: Accuracy of testing cordova1

The following images are part of the detected result of washington1:

Figure 74: The detected results of washington1 scene

In this scene, a total of 337 frames were detected in this experiment, in which the number of frames detected incorrectly was 53.

The table below shows a comparison of the results[21][22]:

Method	Accuracy(%)
This paper	84.27
Reference[22]	89.55
Reference[21]	97.05

Table 3: Accuracy of testing washington1

The following images are part of the detected result of washington2:

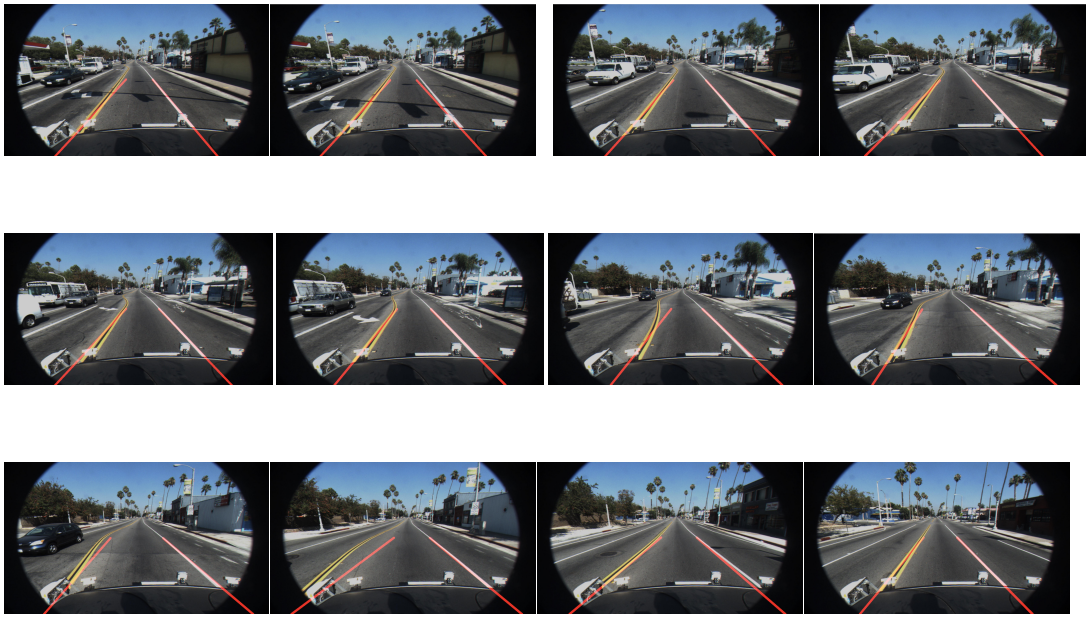

Figure 75: The detected results of washington2 scene

In this scene, a total of 220 frames were detected in this experiment, in which the number of frames detected incorrectly was 34.

The table below shows a comparison of the results[21][22]:

Method	Accuracy （%）
This paper	84.55
Reference[22]	89.13
Reference[21]	96.58

Table 3: Accuracy of testing washington2

4.3 Result Discussion

As we can see from the above data, the accuracy of the traditional *Hough* transform method is not ideal, it is far lower than the improved *Hough* transform method and the deep learning method. The photos in the dataset are real photos with distracting information such as lighting and shadows, and in some frames the lane lines are not clear, all of which have a significant impact on traditional *Hough* transform method and result in lower accuracy than the other two methods.

In addition, the coordinates ROI of three scenes in this experiment of traditional *Hough* transform method are all need to be set manually, which means that this algorithm will be affected by the situation. And if the ROI area is defined as the bottom half part of all the images, the result is also not ideal, just as the last chapter described (in the part of *"Extract the ROI"*), because there will be more interference information included. And there are many parameters in traditional method that need to be set manually, such as the parameters in the *Hough* transform function, not just the ROI. If these parameters are not set properly, they will have a negative impact on the results.

Finally, the traditional *Hough* transform method can not detect the curve lane lines, this method can only detect the tangential part of the curve, just as the following image shows:

Figure 76: The effect of detecting curve lane line (1)

The next chapter will detail the improvements that other scholars have made to the traditional method.

Chapter 5. Improvements and Discussion

Scholars P. Maya and C. Tharini [59] proposed a method for partial *Hough* transform for lane line detection.

This is the *Hough* transform equation:

$$\rho = x \cdot \cos\theta + \mathrm{y} \cdot \sin\theta \qquad \theta\varepsilon(\pi - \pi)$$

θ is the Angle formed by the vertical line and the positive direction of the X-axis. The range of θ in the *Hough* transform is 0 to 360 degrees, whereas actual lane line detection does not require that large range, which means that it produces more data than will be used in real lane detection. So P. Maya and C. Tharini limited the range of θ. They analyzed the results obtained by using different range θ in the *Hough* transform in their papers, and the ranges of θ are set as (0 to 30), (30 to 60), (60 to 90), (-90 to -60), (-60 to -30) and (-30 to 0).

Here is the data they obtained[59]:

Method	Accuracy
Traditional *Hough* Transform method	92.3%
Proposed Partial *Hough* Transform	92.3%

Table 4: Accuracy in reference[59]

Method	Processing Time(ms)
Traditional *Hough* Transform method	0.82
Proposed Partial *Hough* Transform	0.431

Table 5: Processing time in reference[59]

By analyzing the detection results obtained from the angles in different ranges, they concluded that by varying the θ in the *Hough* transform during lane line detection, not only did they not lose detection accuracy, but they were also about 50% faster than traditional detection methods and, based on their analysis, they found that the left lane on the road lies in the theta range of 60 to 90, the center lies in the range of 0 to 30 and the right lane lies in the range of - 60 to -30 range. And the main implication of this research is that by limiting the θ in the *Hough* transform, the computer's computational effort is significantly reduced, thus reducing processing time.

Another shortcoming of *Hough* transform detection is that it only can detect the straight line, it can not detect the curve line on the road:

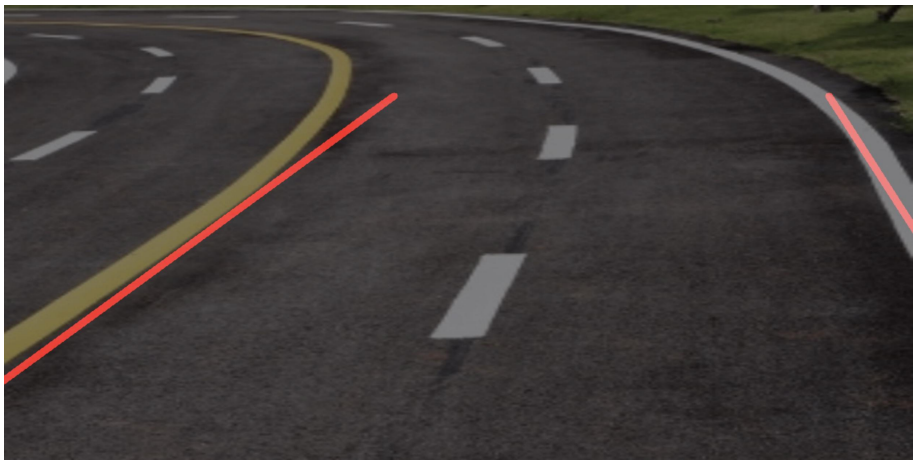

Figure 77: The effect of detecting curve lane line (2)

As we can see through the above image, the marked lines on the image are straight, but the lane lines are curve (in order to show the marked lines more clear, this time the weight of image brightness is set less to make it a little bit darker).

But we can change the parameters in *Hough* Transform, to be specific, this paper used the OpenCV function *cv2.HoughLinesP(image, rho, theta, threshold, lines=None, minLineLength, maxLineGap)* to complete the *Hough* transform, this time we can the length of the lines that *Hough* transform detected to be shorter, because in the previous detection, the straight lines are so long that they can not fit into the curve

lane lines. This time we can change the long straight lines to short lines, although they are still straight, the shorts lines can compose the curves approximatively[25].

As for the parameters in the *cv2.HoughLinesP(image, rho, theta, threshold, lines=None, minLineLength, maxLineGap)*, the threshold parameter indicates the minimum number of curve intersections required to detect a straight line, the previous set is 100, the *minLineLength* parameter indicates the minimum number of points that can form a line, lines with less than that will be discarded, the previous set is 40, the *maxLineGap* parameter indicates the maximum distance of bright points that can be considered to be on a line, the previous set is 5[65].

To detect short lines, this time the threshold is set as 50, and the *minLineLength* is set as 20, here is the detection result:

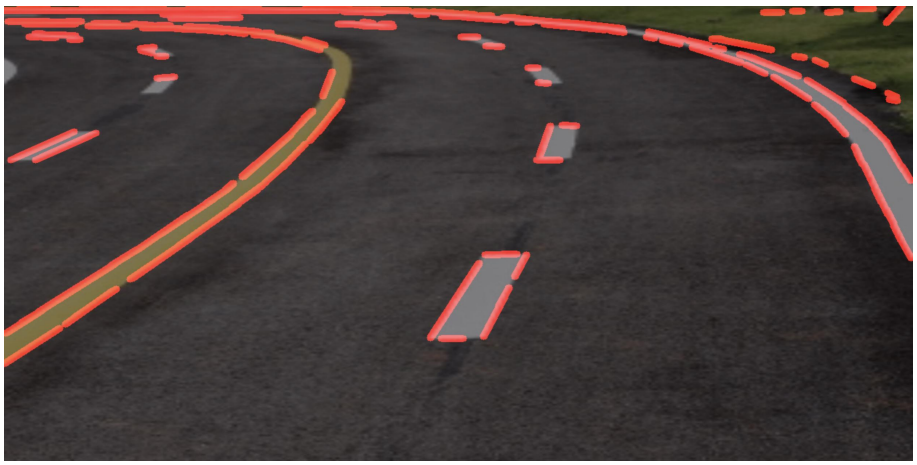

Figure 78: The effect of adjusting the parameters in *Hough* transform (1)

As we can see through the above image, the detection result is a little bit better than before, but not so good enough, because the shorts lines can not connect well.

This time the threshold is set as 10, and the *minLineLength* is set as 10, here is the detection result:

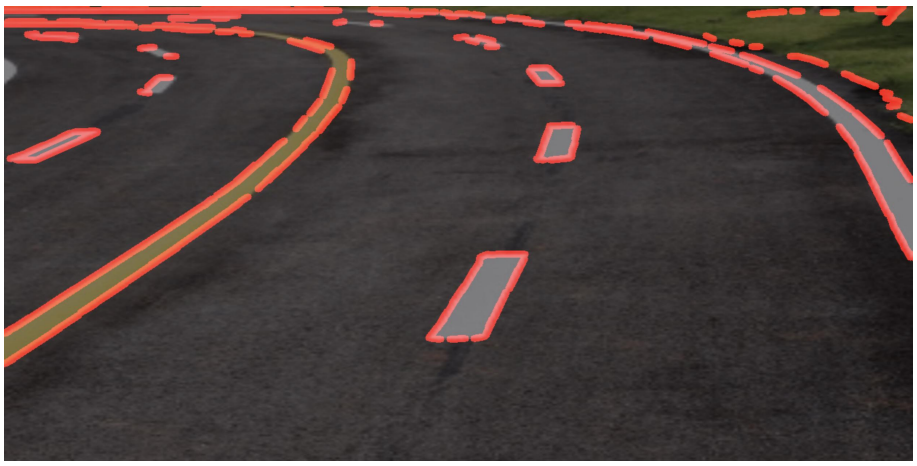

Figure 79: The effect of adjusting the parameters in *Hough* transform (2)

As we can see through the above image, the detection effect is better than before, the gaps between the marked lines on the lane lines have been shrunk.

This time the threshold is set as 10, and the *minLineLength* is set as 5, here is the detection result:

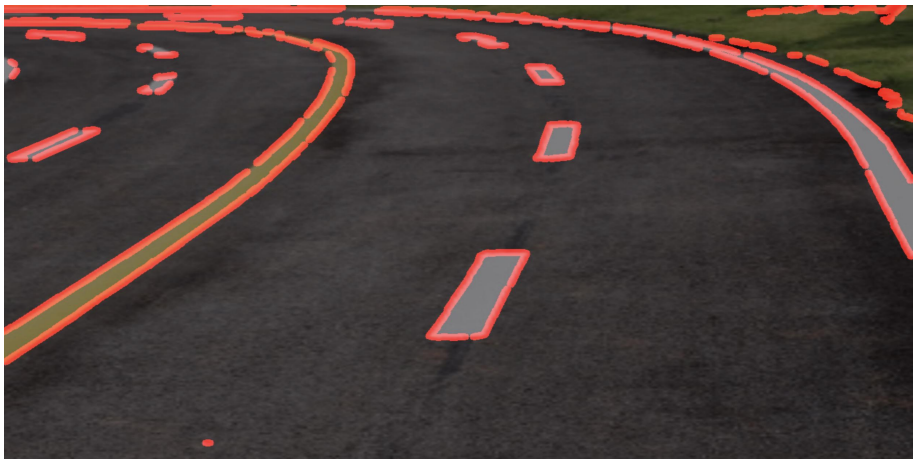

Figure 80: The effect of adjusting the parameters in *Hough* transform (3)

But this time through the above image, the *minLineLength* is so short that there is a noise point has been detected in the left bottom of the image.

For the above images, we can see that setting the *minLineLength* as 10 is suitable, but there short dashed lane lines between the solid lanes lines are also be marked, which

is not what we want. *Hough* transform can detect the short dashed lines, it is for the reason that the *minLineLength* has been reduced. But we can let these dashed lane lines which have been drawn not display.

To be specific, this paper used *for loop* to iterate through the points of the lines which are detected by *Hough* transform, and in the *loop,* using *cv2.line()* to draw the lines through the points. Because in the general photos of the road lane lines which are taken by the cameras, there are solid lane lines on either side and dashed lane lines in the middle. So when the points are iterated, the points which are near the middle of the image should be removed and the points which are in the range of two sides of the road can be left behind and drawn the lines. The range chosen is important.

This time, the range of the points which are left behind are in the range of "the X coordinates of them should less than 1/3 of the image width or greater than 2/3 of the width", here is the detection result:

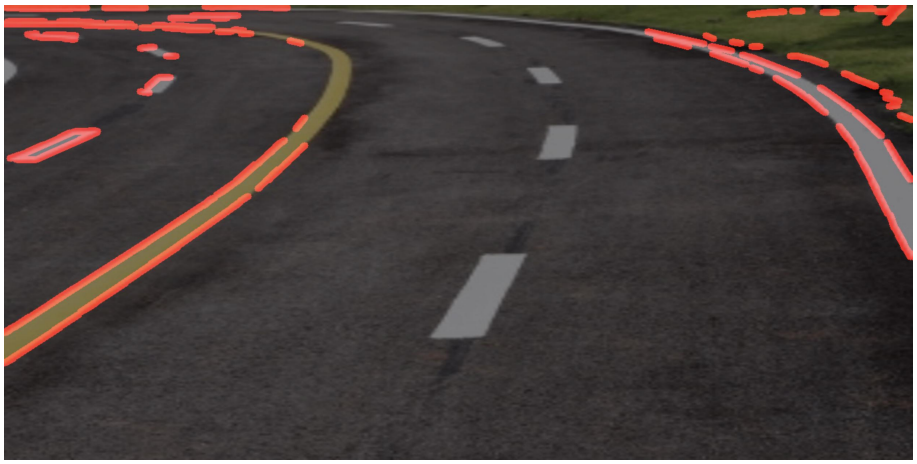

Figure 81: The effect after adjusting the range (1)

As we can see through the above image, although in the middle of the image, the dashed lane line has not been marked, the solid lane line has not been marked complete, especially the yellow lane line. Although the dashed lane lines in the left part of the image have been marked, we can ignore it, because a car only travels in one lane, it will not travel to the left part of the image. But the yellow solid lane line

should definitely be marked, otherwise, it's too dangerous for a car to make a turn. So the range of the points which are left behind should be enlarged, in this time it is set as "the X coordinates of them should less than 2/5 of the image width or greater than 3/5 of the width", the following image is the detection result:

Figure 82: The effect after adjusting the parameters of range (2)

Through the above image, we can see that although the yellow solid lane line has been almost marked completely, there are a little bit dashed lane lines have been marked too, so this time, the right range should be shrunk a little, in this time the range is set as "the X coordinates of them should less than 2/5 of the image width or greater than 2/3 of the width", the following image is the detection result:

Figure 83: The effect after adjusting the parameters of range (3)

As we can see through the above image, this time the effect is much better than before.

But which is must be taken seriously is that this time the range “the X coordinates of them should less than 2/5 of the image width or greater than 2/3 of the width” is suitable for this image, but it doesn’t mean it is the most suitable range, because each road is different and the placement of the cameras on the car is different, this paper only tell a method of roughly detecting curved lane lines using the *Hough* transform, but in practice, the feasibility of this method is still not high: on the one hand, the selection of this range depends on the different road surfaces, different camera placements, etc. to judge, on the other hand, the two parameters *threshold* and *minLineLength* are set too low, there will be a lot of irrelevant points and even noise that will be detected.

Two scholars, Linjun Shi and Su Yu[23], proposed a lane line detection method based on the *Hough* transform under multiple constraints. They take information from each color channel of the original RGB image and change the 3 component weights of RGB to grey the image. In addition, they proposed several constraints on the binary map of the image extraction to eliminate a large amount of noise, followed by the *Hough* transform to detect and extract the lane markers. They then improved the *Hough* transform on the basis of the extracted lane segments, and they used a probability-based voting procedure to estimate vanishing points, which were used to constrain the lane segments[35]. Finally, they clustered the remaining useful lane lines.

To be specific, first of all, in the step of converting a color map to grayscale in image pre-processing, the traditional equation for converting the RGB model to grayscale space is the following equation:

$$I = 0.299*R + 0.587*G + 0.114*B$$

As the road lanes are usually white, yellow, or red in color, they are always painted in a high contrast color to the road surface. To enhance the characteristics of the lane marking information, they used an effective method. The following equation is they have used:

$$I = R + G - B$$

As for blurring the image, they use the weighted median filter, weighted median filter algorithm is basically the same as the standard median filtering algorithm, but the difference is that all the pixels in the image window are given weights, and then the pixels are sorted according to the number of occurrences of the weights. The weighted median filter algorithm protects the image edges and detail information better than the standard median filtering algorithm and removes noise better[61].

For the part of the *Hough* transform detection, because the things that *Hough* transform detects may not be lane lines, they may be guardrails, light, and shadow as well as other disturbing information, the two scholars designed several constraints on the *Hough* transform in order to improve the *Hough* transform detection accuracy. Firstly, depending on the position of the universal cameras in the car, the absolute value of their slope $|\theta|$ will be constrained to a range ($\theta_{min} < |\theta| < \theta_{max}$), just like the following figure:

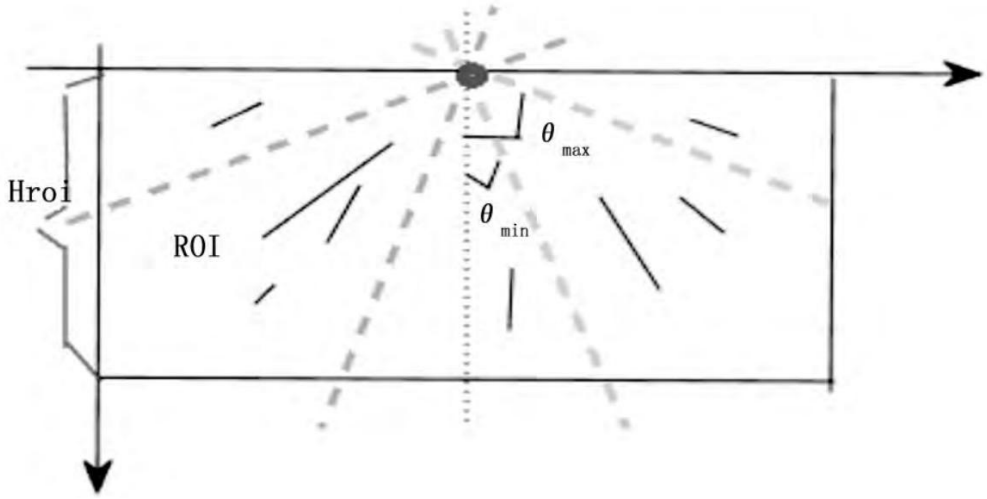

Figure 84: The ranges of angles of lane lines [23]

$$\theta_{\min} = 20^{\circ}, \theta_{\max} = 90^{\circ} - \arctan\left(\frac{H_{roi}}{2W_{roi}}\right)$$

H_{roi} denotes the height of the ROI and W_{roi} denotes the width of the ROI. If the candidate line is out of range, it can not be considered as a lane line.

Secondly, the detected straight lines not only include the lane lines, but also include the interference lines. It is verified through sufficient experiments that the angular difference between every two detected lane markings cannot be less than 10°. When a candidate line is detected, then the angular difference between that line and the individual detected lines should be calculated. If any difference is less than 10°, then the candidate line is treated as interference information and it will be removed. By using the above method, the interference lines can be eliminated.

Finally, as the lanes have a certain width, there will be a certain distance between the lane lines. The first step is to determine the center line of the ROI horizontally, this center line divides the ROI into two parts and the height of the top and bottom parts are equal, which means that the height of each part is $H_{roi}/2$. Then finding the intersection points between the center line and all detected lane markings. The distance between the intersections should be greater than a threshold T. With sufficient experimental validation, the two scholars set T to $Wroi/6$. When a candidate

line is obtained, the intersection between the center line and the candidate line is calculated, and then all distances between the intersection and the lane markings examined are calculated. If one of the distances is less than the threshold, the candidate line is removed directly. Smears and pseudo-edges in the pavement image can be effectively eliminated with this method[23].

In real life the lane lines are parallel, but in a photograph taken by a camera the lane lines intersect at a point where the lane lines intersect at a vanishing point. The two authors proposed a robust vanishing point estimation method using a probabilistic voting procedure, and then the vanishing points are used to constrain the lane line candidates. This algorithm is based on the thought that "vanishing point detection can be transformed into a one-dimensional histogram search problem". This algorithm has the advantage of accurate vanishing point estimation, but the disadvantage is that it takes a long time to process and requires a high level of real-time performance for lane line identification, so it has not been adopted by other scholars. However, two scholars have previously used the *Hough* transformation with multiple constraint limits, and the number of pairs of line segments left after processing is relatively small, which means that the interference information is reduced, so the processing time is also reduced substantially. After finding the vanishing point, the detected lines are filtered by the vanishing point and a small circle with wi/10 as the diameter is drawn at the center of the vanishing point. W_i is the width of the line segment extracted by the *Hough* transform algorithm. The lines that pass through the small circle on the vanishing point are retained and the rest of the lines are deleted[23].

Through the above steps, although the interference of invalid information on the road surface can be eliminated to a certain extent, there will still be some interference information that cannot be removed. The two scholars use multiple iterations of K-means clustering to automatically select the optimal clustering center and combine feature points in the longitudinal direction through the clustering algorithm, using this method to effectively eliminate isolated feature points and clearly detect the target,

thus reducing the interference of useless information. The K-means clustering algorithm is a typical distance-based clustering algorithm that used the Euclidean distance as the evaluation index of similarity, which means that it is considered that the closer two objects are to each other, the greater their similarity[23].

The same vanishing point constraint approach is used in Liu and other scholars' reference[63]. They use an adaptive ROI localization algorithm to extract the ROI region and use the Laplacian operator to remove most of the interference from the environment. And they used the LSD algorithm when detecting lane lines, as for LSD algorithm, it is a straight line detection segmentation algorithm that produces sub-pixel level accuracy in linear time. The LSD straight line extraction algorithm does not need to calculate the edge map first and can perform straight line extraction directly in a greyscale image. It is designed to work on any digital image without parameter tuning, the algorithm is fast, the line extraction features are accurate and the false detection is controlled, by contrast, the *Hough* transform algorithm needs to first calculate the image edge map as input, which requires more system resources and poor real-time performance, and the extracted straight-line positioning accuracy is not high, and often connects discontinuous straight-lines together to produce false detection, and at the same time cannot accurately locate the end points of the straight line, which is easy to detect false straight lines[64]. The algorithm first calculates the gradient and orientation of the pixels, selects the dominant pixel point as the seed point, then sets the angular tolerance for the growth of the region, obtains the linear support domain, and extracts the straight line by calculating the NFA value (number of false Scherks) of the rectangular region corresponding to the linear support domain. It controls the number of false detection results it makes: on average, one false alarm is allowed per image. The method is based on the approach of Burns, Hanson, and Riseman and used a back-validation method based on Moisan and Morel's theory[62]. However, this algorithm can only detect straight lines, and is useless for curves, and even curved segments with small curvature are difficult to detect.

They then eliminate the non-lane markers by constraining the direction and vanishing point. In this paper, they mention that the left lane line of the ROI has a directional angle between -20 degrees and -70 degrees, and the right lane line of the ROI has a directional angle between 20 degrees and 70 degrees, so they see the left (right) line with a directional angle outside of -20 degrees to -70 degrees (20 degrees to 70 degrees) as a non-lane line. Finally, they extract the exact lane lines from the remaining candidate lane markings.

In Linjun Shi and Su Yu reference and Weirong Liu reference, they both used the method of angle and vanishing point constraining on lane lines. However, Linjun Shi and Su Yu used morphology more to constrain the feature lines, they improved the *Hough* transform by adding constraints and then a probabilistic voting procedure based on estimating vanishing points to constrain candidate lane lines. In contrast, Weirong Liu and other scholars used the LSD algorithm to detect straight lines in the paper.

Linjun Shi and Su Yu compared their method with that of Liu et al. by testing the dataset they tested two working conditions, the sunny day and the overcast sky, with 250 frames for each type of library, on a PC platform. In the following table, the method based on the vanishing point of the conventional *Hough* transform is defined as A, the method used in the reference [63] (by Weirong Liu et al.) is defined as B, and the method used in the reference [23] (by Linjun Shi and Su Yu) is defined as C.

	Sunny Day			Overcast Sky		
	A	B	C	A	B	C
Error Rate	19%	0%	3%	21%	2%	2.5%
Average processing time	37ms	27ms	23.5ms	45ms	32ms	24ms

Table 6: Error rate comparison in reference[23][23]

According to the above table, the algorithm used in reference [23] (by Linjun Shi and Su Yu) has a significantly lower processing time and error rate than the algorithm for finding the vanishing point constraint based on the traditional *Hough* transformation. The detection rate of the algorithm in the reference [63] (by Weirong Liu et al.) is marginally higher than that of the algorithm proposed in the reference [1] (by Linjun Shi and Su Yu), but the processing time of the algorithm in the reference [1] (by Linjun Shi and Su Yu) is much less than that of the reference [63] (by Weirong Liu et al.).

Chapter 6. Conclusion and Future direction

6.1 Conclusion

The traditional *Hough* transform method is simple and easy to implement for detecting lane lines, and the last chapter has described the improvements made by several scholars to the *Hough* transform method: Adding constraints to the *Hough* transform, limiting the lane line slope or angle to the lane line, and using vanishing points to constrain the lane line, etc.

Although the above improvement can improve the algorithm efficiency, save the detection time and make the detection accuracy increase, the method of *Hough* transform for detecting lane lines still has three main shortcomings:

1. It is only applicable to straight lanes, while as for the curve lanes, it can only detect the tangent line part of the curve lines. It cannot be fitted to curved lanes. Even though in the last chapter, when the parameter of the minimum straight-line length of the *Hough* transform is set very small in the paper, these short straight lines can barely form lines that approximate curves, they cannot be fitted to the curve well, this approach is obviously not suitable for the complex road conditions in the real world, as these short segments are just approximated as curves to mark lane lines.

2. Because the *Hough* transform method is based on the features of the lane lines on the road to get the lane line information, which means that it requires a high level of clarity in the image of the lane lines in the road to get the features, if there is some interference information on the lanes lines, for instance, the lane line is broken, incomplete or obscured, then the lane line can not be detected well, the robustness is poor.

3. The algorithm is poorly adapted to the environment, and improperly selected judgment thresholds can easily lead to recognition errors. Because the parameters in the algorithm of the *Hough* transform method should be set previously, and they are difficult to set, it is difficult to choose the parameters that are suitable for different road conditions.

6.2 Future direction

Using deep learning methods to detect lane lines is a great idea. Deep learning is a new field in machine learning, it is enlightened by neural networks in the human brain and aims to mimic the human brain to build neural networks that mimic brain mechanisms to interpret data such as images, sounds, text, etc.

Lane line detection can be seen as a combination of target detection and target segmentation. For lane line detection, the deep learning approach extracts more essential features of the target by building artificial neural network models that are trained using massive datasets.

The advantage of the deep learning method is that the target features can be automatically learned during the training of the model with automatically corrected parameters, avoiding the drawbacks of *Hough* transform methods such as the need to define parameters manually and poor environmental adaptation. And deep learning method can detect curved lane lines, which is more suitable for the different and complex road conditions[14].

References

[1] Z. Xu, B.S. Shin. Accurate and robust line segment extraction using minimum entropy with Hough transform[J]. IEEE Trans, 2015, 24(3): 813-822

[2] B. Fardi, G. Wanielik. Hough transformation based approach for road border detection in infrared images[C]. IEEE Intelligent Vehicles Symposium, Parma, 2004, 549-554

[3] A. Abdelhamid, B. Azzedine. Lane detection and tracking system based on the MSER algorithm[C]. Proceedings of the 17th ACM international conference on modeling, New York, 2014, 259-266

[4] J. Tang, S. Li, and P. Liu, "A Review of Lane Detection Methods based on Deep Learning," Pattern Recognition, p. 107623, 2020/09/15/ 2020.

[5] Luo Yang. Lane Line Detection in Complex Environments [D]. University of Electronic Science and Technology, 2020.

[6] L. Jiang, J. Li and W. Ai, "Lane Line Detection Optimization Algorithm based on Improved Hough Transform and R-least Squares with Dual Removal," 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2019, pp. 186-190, doi: 10.1109/IAEAC47372.2019.8997573.

[7] Zhang YG, Yang JH. Lane line detection based on inverse perspective transform and Hough transform[J]. Journal of Yunnan University (Natural Science Edition),2009,31(S1):104-108.

[8] R. K. Satzoda, S. Sathyanarayana, T. Srikanthan and S. Sathyanarayana, "Hierarchical Additive Hough Transform for Lane Detection," in IEEE Embedded Systems Letters, vol. 2, no. 2, pp. 23-26, June 2010, doi: 10.1109/LES.2010.2051412.

[9] BAI Bingjie, HAN Junfeng, PAN Shenghui, LIN Chuan, YUAN Huidong. A dual-threshold segmentation-based lane line detection method[J]. Information Technology,2013,37(03):43-45.

[10]WANG Wenhao,GAO Li. An OpenCV-based lane line detection method[J]. Laser Journal,2019,40(01):44-47.

[11] H. Li, Z. Zhang, X. Zhao and M. He, "Two-stage Hough transform algorithm for lane detection system based on TMS320DM6437," 2017 IEEE International Conference on Imaging Systems and Techniques (IST), 2017, pp. 1-5, doi: 10.1109/IST.2017.8261489.

[12] SATZODA R K, SATHYANA R AYANA S, S R IKANTHAN T, et al. Hierarchical additive Hough transform for lane detection [J]. IEEE Embedded Systems Letters, 2010, 2(2):23 - 26.

[13] Cai YF, Gao L, Chen L, Yuan J, Chen J. A lane line detection method based on clustering algorithm[J]. Journal of Jiangsu University (Natural Science Edition),2017,38(06):621-625.

[14] He XG, Jiang L, Luo YP, Zhang WW. Design of lane line detection algorithm based on Hough transform[J]. Agricultural equipment and vehicle engineering,2019,57(11):90-91+107.

[15] Liu M. Research and implementation of grayscale algorithm for colour images[D]. Changchun University of Technology,2019.

[16] Wu Xi. Research and implementation of MATLAB-based image edge detection algorithm[D]. Jilin University,2014.

[17] PhaniK, TejaswiD, ShanmukhaR, etal. GPU accelerated automated feature extraction from satellite images[J]. International Journal of Distributed and Parallel Systems, 2013, 4(2): 1-15.

[18]P. Hsiao, C. Yeh, S. Huang and L. Fu, "A Portable Vision-Based Real-Time Lane Departure Warning System: Day and Night," in IEEE Transactions on Vehicular Technology, vol. 58, no. 4, pp. 2089-2094, May 2009,
doi: 10.1109/TVT.2008.2006618.

[19] Cha, Junlin. Research on vision-based road image processing and lane line detection [D]. Chongqing University,2019.

[20] J. Kim, S. Kim, S. Lee, T. Lee and J. Lim, "Lane recognition algorithm using lane shape and color features for vehicle black box," 2018 International Conference on Electronics, Information, and Communication (ICEIC), 2018, pp. 1-2, doi: 10.23919/ELINFOCOM.2018.8330549.

[21] Fu LJ, Lan FP. Real-time lane line detection algorithm with improved Hough transform coupled with edge information[J]. Journal of Electronic Measurement and Instrumentation,2019,33(08):166-172.

[22] Hui Liu. Research on structured lane line detection method in multiple scenes [D]. Shanghai University of Engineering and Technology,2017.

[23] Shi, Linjun, Yu, Su. A Hough transform lane line detection method based on multiple constraints[J]. Computer Measurement and Control,2018,26(09):9-12+38.

[24] D. Chowdhury, S. K. Das, S. Nandy, A. Chakraborty, R. Goswami and A. Chakraborty, "An Atomic Technique For Removal Of Gaussian Noise From A Noisy greyscale Image Using LowPass-Convoluted Gaussian Filter," 2019 International Conference on Opto-Electronics and Applied Optics (Optronix), 2019, pp. 1-6, doi: 10.1109/OPTRONIX.2019.8862330.

[25] ZHANG Hongxia,WANG Can,LIU Xin,SHEN Kaiyi,FU Xiujuan,WANG Xuehua. New advances in image edge detection algorithm research[J]. Computer Engineering and Applications,2018,54(14):11-18.

[26] Zhao Lingjun, Jia Chengli, Kuang Zhangyou. A review of SAR image edge detection methods[J]. Chinese Journal of Image Graphics, 2007, 12(12): 2042-2049.

[27] Asatryan D G, Patera J.Edge- detection algorithm based on DCT continuous extension technique[J].Elementary Particles and Fields, 2008, 71(5)： 795-799.

[28] Awad A， Man H.Similar neighborhood criterion for edge detection in noisy and noise- free images[C]//Proceedings of the International Multiconference on Computer Science and Information Technology, 2008(3)： 483-486.

[29] Steger C, Ulrich M, Wiedemann C.Machine vision algorithms and applications[M].[S.l.]: John Wiley & Sons, 2007.

[30] Gu J, Pan Y, Wang H.Research on the improvement of image edge detection algorithm based on artificial neural network[J].Optik- International Journal for Light and Electron Optics, 2015, 126(21): 2974-2978

[31] Asp F, Olofsson A , Berninger E.Corneal-reflection eye tracking technique for the assessment of horizontal sound localization accuracy from 6 months of age[J].Ear & Hearing, 2015, 37.

[32] Jin Chun, Li Yaping.Research of gaze point compensation method in eye tracking system[C]//Proceedings of the 2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics.Zhejiang: IEEE Computer Society, 2015: 1174-117

[33] He Q, Zhang Z.A new edge detection algorithm for image corrupted by White-Gaussian noise[J].AEU- International Journal of Electronics and Communications , 2007, 61(8): 546-550.

[34] WANG Yunyan,ZHOU Zhigang,LUO Lengkun. Image enhancement algorithm based on Sobel operator filtering[J]. Computer Applications and Software,2019,36(12):184-188.

[35] Hu, Xu-Yi, Wang, Chao, Li, Dan. Research on edge detection algorithm based on improved Sobel operator[J]. Fujian Computer,2018,34(09):13-15

[36] Yang Z.Y.,Wang Y.,Wang C.D. An improvement scheme for LOG operator edge detection method[J]. Computer Applications and Software,2004(09):87-89+83.

[37] Xu Jianhua, Image processing and analysis [M] , Beijing:Science Press, 1992.

[38] Li Jiegu, Theory and practice of computer vision [M] , Shanghai: Shanghai Jiaotong University Press, 1991.

[39] Dong Xue,Lin Zhixian,Guo Tailiang. An improved adaptive threshold wavelet denoising algorithm based on LoG operator[J]. Liquid Crystal and Display,2014,29(02):275-280.

[40] Qian RW, Ma Qiang, Xu Dandan, Zhou Han. Relative displacement detection of iron shoes based on improved DOG edge detection operator[J]. Combined Machine Tools and Automatic Machining Technology,2021(03):94-97.

[41] H. Zhang, J. Han, Y. Guan and Y. Zhang, "A SIFT Algorithm Based on DOG Operator," 2018 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS), 2018, pp. 609-612, doi: 10.1109/ICITBS.2018.00159.

[42] ZHANG Lu, GUO Qingyu, LIANG Jingjing, LIU Weiguang. An edge extraction and tree age measurement method for annual rotation images based on DoG operator[J]. Journal of Zhongyuan Engineering College,2019,30(06):50-54+90.

[43] Tian Baozhong. Blind deconvolution and deblurring of images based on Gaussian difference operator[J]. Intelligent Robotics, 2017(3): 58-61

[44] Liu Jilei, Li Qiangzi, Du Xin. Remote sensing recognition of snow disturbance traces based on Gaussian difference model[J]. Remote Sensing Technology and Applications, 2015, 30(1): 140-147

[45] Qian RW, Ma Qiang, Xu Dandan, Zhou Han. Relative displacement detection of iron shoes based on improved DOG edge detection operator[J]. Combined Machine Tools and Automatic Machining Technology,2021(03):94-97.

[46] J. Canny. A computational approach to edge detection. [J],IEEE Transactionson Pattern Anlaysis and Machine Intelligence,1986(6):678-69

[47] Chao Zhou. Research and improvement of the Canny operator for edge detection[D]. Chongqing Normal University, 2012.

[48] HORNUNG A, KOBBELT L. Hierarchical Volumetric Multi-view Stereo Reconstruction of Manifold Surfaces Based on Dual Graph Embedding[C]//Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. Washington D. C., USA: IEEE Computer Society, 2006: 503-510.

[49] Li Yadi, Huang Haibo, Li Xiangpeng, Chen Liguo. Nighttime lane line detection based on Canny operator and Hough transform[J]. Science Technology and Engineering,2016,16(31):234-237+242.

[50] Fan Hei, Fei Shengwei, Chu Youbing. An improved image edge extraction algorithm based on Canny operator[J]. Automation and Instrumentation, 2019,34(01):41-44.

[51] Song R J,Liu Chao,Wang Baojun. An adaptive Canny edge detection algorithm[J]. Journal of Nanjing University of Posts and Telecommunications (Natural Science Edition),2018,38(03):72-76.

[52] PERRONNIN F, MENSINK T, VERBEEK J. Image classification with the fisher vector: Theory and practice[J]. International Journal of Computer Vision, 2013, 105(3): 222-245.

[53] C. Y. Fang, C. S. Fuh, P. S. Yen, et a1. An automatic road sign recognition system based on a computational model of human recognition processing[J]. Computer Vision&Image
Understanding, 2004, 96(2):
[54] Zhang Luyao, Liang Yuming, Zhang Tianlu. An improved algorithm for lane line detection based on OpenCV[J]. Information Technology and Informatization,2019(12):108-111.
[55] Mejdani S, Egli R, Dubeau F. Old and new straight-line detectors[J]. Pattern Recognition, 2008, 41(6):1845-1866.
[56] Liu Tianhui. Research on road detection algorithm in vehicle vision navigation[D]. Shenyang University of Technology,2015.
[57] Xie Xie. Vision-based lane line detection and recognition[D]. Wuhan University of Technology, 2013.
[58] Wen Yunyan, Yang Weijun, Nie Yongyi, Bu Guofu, Peng Xiaoxin, Zheng Yuhuang, Yu Jingxiao. Research on lane line detection in automatic driving[J]. Industrial Control Computer,2020,33(10):80-81+84.
[59] P. Maya and C. Tharini, "Performance Analysis of Lane Detection Algorithm using Partial Hough Transform," 2020 21st International Arab Conference on Information Technology (ACIT), 2020, pp. 1-4, doi: 10.1109/ACIT50332.2020.9300083.
[60] Yuan J, Tang S, Pan X, etal. A robust vanishing point estimation method for lane detection[A]. Control Conference[C]. IEEE, 2014.
[61] Qu ZG, Niu SQ. A research on an improved adaptive weighted median filtering algorithm[J]. Computer Technology and Development,2018,28(12):86-90.
[62] Guo Keyou,Wang Yewei,Guo Xiaoli.A lane line classification detection algorithm combining LDA and LSD[J]. Computer Engineering and Applications,2017,53(24):219-225.
[63] Weirong Liu,Shutao Li,Xu Huang. Extraction of lane markings using orientation and vanishing point constraints in structured road scenes[J]. International Journal of Computer Mathematics,2013,91(11).

[64] Wang X. Research on straight line extraction algorithm [D]. University of Defense Science and Technology, 2013.

[65] Xu Bangbang. Research on lane line recognition technology[D]. University of Electronic Science and Technology, 2020.