Nighttime Lane Detection and Vehicle Position Determination based on OpenCV

Leilei Chen*, Yingshun Liu, Weiye Zhang
Nanjing University of Science and Technology, No.200 Xiaoling Wei Street, Nanjing City, 210094, Jiangsu Province, China
*corresponding author email: 122110223429@njust.edu.cn

ABSTRACT

Lane detection plays a significant role in computer vision and autonomous driving applications. However, it encounters challenges under low-light conditions during nighttime, where images are darker and real-time processing is necessary. In this study, a nighttime lane detection method based on OpenCV is proposed. Initially, image contrast stretching is applied to enhance brightness as a preprocessing step. By adjusting the grayscale range of the image, lane line features can be captured more effectively. Median filtering and bilateral filtering are then employed to reduce noise interference in the image. To address potential left-right deviation caused by the camera, a novel dynamic Region of Interest (ROI) method is introduced. This method adaptively adjusts the ROI based on real-time image analysis, thereby reducing false detections and improving overall detection performance. Canny edge detection and Hough transform are utilized to locate the lane lines. Finally, lane fitting and drawing techniques are applied to determine the vehicle's position within the lane. Experimental results demonstrate the high accuracy and robustness of the proposed nighttime lane detection method, which incorporates enhanced images and the dynamic ROI approach. The method accurately identifies the vehicle's lane position, providing essential visual guidance for autonomous driving systems.

Keywords: lane detection, dynamic ROI, Canny edge detection, Hough transformation, autonomous driving

1. INTRODUCTION

The rapid development of autonomous driving technology is transforming our transportation methods and travel experiences. As a crucial component of autonomous driving systems, lane detection plays an indispensable role in the fields of computer vision and intelligent transportation. Lane detection assists autonomous driving systems in ensuring that vehicles stay on the correct driving lanes. By promptly detecting any deviation from the lane lines, the autonomous driving system can take appropriate measures to correct the deviation and avoid potential traffic accidents.

However, lane detection faces numerous challenges during nighttime driving due to limited lighting conditions. Garg et al.[1] proposed an image processing method called gamma correction to address the issue of dark lane lines caused by low brightness at night. Gamma correction enhances contrast and amplifies the image, allowing for better lane recognition in low-light conditions. Liu et al.[2] tackled the problem of low-quality images by employing histogram equalization as a preprocessing technique. Histogram equalization significantly improves image quality, accuracy, and robustness. Sharma et al.[3] started with camera calibration using chessboard images to correct distortion from the input initial image. They also utilized a bird's-eye view transformation to reduce information loss caused by camera issues, thereby enhancing the acquired information content. Wang et al.[4] proposed a lane detection algorithm that combines lane pixel gradient and color filtering in an interest model to meet the real-time requirements of lane detection. The effectiveness of this algorithm was verified through experiments. These research efforts demonstrate various approaches to enhance lane detection under challenging nighttime conditions, including gamma correction, histogram equalization, camera calibration, bird's-eye view transformation, and interest model-based lane detection algorithms. Liu et al.[5] proposed a method that utilizes the least squares adjustment technique after Hough transform to refine lane lines. This method improves computational speed without sacrificing algorithm accuracy, making it practical for real-time applications. Stević et al.[6] introduced an algorithm that detects lane edges using color thresholding and combines perspective transformation with Hough transform to address the issue of blurry lane lines. The algorithm achieves satisfactory detection results. Xing et al.[7] proposed a method that combines Hough transform with Shi-Tomasi corner detection to enhance lane detection accuracy significantly. Li et al.[8] improved upon the basic Hough transform by employing the Progressive Probabilistic Hough Transform (PPHT) for lane line identification. Simulation results demonstrated the effectiveness of this method in lane detection.

International Conference on Smart Transportation and City Engineering (STCE 2023),
edited by Miroslava Mikusova, Proc. of SPIE Vol. 13018, 1301812
 · 0277-786X · doi: 10.1117/12.3024000

This study proposes a nighttime lane detection method based on OpenCV to address various issues related to nighttime lane detection. The method incorporates key techniques such as image contrast stretching, filtering processing, and dynamic ROI searching to improve the accuracy and robustness of nighttime lane detection. The specific procedure is illustrated in Figure 1.

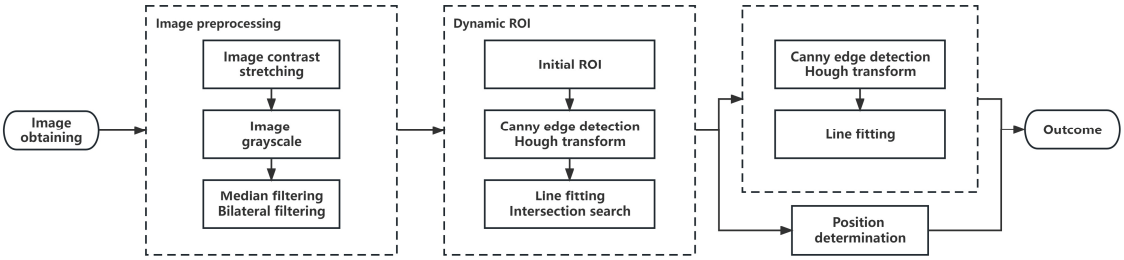

Figure 1. The specific process of detecting and determining the position of vehicles at night lane lines

2. IMAGE PREPROCESSING

2.1 Image contrast stretching

Before performing contrast stretching, the original image, which is a three-channel RGB color image, needs to be split into three separate channels. Next, contrast stretching is applied to each channel individually. Finally, the contrast-stretched channels are merged back together to obtain the final contrast-stretched color image. This ensures that appropriate contrast stretching is applied to each color channel, enhancing the visual effect and detail representation of the entire color image. Through contrast stretching, smaller pixel values and larger pixel values are both mapped, expanding the dynamic range of pixel values and enhancing the image's contrast. The specific code implementation steps are as follows. The processed result is shown in Figure 2.

1. Channel Separation: Separate the RGB channels using the cv::split function.

2. Find the Minimum and Maximum Pixel Values: Traverse the entire image using cv::minMaxLoc() to find the minimum pixel value (minVal) and maximum pixel value (maxVal) in the image.

3. Determine Thresholds: Determine the minimum brightness value (contrastMin) and maximum brightness value (contrastMax) for stretching.

4. Linear Mapping of Pixel Values: Use the cv::convertTo() function to scale each pixel value in the image. The key is to set the parameters alpha and beta in the function to achieve contrast stretching. The parameter alpha controls the scaling factor, while the parameter beta adjusts the offset. Specifically:

$$alpha = \frac{contrastMax - contrastMin}{\max Val - \min Val} \tag{1}$$

$$beta = \frac{\min Val \times (contrastMax - contrastMin)}{\max Val - \min Val} + contrastMin \tag{2}$$

5. Channel Merging: After performing contrast stretching on each channel, use the cv::merge() function to merge the separated RGB channels back together, resulting in the final enhanced color image.

2.2 Image grayscale

Image grayscale conversion is the process of converting a color image to a grayscale image, where each pixel's color information is transformed into a corresponding brightness value. This significantly reduces the data size and memory usage of the image, improving computational efficiency. Grayscale conversion lowers the complexity of the image by removing color information, making it more concise and easier to process. Moreover, when dealing with nighttime images, converting them to grayscale can reduce the interference caused by factors such as color and lighting, thus improving the robustness and effectiveness of algorithms. The specific code implementation involves using the cv::cvtColor() function in OpenCV to convert the contrast-stretched image to a single-channel grayscale image.

$$Gray = 0.299 \times R + 0.587 \times G + 0.114 \times B \tag{3}$$

Figure 2. The upper left corner is the original image, the upper right corner is the image after contrast stretching, the lower left corner is the image after grayscale processing, and the lower right corner is the image after median filtering and bilateral filtering

2.3 Median filtering and Bilateral filtering

The purpose of filtering an image is to change or enhance specific attributes of the image or reduce noise present in the image. Filtering operations can be achieved by applying various filters. In this study, a combined approach of median filtering and bilateral filtering is utilized, which offers the following advantages:

1. Comprehensive denoising effect: Median filtering and bilateral filtering are suitable for different types of noise. Median filtering effectively removes salt-and-pepper noise, while bilateral filtering smooths the image while preserving edge information. By combining both methods, the influence of different types of noise can be comprehensively considered, resulting in a more comprehensive denoising effect.

2. Edge preservation: Bilateral filtering is capable of preserving the sharpness of edges while smoothing the image, thus avoiding edge blurring. By first applying median filtering to remove noise and then performing bilateral filtering, the protective ability of edge preservation can be further enhanced, ensuring accurate retention of the edge information in the image.

3. Adaptive adjustment: Bilateral filtering has the ability to adaptively adjust the filter based on the similarity between pixels through weighted calculations. This allows for flexible processing of details in different regions, resulting in a more natural overall filtering effect that adapts to the characteristics of the image.

4. Consideration of image details: Median filtering can preserve image details while removing noise, and bilateral filtering can smooth the image while maintaining clarity of details. By combining these two filters, a better balance can be achieved between denoising and detail preservation, resulting in an image with good visual quality and minimal noise interference.

The principle of bilateral filtering is that it considers both the spatial and value domains. It achieves effective noise reduction while preserving edges well. This is due to the fact that the filter kernel consists of two functions: a spatial domain kernel and a value domain kernel.

The principle of median filtering is shown in Figure 3.

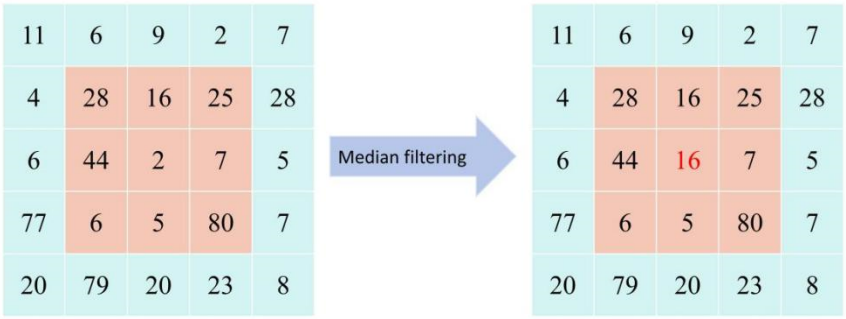

Figure 3. Median filtering involves sorting the pixel values within a local neighborhood and replacing the original pixel value with the median value, which is the middle value in the sorted sequence of pixel values

3. ESTABLISHMENT OF DYNAMIC ROI

The camera used in this study is installed at the exact center position on the top of the vehicle. However, due to certain uncertainties such as left-right camera misalignment, the position of the ROI in the input image may be uncertain. Considering this issue, this study proposes a dynamic ROI selection strategy. Compared to static ROI, dynamic ROI offers the following advantages.

1. Adaptability: Static ROI is predefined and fixed, while dynamic ROI can adaptively adjust based on the actual circumstances. Dynamic ROI can determine the region of interest by considering dynamic changes in the target's position, scale, shape, etc., enabling more effective capturing and tracking of the target.

2. Flexibility: Dynamic ROI can be flexibly applied in different scenarios and tasks. For targets with diverse shapes, dynamic ROI can be adjusted on-the-fly to accommodate different target characteristics as needed.

3. Error Reduction: Static ROI may introduce errors due to target movement or changes. Dynamic ROI, on the other hand, can adjust based on the dynamic changes of the target, thereby improving the accuracy of detection and tracking.

3.1 Establish initial ROI

In this study, an initial ROI region is established based on the placement position of the camera. Considering the actual situation, it is determined that a significant portion of the original image consists of irrelevant and interfering areas, while the key region for lane detection is primarily concentrated at the bottom of the image. Therefore, an initial rectangular ROI is set up, occupying the lower half of the image. The specific implementation process of the code is as follows.

1. Firstly, create a mask with the same size as the original image. The mask is a single-channel image with an initial value of all zeros, representing black pixels.

2. Next, define the ROI range, which is a rectangular region. Its length is the same as the original image's length, and the width is half of the original image's width.

3. Then, using the cv::rectangle() function provided by OpenCV, draw the ROI rectangle on the mask. This will set the corresponding pixel positions on the mask to non-zero values.

4.Finally, apply the mask to the original image by performing a bitwise AND operation using the cv::bitwise_and() function. This means that only when both corresponding pixels in the two images have non-zero values, the resulting image will retain the corresponding pixels, forming the ROI, as shown in the Figure 4.

3.2 Canny edge detection and Hough transform

The Canny edge detection algorithm is a classic method used to detect prominent edges in an image. The main steps of the Canny algorithm include the following.

1. Gaussian filtering: Firstly, the input image is subjected to Gaussian filtering to smooth the image and reduce the impact of noise. Gaussian filtering applies a convolution operation using a Gaussian kernel to blur the image.The formula for the Gaussian function is $G(x, y)$, where σ is the standard deviation.

$$G(x,y)=\frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{4}$$

2. Computing gradient and direction: Next, the Sobel operator is used to calculate the gradient magnitude and direction for each pixel in the image. The formulas to compute the gradient magnitude G and gradient direction θ are as follows.

$$G_x=\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix}\times A \quad G_y=\begin{bmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}\times A \tag{5}$$

$$G=\sqrt{G_x^2+G_y^2} \quad \theta=\tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{6}$$

3. Non-maximum suppression: In this step, non-maximum suppression is applied to the gradient magnitude values in order to preserve only the local maximum values within the edges and suppress other non-edge regions.

$$M'(x,y)=\begin{cases}M(x,y) & if\ M(x,y)\geq M_L\ and\ M(x,y)\geq M(x+\delta_x,y+\delta_y)\\ 0 & otherwise\end{cases} \tag{7}$$

In this context, M_L refers to the low threshold for gradient intensity, while δ_x and δ_y represent the offset values for gradient direction. After the aforementioned processing, for a set of edge points in the same direction, only one is typically retained, achieving the purpose of edge thinning or skeletonization.

4. Dual thresholding: The suppressed gradient magnitude image is subjected to thresholding, where two thresholds (high threshold and low threshold) are set to classify pixels into strong edges, weak edges, and non-edge pixels. Typically, the high threshold is used to determine strong edges, while the low threshold is used to identify weak edges. The formula for dual thresholding is as follows:

$$Canny(x,y)=\begin{cases}Strong & if \quad M'(x,y)\geq M_H\\ Weak & if \quad M'(x,y)<M_H and M'(x,y)\geq M_L\\ Non-edge & otherwise\end{cases} \tag{8}$$

5. Edge detection: In this step, the strong edge pixels are connected with adjacent weak edge pixels that have a certain connection relationship, forming complete edges. This process is known as edge tracing or edge linking.

In this study, the Canny algorithm is utilized to perform edge detection on the ROI region in the image. The resulting image after detection is shown in the Figure 4.

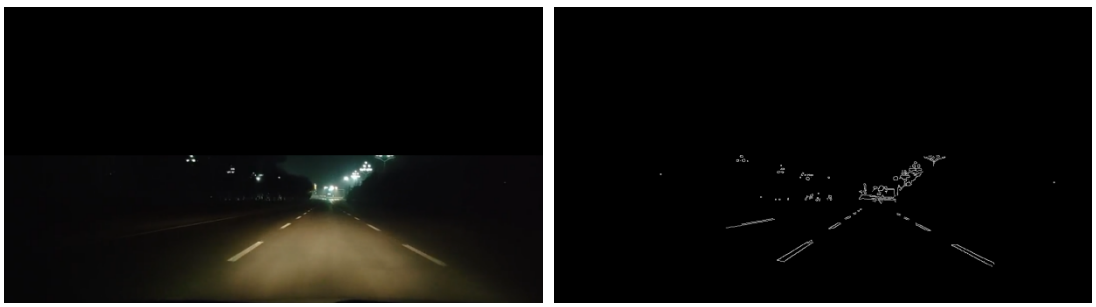

Figure 4. The left image displays the initial ROI overlaid on the original image, while the right image shows the result of applying Canny edge detection algorithm within the initial ROI

Hough Transform is a classic image processing technique used to detect geometric shapes (such as lines, circles, etc.) in an image. It achieves this by mapping the points in the image to a parameter space, which transforms the geometric shapes in the original image into curves in the parameter space. In the parameter space, the geometric shapes in the original image can be determined by finding intersections of the curves. The line Hough Transform utilizes the polar coordinate representation.

$$\rho = x\cos\theta + y\sin\theta \tag{9}$$

In the Hough Transform, a two-dimensional parameter space is used, where the X-axis represents the range of values for parameter and the Y-axis represents the range of values for parameter. For each edge point in the input image, the corresponding curve equation in the parameter space is used to perform accumulation. Ultimately, the points in the parameter space with the highest accumulation values are selected, which correspond to lines in the original image. The principle is shown in Figure 5.

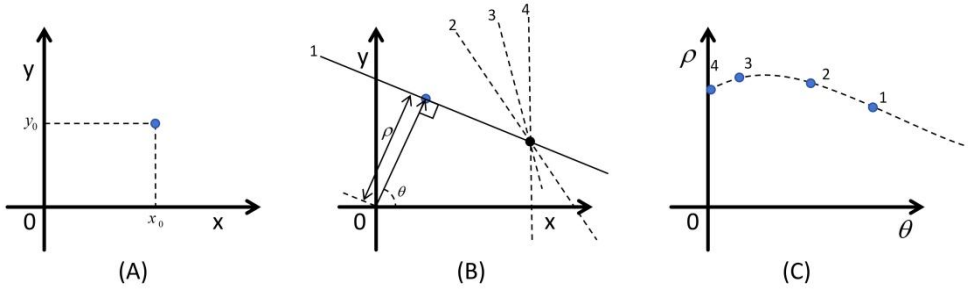

Figure 5. In the image (A), there exists a point that can represent multiple lines, where each line can be described by the parameterized equation (B) with variables ρ and θ When these lines are combined together in the parameter space (ρ,θ), they form the shape of a specific curve (C).

3.3 Line filtering

After Canny edge detection, Hough Transform, and line fitting, this study performed filtering on the obtained lines to remove redundant lines and retain only the ones that best represent the lane lines. The specific steps of the filtering process are as follows: First, all fitted lines are traversed. Then, by calculating the angles between the lines, the lines that meet the following conditions are selected: the two lines are not parallel, and their angle is less than 15°. In such cases, only one of the lines is chosen to be retained, ensuring that the final set of retained lines accurately represents the position and direction of the lane lines. Please refer to the Figure 6. for illustration.

3.4 Calculate Intersection Points

After the line filtering process, the obtained set of lines can roughly represent the position of the lane lines. Building upon this, the next step involves finding the intersections of these lines to determine their coordinates, which will be used for subsequent processing. By obtaining this set of intersection coordinates, further data processing and analysis can be performed to extract more accurate lane information. Please refer to the accompanying Figure 6. for illustration.

Figure 6. Draw the filtered directly on the original image and mark the intersection points

3.5 Dynamic ROI

Based on the obtained intersection coordinates from the previous step, locate the intersection point with the maximum y value $P(x_0, y_0)$ and record the values of x_0 and y_0. Using this point, establish a new rectangular ROI as shown in the accompanying Figure 7. Since the position of the lane lines varies in each image, the intersection coordinates also change accordingly. Therefore, this new ROI is dynamically adjusted based on the changing position of the lane lines in the image. It is referred to as a dynamic ROI. The dynamic ROI effectively filters out irrelevant information from the upper part of the image, allowing the focus to be concentrated on the road surface and lane area for more accurate lane detection and analysis.

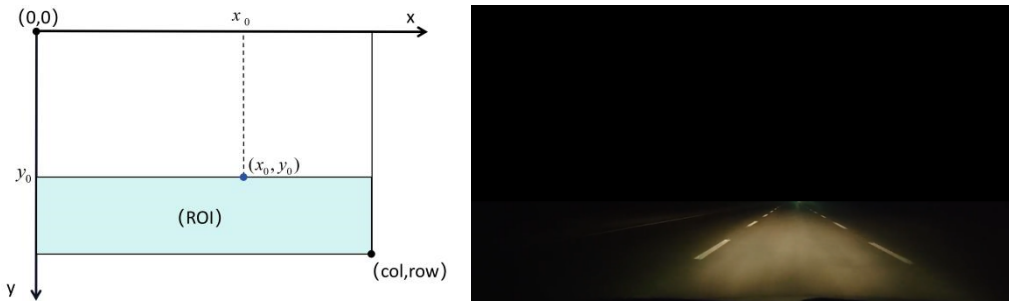

Figure 7. The left image shows the search process for dynamic ROI, while the right image shows the results

4. DETERMINE THE LOCATION AND DRAW THE LANE LINE

4.1 Location determination

Based on the obtained intersection points, select all the x of the pixel coordinates and calculate their average. This step aims to obtain the approximate average X-coordinate, denoted as x_ϕ, for all the intersection points. Then, compare x_ϕ with the pixel length of the image, denoted as $\frac{col}{2}$. By comparing these values, the half-region in the image where the vehicle is located can be determined. This comparison result provides an estimate of the vehicle's position within the lane, which helps further analyze the extent of the vehicle's deviation from the lane lines. The specific judgment ideas are as follows.

1. First, divide the length of the image into 20 equal parts to obtain a pixel length of $\frac{col}{20}$ for each part. This step is carried out to determine a threshold that facilitates the determination of the vehicle's position.

2. Next, determine the position P of the vehicle in the lane using the following formula:

$$P = \begin{cases} left & x_\phi < \frac{col}{9} \\ middle & \frac{col}{9} \le x_\phi \le \frac{col}{11} \\ right & x_\phi < \frac{col}{9} \end{cases} \tag{11}$$

4.2 Draw the lane line

Based on the dynamic ROI, perform Canny edge detection and Hough transform again. Then, conduct line fitting and filtering to obtain the final set of lane lines. Finally, draw the lane lines on the original image and mark the vehicle's position in the lane based on the determination result. The final processing result is shown in Figure 8.

Figure 8. The left image shows the results of Canny edge detection on dynamic ROI, while the right image shows the final output result

5. CONCLUSION

This research focuses on nighttime environments and explores road lane detection in particular. Lane line information extracted from a monocular camera is used as the data source. With the help of C++ programming environment and the OpenCV platform, nighttime lane detection is implemented, enabling the determination of the vehicle's position in the lane.

Given the low brightness characteristic of nighttime images, contrast stretching technique is employed to enhance the image brightness. Additionally, to address the challenges of complex backgrounds and high interference in lane detection, a dynamic ROI method is introduced. Initially, an initial ROI is established, and then a series of image transformations are applied to obtain the fitted intersection points of the lines. The dynamic ROI is adaptively updated based on these intersection points. The dynamic ROI allows for flexible adjustment based on the variation between the target and the background, enabling better differentiation and improving detection and recognition performance. Subsequently, techniques such as Canny edge detection, Hough transform, and line fitting are utilized to identify the lane lines. Finally, by comparing the found intersection points with the image, the position of the vehicle within the lane lines can be accurately determined.

Experimental validation has confirmed that this method achieves high detection accuracy, accurately detecting and drawing lane lines, as well as precisely determining the vehicle's position in the lane. Moreover, this method possesses practicality and can be applied to safety-assist driving systems and autonomous driving systems. Future research directions include vehicle deviation warning and pedestrian detection, which will effectively enhance the safety performance of autonomous driving.

REFERENCES

[1] M. Garg, A. Sehrawat and P. Savaridassan., "Vehicle Lane Detection for Accident Prevention and Smart Autodrive Using OpenCV," 2023 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2023, pp. 1-5, doi: 10.1109/ICCCI56745.2023.10128394.

[2] Y. Liu, R. Nan and W. Feng, "Lane line detection based on OpenCV," 2022 7th International Conference on Intelligent Informatics and Biomedical Science (ICIIBMS), Nara, Japan, 2022, pp. 301-304, doi: 10.1109/ICIIBMS55689.2022.9971627.

[3] Sharma, A., Vir, T., Ohri, S., Chowdhary, S.K. (2023). Road Lane Line and Object Detection Using Computer Vision. In: Shukla, A., Murthy, B.K., Hasteer, N., Van Belle, JP. (eds) Computational Intelligence. Lecture Notes in Electrical Engineering, vol 968. Springer, Singapore. https://doi.org/10.1007/978-981-19-7346-8_60

[4] Z. Wang, Y. Fan and H. Zhang, "Lane-line Detection Algorithm for Complex Road Based on OpenCV," 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Chongqing, China, 2019, pp. 1404-1407, doi: 10.1109/IMCEC46724.2019.8983919.

[5] W. Liu, C. Lu and N. Zhang, "Lane Line Detection Technology Based on OpenCV for Specific Scenarios," 2022 6th CAA International Conference on Vehicular Control and Intelligence (CVCI), Nanjing, China, 2022, pp. 1-4, doi: 10.1109/CVCI56766.2022.9964526.

[6] S. Stević, M. Dragojević, M. Krunić and N. Četić, "Vision-Based Extrapolation of Road Lane Lines in Controlled Conditions," 2020 Zooming Innovation in Consumer Technologies Conference (ZINC), Novi Sad, Serbia, 2020, pp. 174-177, doi: 10.1109/ZINC50678.2020.9161779.
[7] Xing, X., Yao, Y., & Li, J. (2022). Design and Implementation of Lane Detection Algorithm Based on OpenCV. Computer Knowledge and Technology, 18(24), 91-92+98. DOI: 10.14004/j.cnki.ckt.2022.1435.
[8] Li, J., & Zhong, P. (2021). Lane Detection Method Based on OpenCV. Journal of Huaqiao University (Natural Science), 42(04), 421-424.