Academic Journal of Science and Technology
ISSN: 2771-3032 | Vol. 10, No. 3, 2024

OpenCV-based Lane Line Detection Method for Mountain Curves

Weinuo Wang[1, a]

[1]Xi'an Jiaotong-Liverpool University, Suzhou, 215123, China
[a]Email address: weinuo.wang21@student.xjtlu.edu.cn

Abstract: In this paper, a new method based on the OpenCV computer vision library is proposed and used for the detection of mountain bike lane lines. Firstly, a series of pre-processing, HLS color filtering, grayscale map conversion, etc. are performed on the images captured by the on-board camera. Next, the lane line image is processed by Gaussian blur noise reduction, Canney edge detection and other techniques. Then, after masking for the region of interest, the lane lines are fitted after applying the Hough transform to obtain a lane line image that can be used for detection. Finally, to enhance the visual effect, the fitted lane line image is filled and displayed superimposed with the original image. The experimental results show that the proposed method has high detection accuracy and efficiency in general mountain road scenarios, and it plays a certain role in assisting the driver to drive.

Keywords: OpenCv lane line detection, HLS color filtering, Gaussian filter, Hough transform.

1. Introduction

Nowadays, the much-anticipated self-driving car is about to become one of the mainstream tools for transportation. Based on computer and modern automobile industry technology, it realizes automated driving through digitalization and intelligence, which not only brings new opportunities for the development of the automobile field in the new era, but also becomes a representative of the brand-new form of the future automobile. Artificial Intelligence (AI) is mainly used in autonomous driving technology, human-computer interaction, intelligent navigation, in-vehicle entertainment, vehicle motion control and multi-sensor fusion perception in self-driving cars. With the continuous progress of technology and the expansion of applications, self-driving cars will bring more convenient, safe, and efficient travel for human beings in the future, and revolutionize the field of transportation [1,2]. OpenCV (Open-Source Computer Vision Library) is a widely-used computer vision library, which provides a rich set of functions and tools specifically designed to be used for processing image and video data. Since its creation, OpenCV has become the de facto standard in the field of computer vision and plays an important role in various applications. The purpose of this paper is to utilize the OpenCV library to achieve real-time high-precision recognition and detection of curved lane lines in mountainous areas by calling a series of corresponding functions, to make up for the current vacancies for the detection of curved lane lines information. The specific implementation process is shown in Figure 1:

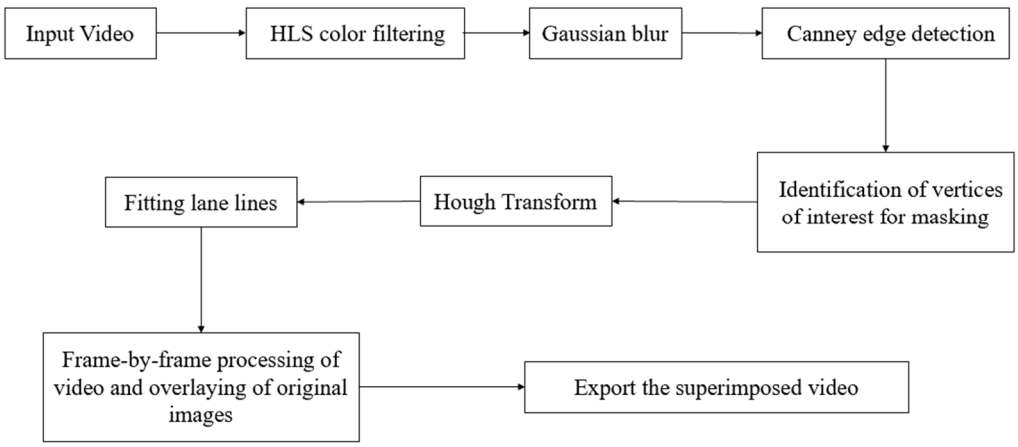

Figure 1. Overall flow of lane line detection

2. Materials and methods

2.1. HLS color filtering

Unlike the traditional method of directly converting an image to grayscale and then proceeding to the next step of processing, which is common in most papers, this paper adds a new method of HLS color filtering before converting the image to grayscale processing. The HLS (Hue, Lightness,

Saturation) color model is a commonly used method of color representation, which divides colors into three dimensions, namely hue, lightness and saturation [3]. By independently adjusting and processing the three dimensions of Hue, Lightness and Saturation, it allows us to filter colors precisely according to our specific needs, and filter out objects with specific colors that we want.

Normally, JPG files are in BGR color space. Firstly, to use HLS color filtering, we need to convert the image from BGR color space to HLS color space, so we use the cv2.cvtColor() function to achieve this purpose. Secondly, in general, most of the lane lines are white, by setting the white color threshold in advance, the lowest white lower limit [0, 138, 0], the highest white upper limit is [255, 255, 255]), and then use cv2.inRange() to generate the corresponding mask for the white image. Finally, the mask is applied to the original image by the cv2.bitwise_and () function, and only the pixels of the original image corresponding to the white part of the mask are retained, while the other parts are set to zero value. Figure 2 below shows the original image and Figure 3 shows the HLS filtered image.

Figure 2. Original image

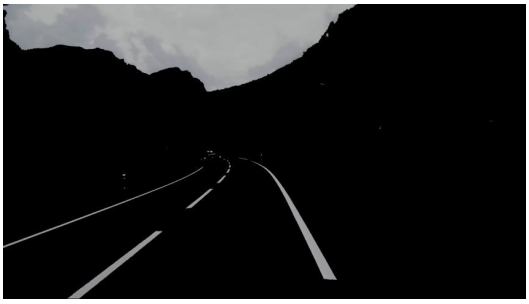

Figure 3. HLS Color Filtering

2.2. Gaussian blur

Since the system can extract the lane lines quickly and completely in the grayscale image, we grayscale the HLS color filtered ones. After grayscaling, the image may still have noise which may interfere with the recognition of features such as lane lines. Therefore, to reduce the effect of noise, Gaussian fuzzy filtering is used to process the image. This processing reduces the abrupt changes in pixel values in the image and makes features such as lane lines more legible [4]. In Gaussian blurring, the new value of each pixel is a weighted average of its surrounding pixels. The weights are derived from the calculation of Gaussian function which is specified as follows:

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

In the above equation, x and y denoting the distance between pixels represent the distance between pixels and σ is the standard deviation of the Gaussian function. The Gaussian function has the shape of a bell-shaped curve, with the center pixel having the largest weight and the surrounding pixels having decreasing weights. The larger the standard deviation, the flatter the curve and the more obvious the blurring effect. Its function in the library is cv2.GaussianBlur(), which is called to achieve the purpose of noise reduction processing and image smoothing. Figure 4 below shows the effect of Gaussian blur:

Figure 4. Gaussian Blur

2.3. Canney edge detection

The image become smoother, which allows the Canney algorithm to detect more continuous edges and avoid producing intermittent edge lines. The Canney algorithm has the advantages of high accuracy, resistance to noise, and the ability to detect fine edges. The mathematical method is to use gradient operators (usually Sobel operators) to calculate the gradient amplitude and direction of the image[5]. In the Canney algorithm, gradients represent the rate of change of pixel values in an image, while edges typically cause sharp changes in pixel values. Therefore, by calculating the gradient amplitude and direction of the image, the Canney algorithm can find potential edge positions. Next, the algorithm preserves the local maximum values in the gradient image by suppressing non maximum values on the gradient image. This step preserves the pixel with the highest local gradient amplitude and suppresses other pixels by comparing the gradient amplitudes of the pixels. After completing non maximum suppression, the algorithm further processes the gradient image by setting high and low thresholds to determine possible edge pixels and edge connecting pixels. If the gradient amplitude of a pixel exceeds a high threshold, it is considered a strong edge pixel; If the gradient amplitude is between the high and low thresholds, it is considered a weak edge pixel. Strong edge pixels are directly considered as edges, while weak edge pixels need to be connected to strong edge pixels to be recognized as edges. Finally, the algorithm connects weak edge pixels with strong edge pixels through edge tracking to form continuous edge lines. The function in the library is cv2. Canney(), with a closed value of [50-150]. Figure 5 shows the image after Canney edge detection;

Figure 5. Canny edge detection

2.4. Identification of vertices of interest for masking

The images captured by the vehicle camera contain a large amount of non-lane line information, such as sky, vegetation, clouds, etc., in addition to lane line information. In order to simplify the image processing process and improve the processing efficiency, the selection and division of the region of interest is carried out on the basis of the image after Canny edge detection. Other non-lane line information is masked out by masking. The lane line effective information in the video recorded by the vehicle camera are distributed in the lower half region of the image. Therefore, the image is cropped to retain the lower half region of the image and only that part of the region is subjected to subsequent processing. Once the region of interest is found, the selected region and the pre-selected mask are operated by bitwise and more accurate lane line information can be obtained, which removes irrelevant information from the image and makes the subsequent lane line detection more accurate and reliable. Figure 6 below shows the image after determining the vertices of interest for mask processing.

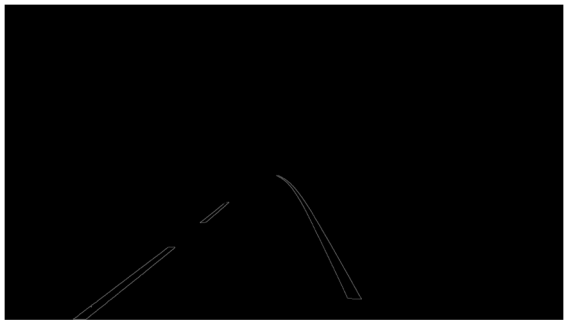

Figure 6. Identifying vertices of interest for masking

2.5. Hough Transform

The Hough transform is a technique used to detect specific shapes in an image, initially used to detect straight lines and later extended to detect other shapes such as circles or ellipses. The main idea of the Hough Transform is to map specific shapes in the image space to the parameter space and to determine the presence of specific shapes in an image by detecting accumulated peaks in the parameter space. For straight line detection, the Hough transform uses polar coordinates to represent a straight line. A straight line can be represented by its two parameters in polar coordinate space, usually the polar diameter r and the polar angle θ [6]. A two-dimensional accumulator array is then created such that each cell in it corresponds to one of the possible straight lines in the parameter space. A straight line corresponds to a curve in the parameter space whose parameters denote the slope and intercept of the line, or the polar diameter and polar angle in the polar coordinate representation. Afterwards, for each edge pixel in the image, the parameter space is traversed and the count value of the relevant cell of the straight line corresponding to that pixel is increased. This process is called voting process and each pixel on the straight line votes for the associated cell. Finally, peaks in the accumulator array are found where the cumulative value is above a threshold, which corresponds to distinct straight lines in the image, and non-maximum suppression is used to find local maxima to reduce the number of repeatedly detected straight lines. Its library function is cv2.HoughLinesP(), Figure 7 below shows the image after Hough transform.

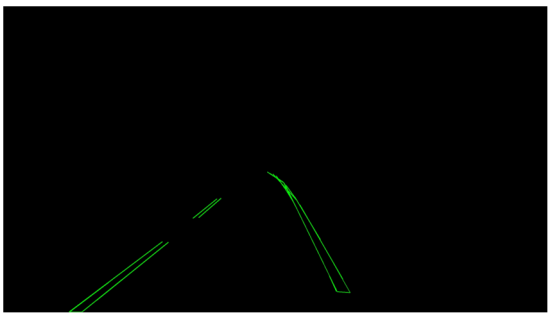

Figure 7. Hough Transform

2.6. Fitting lane lines

Finally, on the basis of the Hough transform, the fitting is performed for the lane lines. First, the set of left and right lane line points is initialized, and two empty lists left_points and right_points are used to store the coordinates of the endpoints of the detected left and right lane lines. Then traverse the detected lines and for each detected line, extract the coordinates of its two endpoints (x1, y1) and (x2, y2). Calculate the slope of the line slope, by (y2 - y1) / (x2 - x1) [7]. Some conditions are then used to filter out lines that may be lane lines, excluding vertical and overly horizontal lines if the absolute value of the slope is between 0.5 and 2.0. If the slope is negative and both endpoints of the line are located in the left half of the image, it is categorized as a left lane line. If the slope was positive and both endpoints of the line were located in the right half of the image, it was categorized as a right lane line. The lane lines were then fitted, and after enough left and right lane line points were found, i.e., left_points and right_points were non-empty, polynomial coefficients were obtained by fitting these points using the np.polyfit() function. These polynomial coefficients describe the shape of the lane lines. Finally the fitted lane lines are returned: the polynomial coefficients left_lane and right_lane are returned for the left and right lane lines. Figure 8 shows the result of fitting the lane lines:

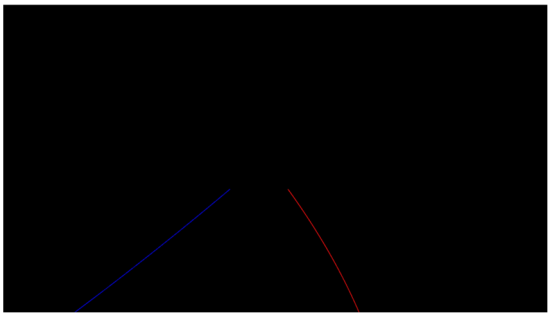

Figure 8. Fitting lane lines

2.7. Frame-by-frame processing of video and overlaying of original images

Using a loop, read the video frames (each frame is displayed at 25ms intervals in the code) and determine if they have been successfully read. Then the video frames are sequentially tested as described above, then the video frames are sequentially written to a file and the actual result of the superimposed original image is displayed, and finally the loop is skipped by waiting for the end of the video or by pressing the 'q' key.

Figure 9. Final image

3. Results and Discussion

The images used in the experiments in this paper are all from the lane line images captured by the on-board camera. The experiments are performed according to HLS color filtering, Gaussian blurring, Canny edge detection, determining the vertices of interest for masking, Hough transform, fitting the lane lines, processing the video frame by frame, and superimposing it on the original video, and finally obtaining the detected images and videos. Figures 10-11 below show the detection effect under other video frames, which demonstrates that this experiment is able to detect lane lines in mountain curves with stability and high accuracy, and has certain reference significance for driver assistance.

Figure 10. Detection results 1

Figure 11. Detection results 2

References

[1] Li, L. (2023). Research and Prospect on the Development of Autonomous Vehicle Industry. Automotive Digest, (09), 1-10.

[2] Yanhui, Y. (2020). Prospect of Future Development of Intelligent Vehicles. Southern Agricultural Machinery, 51(11), 36+38.

[3] Shan, S., Zhou, J. (2010) Exploration of license plate number localization techniques based on HLS for surveillance. Journal of Fujian Computer, 26(02),85-86.

[4] Luo, H., Cao, L., Jin, Z., et al. (2023) Research on the improvement of lane line detection algorithm based on OpenCV. Northern Communications, 02,69-73.

[5] Yuezheng, C., Wenyu, F., Yunsen, Y., Fei, X., & Hengliang, T. (2022). A new genus and species of the genus Cephalus (Hymenoptera, Staphylinidae) from China. Research on lane line detection method based on OpenCV. Micro-Nano Electronics and Intelligent Manufacturing, 4(04), 82-87.

[6] Li, Z., Zhang, H., Zhao, S. X., & Liang, A. (2020). An OpenCV-based lane line detection method. Proceedings of the 24th Annual Conference on New Network Technologies and Applications, China Computer Users Association Network Application Branch, 247-250.

[7] Liu, L., Hu, G., & Hu, J. (2023). A lane line detection method based on instance segmentation. *Computers and Digital Engineering, 51(07), 1620-1625.