Enhancing Autonomous Vehicle Navigation Through Computer Vision Techniques for Lane Marker Detection and Rain Removal

Nagavarapu, Sarat Chandra; Abraham, Anuj; Li, Sihao; Dauwels, Justin

DOI
10.1007/978-3-031-72959-1_8

Publication date
2025

Document Version
Final published version

Published in
Lecture Notes in Intelligent Transportation and Infrastructure

Enhancing Autonomous Vehicle Navigation Through Computer Vision: Techniques for Lane Marker Detection and Rain Removal

Sarat Chandra Nagavarapu, Anuj Abraham, Sihao Li, and Justin Dauwels

Abstract Autonomous Vehicles (AVs) equipped with camera systems have emerged as a pivotal solution for smart urban mobility. The escalating demand for AVs emphasizes the need to prioritize driving safety, especially in challenging weather conditions like heavy rain. In this context, the accurate perception of environmental features, notably lane markers, becomes imperative for effective autonomous navigation. Severe weather can lead to camera image degradation, including blur and loss of details, impacting the accuracy of subsequent image processing. Despite the prevalence of camera-based methods, sensitivity to environmental noise, such as rain streaks, poses a challenge, necessitating preprocessing mechanisms like rain removal to enhance lane detection accuracy. This chapter focuses on the development of a vision-based algorithm dedicated to detecting and tracking lane markers, coupled with an efficient rain streak removal algorithm. A progressive approach to lane detection on city roads is presented, incorporating sliding windows and Kalman filter methodologies into a model-based method. Integration of the Kalman filter has yielded a notable improvement in video processing speeds, from 1.67 to 2.72 frames/s, enhancing overall operational efficiency. Furthermore, a novel neural network structure, amalgamating convolutional neural networks (CNNs) and long short-term memory (LSTM), is introduced for rain streak removal before performing lane

S. C. Nagavarapu (✉)
Institute for Infocomm Research (I2R), Agency for Science, Technology and Research (A*STAR), Singapore 138632, Singapore
e-mail: saratchandra.nagavarapu@gmail.com

A. Abraham
Technology Innovation Institute, 9639 Masdar City, Abu Dhabi, United Arab Emirates
e-mail: anuj.abraham@tii.ae

S. Li
Baidu, Inc., Shenzhen, Guangdong, China 518000
e-mail: leeszehou@outlook.com

J. Dauwels
Department of Microelectronics, Faculty of EEMCS, TU Delft, Mekelweg 4, 2628 CD Delft, The Netherlands
e-mail: J.H.G.Dauwels@tudelft.nl

A. Abraham et al. (eds.), *Internet of Vehicles and Computer Vision Solutions for Smart City Transformations*, Lecture Notes in Intelligent Transportation and Infrastructure,
https://doi.org/10.1007/978-3-031-72959-1_8

marker detection. Comparative analysis against existing methods demonstrates an average 2.3% improvement in peak signal-to-noise ratio (PSNR) for rain removal and an 8% enhancement in Google Vision test results.

Keywords Autonomous vehicles · Navigation · Rain removal · Lane marker detection · Neural networks · Computer vision

1 Introduction

In smart cities and urban transportation applications, AVs have emerged as prominent modes of transportation, revolutionizing the way people move through urban landscapes [1]. Despite ongoing advancements in autonomous vehicle technology, a significant challenge yet to be overcome is ensuring safe driving in inclement weather. Operating autonomous vehicles in less-than-optimal conditions reduces visibility on the road, increasing the susceptibility of your fleet to potential accidents. The U.S. Department of Transportation reports an annual average of over 5,891,000 vehicle crashes [2], with approximately 1,235,000 linked to adverse weather conditions such as rain, fog, snow, and severe wind. Of these, rainy conditions account for the majority of weather-related accidents, comprising 46% of these crashes. A recent study by UMTRI, VTTI, GM, and Cruise [3] revealed that human ride-hail drivers in San Francisco urban areas have a crash rate of 50.5 crashes per million miles (CPMM), while self-driving cars have a lower rate of 23 CPMM. In the collected data, human drivers contributed to 69% of crashes, whereas self-driving cars contributed to only 10% of the overall crashes. Regardless of the comparatively low numbers, prioritizing the safety of autonomous vehicles (AVs) remains critical [4] for their potential replacement of human driving in the near future.

According to 2021 statistics, the United States experienced 42,939 fatal vehicle crashes, resulting in 37,133 deaths from human-driven vehicles, translating to 12.94 deaths per 100,000 people and 1.37 deaths per 100 million miles traveled [5]. Replacing human-driven vehicles with autonomous vehicles (AVs) could potentially save nearly 43,000 lives and prevent tens of billions of dollars in accident-related costs. Additionally, AVs provide significant convenience for long trips, reduce human effort, and alleviate the monotony of stop-and-go traffic in high-density urban areas.

The advancement of AV technology has led to the development of sophisticated autonomous cars equipped with advanced sensor systems and state-of-the-art algorithms supporting advanced driver assistance systems (ADAS) [6]. Among various sensors, camera-based solutions using optical sensors are the most cost-efficient and reliable for AVs. These cameras, serving as the *eyes* of autonomous perception and navigation, detect urban road scenarios such as lane markers, traffic signs, and traffic lights. They help localize road boundaries and identify lane variations and road geometry. Figure 1 illustrates the AV system framework, highlighting the camera as a primary sensing element and lane detection as a key component of the perception subsystem.

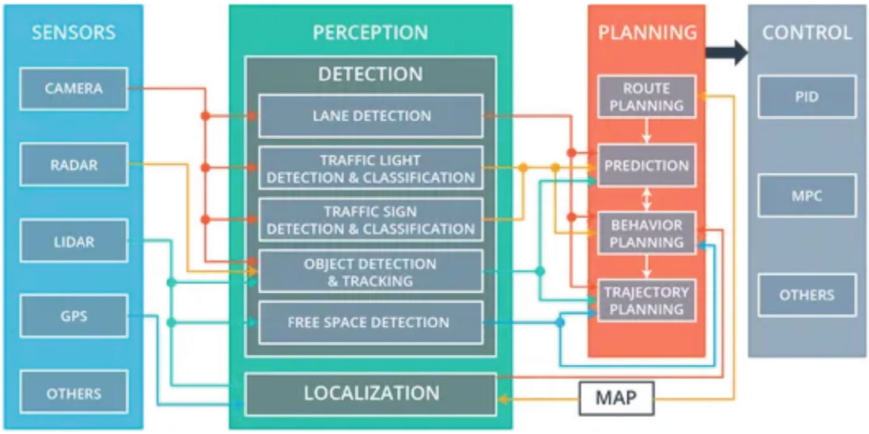

Fig. 1 Autonomous vehicle: subsystems and information flow [7].

In spite of advancements in machine learning algorithms for autonomous navigation and advanced driver assistance systems (ADAS), diverse driving scenarios, such as adverse weather conditions like rain, snow, and fog, pose significant challenges. These conditions can impair visibility and degrade image quality, compromising vision systems in target tracking, identification, and information retrieval, especially in heavy rain. Thus, removing rain streaks from camera images is crucial for accurate lane detection and enhancing the safety of autonomous navigation on urban roads.

At the core of AV operation and navigation is computer vision, which enables vehicles to accurately perceive and interpret their surroundings. This technology allows AVs to detect and respond to dynamic obstacles, remove dynamic objects from the scene [8], recognize traffic signals, and understand road layouts. These capabilities are essential for ensuring safe and efficient navigation, as computer vision techniques for lane marker detection and rain removal enable AVs to make real-time decisions and autonomously navigate complex urban environments. The following section provides a concise overview of the current advancements in rain removal and lane marker detection.

1.1 Rain Removal

Rain is a prevalent dynamic weather condition, characterized by rapid and random spatial distribution of raindrops, often resulting in rain streaks that significantly degrade visual quality. Research in this area can be categorized into video-based and single-image-based approaches. Video-based methods use temporal information but struggle with camera motion or dynamic scenes [9]. Whereas single-image methods, the focus of this work, address rain removal using only one image, making them more practical for common scenarios.

Video-based rain removal involves detecting and removing rain through visual models based on rain's optical and directional properties [10]. Techniques like frame difference detect raindrops by light intensity changes [11], and frequency-domain analysis uses rain's visual characteristics to remove it and restore pixel values [12]. Single-image rain removal employs image decomposition, rain position detection, and deep learning. Image decomposition separates images into components to isolate rainy parts using dictionary learning [13] and sparse coding [14]. Rain position detection refines raindrop positions with methods like Gaussian Mixture models [15] and image inpainting [16]. Deep learning trains convolutional networks [17] and uses generative adversarial networks (GANs) [18] to generate rain-free images.

1.2 Lane Marker Detection

In this section, we explore lane marker detection, crucial for AV navigation, advanced driver assistance systems, and robotics. Despite progress, challenges remain due to diverse road environments. We review feature-based, model-based, and deep learning methods for lane marker detection.

Feature-based methods use techniques like line segment detectors (LSD), fuzzy c-means, and vanishing point voting. A study [19] uses a conjugate Gaussian model for robustness in various scenarios. Another approach [20] employs LSD and k-means clustering, effective but time-consuming in complex environments. The Hue-Saturation-Intensity (HSI) color model [21] is robust in shadows but not in low light. A method for worn roads [22] estimates vanishing points and uses Gabor filters for improved performance.

Model-based methods establish lane line models, using the Hough transform [23] or least squares methods based on geometric characteristics. These models include straight lines, hyperbolic curves, parabolas, and spline curves [24]. A B-Snake spline curve model [25] detects various lane shapes without camera calibration. Another algorithm [26] uses inverse perspective mapping and the statistical Hough transform (SHT), supplemented by particle filtering, though it may be computationally intensive.

Deep learning methods show promise for enhancing lane marker detection. Research focuses on image semantic segmentation, dividing and recognizing image content. A neural network structure [27] combines classification, detection, and segmentation with a shared encoder for faster processing. Another approach [28] employs a fully convolutional network (FCN), integrating convolutional, upsample, and skip layers for efficient pixel-level recognition. Table 1 summarizes the advantages and relative efficacy of various methodologies employed in rain removal and lane marker detection.

The primary focus of this research work includes the following key aspects: (1) Designing a modified neural network structure, named as *rain removal by circulation* (RRBC) that combines CNN [17] and LSTM [29] to remove rain streaks before lane marker detection and tracking. (2) Developing a *progressive method* for lane detection

Table 1 Benefits and effectiveness of rain removal and lane marker detection techniques

Technique category	Benefits and effectiveness
Rain removal	
Video-based methods	Utilizes temporal information; struggles with camera motion [9]
Single-image methods	Practical for common scenarios; employs deep learning for robustness [13–18]
Lane marker detection	
Feature-based methods	Robust in various scenarios; includes LSD, fuzzy c-means [19, 20]
Model-based methods	Uses Hough transform, effective for geometric lane detection [23–26]
Deep Learning methods	Enhances accuracy with semantic segmentation; utilizes FCN for pixel-level recognition [27, 28]

on city roads by integrating classical Hough transform, sliding windows, and the Kalman filter approaches into a model-based approach. (3) Conducting experimental analysis of the proposed techniques with respect to existing techniques from the literature.

The subsequent sections of this chapter are structured as follows: Sect. 2 presents the proposed RRBC method for de-raining single images. Section 3 details the implementation of a basic lane marker detection algorithm using the classical Hough transform. It also introduces a progressive methodology for lane marker detection using a divide and rule method, and sliding windows. Section 4 provides a comparative analysis of the proposed techniques with existing approaches from the literature. Finally, Sect. 5 concludes the chapter, highlighting potential future directions.

2 De-raining Algorithm for Single Images

For autonomous vehicles operating in adverse weather conditions like rain, it is crucial to remove rain streaks from images to ensure accurate object detection and safer navigation. To address this issue, in this section, we introduce a modified neural network framework that combines CNN and RNN for rain removal. Initially, a CNN structure is employed to extract rain streak features, and by subtracting these features from the original image, a rain-free image is obtained at each stage. Next, an RNN structure passes useful information to the next stage, repeating the process.

Unlike traditional methods that decompose the rainy image into different parts for processing, our approach trains a model to learn the parameters of a function f, representing the relationship between the rain image and rain streak layers. The rain removal is conducted in multiple stages, with each stage removing rain once and using an LSTM structure to inherit useful feature maps or other relevant information from the previous stage. This approach effectively removes rain streaks in most scenarios. The next section elaborates on the rain model, a key element for rain detection in images and subsequent removal.

2.1 Rain Model

In real-world environments, raindrops are randomly dispersed in the air, appearing predominantly as rain streaks to humans and cameras alike. These rain streaks, formed under natural illumination, diminish image clarity by inducing *blur*. Establishing a rain streak model in the time domain is typically challenging. Consequently, the majority of existing literature resorts to frequency domain models, which more accurately depict rain streak characteristics and closely reflect real-world scenarios. The *blur* induced by falling rain can be mathematically represented by an integral in the time domain [30] as follows:

$$\mathrm{g(x, y, a, z, \theta, \mu)} = \int_{0}^{l(a,z)} \exp\left(-\frac{(x - \cos(\theta)\gamma - \mu_x)^2 + \left(y - \sin(\theta)\gamma - \mu_y\right)^2}{b(a,z)^2}\right) d\gamma, \tag{1}$$

where $\mu = (\mu_x, \mu_y)$ represents the coordinate of a rain streak in an image, and θ, b, l, a, z correspond to the rain streaks' orientation, width, length, diameter, and its' distance from the camera.

From the rain streak expression in Eq. (1), the scene model with rain can be given by

$$\sum_{n=1}^{n_t} g\left(x, y, a_n, z_n, \theta_n, \mu_n\right), \tag{2}$$

where n_t represents the number of rain streaks in a scene. It's evident that in the time domain, a rainy image can be represented by the summation of the visual effects of numerous rain streaks. By performing a discrete fourier transform (DFT) on Eq. (2), we obtain the model of a rainy image in the frequency domain as

$$\left\| F\left\{\sum_{n=1}^{N} g\left(x, y, a_n, z_n, \theta_n, \mu_n\right)\right\}\right\|. \tag{3}$$

Alternatively, Eq. (3) can be expressed as

$$\sum_{n=1}^{N} \left\| G\left(u, v, a_n, z_n, \theta_n, \mu_n\right)\right\|. \tag{4}$$

In Eq. (4), G represents a blurred Gaussian model, and N determines the intensity of rain in a rainy image. Subsequently, we can construct a standard model of a rainy scenario in the frequency domain as

$$R^*\left(u, v, \varphi, \theta_{\max}, \theta_{\min}\right) = \varphi \int_{\theta_{\min}}^{\theta_{\max}} \int_{a_{\min}}^{a_{\max}} \int_{z_{\min}}^{z_{\max}} z^2 \|G(u, v, a, z, \theta)\| dadzd\theta. \tag{5}$$

In Eq. (5), φ denotes the overall brightness in the image. With these models, we can create synthetic rainy images and analyse them. In the neural network method, a common approach is to utilize a rain model to decompose a rainy image O into two layers: the rainless background layer B and the rain streaks layer R [31].

$$\mathrm{O} = \mathrm{B} + \mathrm{R}. \tag{6}$$

The fundamental concept in rain removal is to eliminate the raindrop layer R to obtain rainless images. Considering various scenarios, such as different directions, widths, and densities of rain streaks, along with additional factors like fog or poor illumination, we can incorporate these elements into the model by introducing image atmosphere illumination conditions. To streamline computation, these elements can be linked using two different weights based on the layers' intensity transparencies, given by

$$\begin{gathered} \mathrm{O} = \left(1 - \sum_{i=0}^{n} \alpha_i\right) \mathrm{B} + \alpha_0 C + \sum_{i=1}^{n} \alpha_i R_i, \\ \alpha_i \geq 0, \quad \sum_{i=1}^{n} \alpha_i \leq 1, \end{gathered} \tag{7}$$

where C represents the image's atmospheric light condition, α_0 is the scene transmission, and α_i corresponds to the brightness of different rain layers. The next section presents the proposed deep learning-based approach for rain removal.

2.2 *Rain Removal by Circulation (RRBC)*

At first, a CNN structure is used to extract rain features R in the original image. The rain-free image B is obtained by subtracting R from the original image O. Since R has a simpler texture and sparse distribution, it is easier to extract using a neural network. Each layer at each stage represents a type of rain streak, allowing better mapping of rain streaks. Dilated convolution and fully convolutional networks (FCN) [32] are employed to capture more texture with a larger receptive field. Adding a squeeze-and-excitation (SE) block [33] to each layer allows for separate weighting of different dimension feature maps, improving accuracy. This method divides R into several layers, with each stage of the framework removing one type of rain streak through recursive removal, hence the name, *rain removal by circulation* (RRBC). Figure 2 illustrates the steps involved in the rain removal model using a flow chart.

It assumes that each layer's rain has the same direction, shape, and lighting conditions. We can use a function f to represent the relationship between the original image O and the rain streak R. The loss function to assess the model performance is given by:

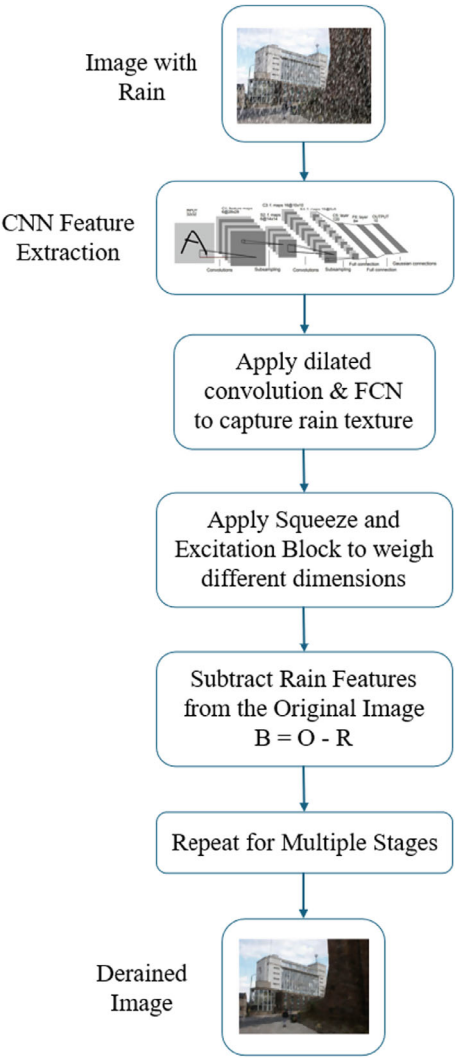

Fig. 2 Rain Removal By Circulation (RRBC): Flow chart

$$\mathrm{L}(\varphi) = \sum_{m=1}^{M} \|R_m - R\|^2, \tag{8}$$

$$\text{where } R_m = \mathrm{f}\left(O_m, X_{m-1}\right), \text{ and } O_m = O - R_{m-1}. \tag{9}$$

Here, X_m denotes the feature map of the mth stage, O_m represents the output of the mth stage, which is also the input for the next stage, and R_m signifies the predicted rain streaks for the mth stage.

2.2.1 Neural Network Structure

The neural network structure of the RRBC encompasses both CNN and RNN structures. Unlike standard RNNs, LSTM is utilized to tackle gradient disappearance during backpropagation. LSTM's multiple gates allow for effective information retention and selective processing, addressing challenges like removing randomly distributed rain streaks. Given CNN and RNN's broad applicability, this section focuses on their fundamental concepts and rationale for their inclusion in this framework.

2.2.2 CNN Structure

CNN is a supervised network model that uses convolution kernels to extract features. Its core concepts include local receptive fields, weight sharing, and down-sampling to achieve displacement, scale, and property invariance. A typical CNN consists of convolution layers, pooling layers, and activation functions. Convolution acts as an image-processing kernel that enhances certain features and reduces noise. The parameters of the convolution kernel determine the features extracted, and the resulting feature maps reflect specific image characteristics. Each layer has multiple feature maps, with each map containing multiple neurons. An $n \times m \times c$ image convoluted with b numbers of $a \times a \times c$ kernels, which yields a $L \times L \times b$ feature map. Pooling averages convolution features to reduce the hidden layers' feature dimensions.

This work uses a FCN [32] combined with SE blocks [33]. FCNs replace fully connected layers with convolution layers, enabling pixel-level classification and end-to-end training. Unlike normal CNNs, which output a probability vector, FCNs output a label image, retaining CNN's strengths such as automatic multi-level feature extraction. Shallower convolution layers learn local features with small receptive fields, while deeper layers learn more abstract features with larger receptive fields, making them less sensitive to object size, position, and orientation. Convnets operate on local regions based on corresponding coordinates, ensuring translation invariance.

We use the sigmoid (*sigm*) and hypertan (*tanh*) activation functions to endow the neural network with nonlinear expression capability, enhancing useful signals while suppressing unnecessary ones, as shown below:

$$\text{sigm: F(x)} = \frac{1}{1 + e^{-x}}. \tag{10}$$

$$\tanh : \text{F(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{11}$$

The structure of FCN is detailed in Table 2, with a depth value set to 7. To increase the receptive field, the dilation of kernels expands exponentially from layers L_2 to L_5. This larger receptive field enables the neural network to extract more contextual information. The first and last layers are designated as special layers: the first layer

Table 2 Structure of FCN

Layer	1	2	3	4	5	6	7
Convolution	3×3	3×3	3×3	3×3	3×3	3×3	1×1
Dilation	1	2	4	8	16	1	1
Receptive field	3×3	5×5	7×7	9×9	11×11	13×13	15×15
Using SE	✓	✓	✓	✓	✓	✓	X

encodes the image into a feature map, and the last layer decodes it back. As it is an FCN network, we use 3×3 convolutions in all layers except the last one. The last layer uses a 1×1 convolution to recover the RGB channels of the color images, and this is the only layer where the convolution operation is not followed by a non-linear operation. Feature maps are crucial for capturing and retaining rain streak layers, so we apply SE blocks to assign different weights (α_i) to each rain streak layer. This assigns varying weights to each channel in each layer, with more useful channels receiving larger weights to retain more information in the feature map.

2.2.3 RNN Structure

The previous structure faced gradient vanishing and disappearance issues, where early inputs had minimal impact on the network due to structural constraints. The LSTM structure, with its forget, input, and output gates, mitigates these problems by retaining useful information. While gated recurrent units (GRU) [34] and LSTM perform similarly depending on the task, LSTM's output gate allows longer retention of hidden states, making it more effective for larger datasets. LSTM is a prominent recurrent neural network structure widely used in speech recognition, image captioning, and translation. In this thesis, combining GRU with CNN led to overfitting and poor performance on the synthetic dataset. To address this, dropout is applied to the structure. The feature map X_{mn} of the nth layer and mth stage is computed using the feature map X_{m-1n} from the same layer in the previous stage and X_{mn-1} from the previous layer in the same stage, as shown in the following formula:

$$f_m^n = \text{sigm}\left(W_f^n x_m^{n-1} + U_f^n x_{m-1}^n + b_f^n\right), \tag{12}$$

$$i_m^n = \text{sigm}\left(W_i^n x_m^{n-1} + U_i^n x_{m-1}^n + b_i^n\right), \tag{13}$$

$$o_m^n = \text{sigm}\left(W_o^n x_m^{n-1} + U_o^n x_{m-1}^n + b_o^n\right), \tag{14}$$

$$\widetilde{c_m^n} = \tanh\left(W_c^n x_m^{n-1} + U_c^n x_{m-1}^n + b_c^n\right), \tag{15}$$

$$c_m^n = f_m^n \odot c_{m-1}^n + i_m^n \odot \widetilde{c_m^n}, \tag{16}$$

$$h_m^n = o_m^n \odot \tanh\left(c_m^n\right), \tag{17}$$

$$h_m^n = x_m^n, \tag{18}$$

where, *sigm* and *tanh* are the activation functions, $\odot$ represents element-wise multiplication, W denotes dilated convolutional kernels, and U is a fixed size kernel.

The depth of the gradient is the product of the activation functions' partial derivatives. If these partial derivatives are very small (<1) or zero, the gradient may vanish over time. Conversely, if the partial derivatives are large (>1), the gradient is likely to explode. This makes gradient calculation and updating challenging. LSTM addresses this by using gate functions to selectively pass information. Each gate consists of a sigmoid unit and an element-wise product operation. The sigmoid unit outputs values between 0 and 1 to decide which states or information should pass or be blocked. When the gate opens, the gradient remains close to 1, preventing vanishing gradients. Gradients will only explode if the sigmoid output exceeds 1, otherwise, they remain stable.

The next section outlines the background for lane marker detection, forming the foundation of the progressive lane marker detection algorithm.

3 Lane Marker Detection

Traditional lane marker detection involves steps such as grayscale conversion, Canny edge detection, Gaussian blur, and the Hough transform. While widely used for its convenience, this approach has drawbacks in terms of robustness and speed. To address these issues, we propose a progressive lane marker detection algorithm. Initially, distortion is removed from the RGB image. Based on the characteristics of highway lane markers, various filtering, color threshold, and morphological operations are applied to produce a binary image. The sliding windows and histogram method are then used to identify the lane markers. For video-based detection, the Kalman filter algorithm estimates the position of lane markers in the next frame, reducing computation and speeding up the process. This progressive algorithm offers enhanced reliability and robustness. The following sections present the implementation details of these two algorithms.

3.1 Basic Lane Marker Detection

This section discusses the theoretical concepts underlying the basic lane marker detection algorithm. It offers a succinct overview of the techniques integrated into the algorithm, and Fig. 3 depicts the sequential implementation of these using a flow chart.

Each phase involved in lane marker detection and the algorithmic details are thoroughly explained in the following sections.

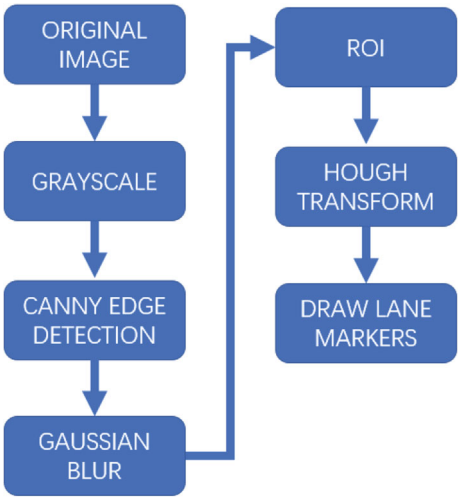

Fig. 3 Basic Lane Marker Detection Algorithm: Flow chart

3.1.1 Grayscale Conversion

Converting images to grayscale reduces the computational load for image processing by eliminating less critical information and mitigating environmental influences. In an RGB image, the R, G, and B components range from 0 to 255 and combine to form different color shades. Directly processing color images requires extensive computation and is highly sensitive to illumination and environmental conditions. In this work, we used OpenCV for grayscale conversion employing the weighted average method. Considering human eye sensitivity, with green being the most sensitive and blue the least, the weights assigned are as follows: the weight of green (G) is the highest, while blue (B) receives the smallest weight.

3.1.2 Edge Detection

For edge detection, we employed the Canny edge detection algorithm [35]. Canny edge detection is a multi-stage algorithm used for identifying the edges of lane markers in a single image. It involves applying Gaussian blur to reduce noise, finding intensity gradients to detect edges, and using non-maximum suppression to thin out the edges, followed by hysteresis threshold to track and link edge pixels. This technique effectively highlights the lane markers by emphasizing their boundaries, making them easier to detect and track.

3.1.3 Gaussian Filtering

The Gaussian filter is a type of linear filter that weights image pixels and calculates their mean, effectively removing noise that follows a normal distribution. Although similar to mean filtering in principle, the Gaussian filter uses a different kernel with σ representing the width of the kernel. It is primarily used for images with a high signal-to-noise ratio (SNR). The Gaussian filter can be represented as:

$$g(x, y) = e^{\frac{x^2+y^2}{2\sigma^2}}. \tag{19}$$

3.1.4 Region of Interest

In most car-mounted camera images captured during motion, a fitting model can represent the lane markers. Parameters of this model include the lane marker coordinates, the distance between the center line and lane markers at the image bottom, the distance from the vanishing point to the image bottom, and the image height and width. Setting the region of interest (ROI) to the area below 0.4 * H of the image is reasonable, as most lane markers are located here. This ROI reduces the detection area and computation while eliminating interference from backgrounds like the sky, trees, and buildings.

3.1.5 The Hough Transform

The Hough transform [36] is a robust feature extraction technique widely used in lane marker detection. It converts each point in the image space to a corresponding line in the parameter space. Points on the same line in image space will intersect at a single point in parameter space. In Hough space, a sinusoidal curve represents points in image space, forming a 2-dimensional parametric space with coordinates (ρ, θ). Here, ρ is the distance from the line to the origin, and θ is the angle between the line and the x-axis.

In this work, we have employed the probabilistic Hough transform (PHT) [37] for feature extraction. PHT is an enhanced version of the standard Hough transform, which reduces computational complexity by randomly sampling a subset of edge points rather than considering all of them. This method increases efficiency while maintaining robustness in detecting features such as lines in an image. The PHT outputs all the lines in the image when an accumulator bin reaches the predefined threshold. In addition to defining the image width and height, it's possible to set lane length limits, ensuring detection exclusively of lanes with specified lengths.

The basic lane marker detection algorithm performs well under good visibility conditions. The classical Hough transform effectively detects straight lines, but setting its parameters is often tricky, requiring testing and tuning based on road conditions. However, on curved roads, its performance diminishes, often misinterpreting

(a) Straight Lane with White Markings (b) Straight Lane with Yellow Markings (c) Curved Lane with Yellow Markings

Fig. 4 Basic Lane Marker Detection Algorithm: Drawback

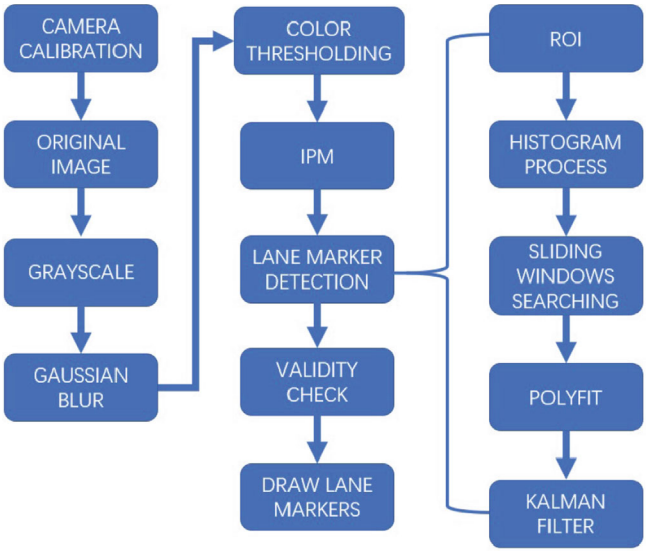

Fig. 5 Flow chart of the Progressive Lane Marker Detection Algorithm

curves as straight lines. Figure 4 illustrates the performance of basic lane marker detection and its limitations when applied to a curved road scenario.

This underscores the need to address this issue by proposing an improved and novel lane marker detection approach.

3.2 Progressive Lane Marker Detection

Figure 5 illustrates the algorithmic flowchart of the progressive lane marker detection. This improved algorithm significantly enhances performance and detects various types of lane markers under different lighting conditions. The implementation of several new techniques as part of the lane marker detection phase allows the algorithm to run faster and more accurately.

3.2.1 Inverse Perspective Mapping

Camera systems are commonly mounted on top of autonomous vehicles to provide an unobstructed view of the surroundings. In the original images, parallel lane lines appear to intersect due to the perspective factor, and the distance between them decreases with increasing distance. This perspective distortion can hinder lane marker detection and tracking. To address this, we use OpenCV to apply inverse perspective mapping [38], converting the image to a bird's-eye view. Calibration provides the camera's intrinsic (focal length and optical center) and extrinsic (yaw angle, pitch angle, and camera altitude) parameters. This transformation eliminates perspective distortion, making lane markers appear parallel and vertical, assuming a mostly flat road surface. Consequently, the Hough transform algorithm becomes more accurate and faster, reducing processing time.

We denote the optical center as C, the focal length as f, the pitch angle as α, the yaw angle as β, and the altitude as h. The camera coordinate is represented as $\{F_c\} = \{X_c, Y_c, Z_c\}$, the image coordinate as $\{F_i\} = \{u, v\}$, and the world coordinate (base frame) as $\{F_w\} = \{X_w, Y_w, Z_w\}$. The horizontal focal length is denoted as f_u, and the vertical focal length as f_v. The optical center is represented by $\{C_u, C_v\}$. The trigonometric functions are denoted as $C_1 = \cos(\alpha)$, $S_1 = \sin(\alpha)$, $C_2 = \cos(\beta)$, and $S_2 = \sin(\beta)$. We apply the homogeneous transformation T to obtain the road plane projection point P_{world} of point $P_{\text{image}} = \{u, v, 1, 1\}$ in the image plane [39] which is given by:

$$
{}^{\text{world}}_{\text{image}}\mathrm{T} = h \begin{bmatrix} -\frac{1}{f_u}C_2 & \frac{1}{f_v}S_1S_2 & \frac{1}{f_u}C_uC_2 - \frac{1}{f_v}C_vS_1S_2 - C_1S_2 & 0 \\ \frac{1}{f_u}S_2 & \frac{1}{f_v}S_1C_1 & -\frac{1}{f_u}C_uS_2 - \frac{1}{f_v}C_vS_1C_2 - C_1C_2 & 0 \\ 0 & \frac{1}{f_v}C_1 & -\frac{1}{f_v}C_vC_1 + S_1 & 0 \\ 0 & -\frac{1}{hf_v}C_1 & \frac{1}{hf_v}C_vC_1 - \frac{1}{h}S_1 & 0 \end{bmatrix}. \tag{20}
$$

And the relationship between P_{world}, P_{image} and the transformation ${}^{\text{world}}_{\text{image}}\mathrm{T}$ is,

$$
P_{\text{world}} = {}^{\text{world}}_{\text{image}}\mathrm{T}P_{\text{image}}\,. \tag{21}
$$

Also $P_{\text{world}} = \{u, v, -h, 1\}$. Using this, the inverse of the transformation ${}^{\text{world}}_{\text{image}}\mathrm{T}$ can be represented as,

$$
{}^{\text{image}}_{\text{world}}\mathrm{T} = \begin{bmatrix} f_uC_2 + C_uC_1S_2 & C_uC_1C_2 - S_2f_u & -C_uS_1 & 0 \\ S_2\,(C_vC_1 - f_vS_1) & C_2\,(C_vC_1 - f_vS_1) & -f_vC_1 - C_vS_1 & 0 \\ C_1S_2 & C_1C_2 & -S_1 & 0 \\ C_1S_2 & C_1C_2 & -S_1 & 0 \end{bmatrix}. \tag{22}
$$

By applying this transformation matrix, we can obtain the Inverse Perspective Mapping (IPM) of the original image.

3.2.2 Calibration

When using a camera, three-dimensional objects are represented as two-dimensional images. However, this conversion can introduce radial distortions, such as the barrel or the pincushion distortions, typically caused by the camera lens. These distortions can bend or shift the position of objects in the image. OpenCV can be used to correct these distortions, improving the accuracy and speed of lane marker detection. The following formula can be used to correct radial distortion:

$$X_{\text{corrected}} = x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right), \tag{23}$$

$$Y_{\text{corrected}} = y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right), \tag{24}$$

and the tangential distortion can be corrected by,

$$X_{\text{corrected}} = x + \left[2p_1 xy + p_2\left(r^2 + 2x^2\right)\right], \tag{25}$$

$$Y_{\text{corrected}} = y + \left[p_1\left(r^2 + 2y^2\right) + 2p_2 xy\right], \tag{26}$$

where $\{x, y\}$ correspond to the original image points, $\{X_{\text{corrected}}, Y_{\text{corrected}}\}$ represent the corrected image points, r represents the radial distance from the center of distortion, and $\{k_1, k_2, k_3, p_1, p_2\}$ are the distortion coefficients.

Before calibration, chessboard images captured by the camera from various positions and angles can be utilized to obtain the calibration matrix (distortion coefficients) and camera parameters necessary for IPM.

3.2.3 Filtering

Noise reduction is crucial for accurate lane marker detection. This involves adaptive filtering, color thresholding, and morphological operations. For the Udacity dataset (US roads), lane markers are either white or yellow, with distinct brightness, saturation, and hue from the road, and are not always continuous.

Different algorithms suit different conditions; extensive experimentation is needed for optimal preprocessing. Analyzing color channels in various color spaces helps separate lane markers from the background. The 'b' channel in the LAB color space effectively isolates yellow lane markers, while the 'R' channel in RGB and HSL spaces works well for white markers.

Morphological processing, particularly the *top-hat* transform, enhances feature extraction by emphasizing lane markers and reducing noise. The *top-hat* transform is defined as:

$$\text{Output\_image} = top\text{-}hat(\text{src}, \text{element}) = \text{src} - open(\text{src}, \text{element}) \tag{27}$$

The morphological opening operation, *open* for source A and element B can be expressed as:

$$A \circ B = (A \Theta B) \oplus B \tag{28}$$

The *open* operation enlarges cracks or local low-luminance regions, and subtracting the opened image from the original highlights areas brighter than their surroundings. To isolate lane markers and remove unwanted bright objects, we apply the *top-hat* operation on the isolated color channels. This operation keeps features smaller than the structuring element and brighter than their neighborhood pixels. The kernel size should be larger than the lane markers but as small as possible to effectively remove large and bright objects.

3.2.4 Thresholding

Threshold values are crucial for lane marker detection as they help reduce noise. Otsu's thresholding is popular but computes all possible values to find the best one, aiming to maximize inter-class variance. However, it can't set low thresholds based on varying image intensities, making it ineffective for objects with different brightness or saturation under varying illumination. Instead, adaptive thresholds can be used to extract lane markers, adjusting based on the characteristics of the center pixel's neighborhood.

Adaptive thresholding is useful for images with isolated bright areas like sunshine or car taillights, which can skew global thresholds. It calculates thresholds based on small image regions, adapting to varying illumination. In these regions, a pixel is selected if its value exceeds the weighted sum of its neighborhood. However, adaptive thresholding may fail by preserving the contours of bright areas surrounded by darker regions, retaining noise. This occurs because it focuses only on pixels brighter than their neighbors, whereas lane markers are typically surrounded by darker regions.

To address this, a modified adaptive threshold uses a cross-shaped kernel to compute average intensities for the left, right, up, and down directions. A pixel is selected if it has a higher intensity than its vertical or horizontal neighbors, filtering out irrelevant bright areas. This method suppresses shadows cast by objects like guardrails, improving clarity by ensuring only true lane markers pass the threshold. Additionally, *top-hat* morphology achieves similar results by eliminating bright shapes under varying illumination conditions.

3.2.5 Fitting Techniques

After thresholding, we can use a quadratic equation to fit the lane lines. The least squares method, a widely used optimization technique, minimizes the square of the errors to find the best-fitting lines for a set of points. This method ensures the smallest square error between the fitting points and actual points. First, we sum all the column pixel values to locate the highest sums on the left-hand and right-hand

sides. To improve accuracy, calculations are restricted to the region of interest. If the top of the images is overly stretched, summing only the bottom part of the images can yield more accurate results.

Next, we select the location with the highest value to start searching for valid points. Using nine small sliding windows, we identify the positions of valid pixels, which correspond to the white pixels in the images. These valid pixels are then collected into a set R shown as,

$$\mathrm{R} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3) \cdots (x_n, y_n)\}. \tag{29}$$

Next, we use a polynomial y to fit all the collected valid points,

$$\mathrm{y} = \mathrm{f(x)} = a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_n x^n. \tag{30}$$

Not all the valid points can be perfectly fit; some will have a minor sum of squared errors, given by

$$\mathrm{s} = \min \left(\sum_{i=1}^{n} [y_i - f(x_i)]^2 \right) \tag{31}$$

After implementing the one-window fitting, we calculate the mean of non-zero pixels' x-axis values of the last windows to determine the starting point for the next sliding windows search. To minimize computation, a second-order polynomial is used to fit the lane markers. Based on the distance from the center line and detected pixels, we can differentiate between the right and left lane markers. The relationship among distance d, detected pixel P_x and the image center line l_{center} is given by,

$$\mathrm{d} = \mathrm{P}_x - l_{\text{center}}. \tag{32}$$

In Eq. (32), if $d < 0$, it indicates that the pixels belong to the left-hand side, while $d > 0$ signifies that the pixels belong to the right-hand side.

3.2.6 The Kalman Filter

In video processing, the Kalman filter [40] concept can optimize lane marker detection by utilizing information from previous frames to guide the detection in subsequent frames. This approach assumes the continuity of lane lines between frames, allowing for faster and more robust detection. When previous frame information is unavailable, a sliding window method is employed to search for lane markers across the entire image or region of interest. Two separate sliding windows, one for each side (left and right), traverse from the bottom to the top of the image. These windows can slide horizontally within a predefined range. Typically, the starting points for these windows are determined by the mean values of the horizontal locations of valid pixels found in the previous search windows.

The width, height, and lateral displacement limits of these windows can be adjusted manually, affecting the performance of the detection process. In this case, a total of nine sliding windows were chosen based on experimentation to optimize performance. However, when past frame data is accessible, the search area can be narrowed, reducing processing time and enhancing robustness. By integrating the Kalman filter, the processing speed increased from 1.67 to 2.72 frames/s.

4 Simulation Results

The proposed rain removal, basic, and progressive lane marker detection algorithms have been implemented using OpenCV on a Linux-based computer with an 8-core Intel processor, 16 GB of RAM, and an inbuilt NVIDIA GPU. The Udacity dataset [41] has been used to evaluate the algorithmic performance. We use mean squared error (MSE) and PSNR as metrics for assessment, where MSE quantifies the average squared differences between the original and processed images, and PSNR measures the quality of the processed image relative to the original, with higher values indicating better performance.

Image quality encompasses two main aspects: fidelity and intelligibility. To evaluate images after rain removal, we use both objective and subjective benchmarks. Objectively, the PSNR quantifies the results effectively, representing the logarithmic measure of the mean squared error between the original and processed images relative to the square of the maximum possible pixel value, i.e., $(2^n - 1)^2$. It is expressed in decibels (dB) and calculated as follows:

$$\text{MSE} = \frac{\sum_{i=0}^{M} \sum_{j=0}^{N} \left(f_{ij} - f_{ij}^{*} \right)^2}{M \times N \times 3}, \tag{33}$$

where, M and N correspond to the number of rows and columns in the image, the factor of 3 in the denominator accounts for the three color channels (R, G, B), f_{ij} and f_{ij}^{*} are the pixel values at position (i, j) in the original and processed images, respectively.

$$\text{PSNR} = 10 \times \lg \frac{(2^n - 1)^2}{MSE}, \tag{34}$$

where n is the bit depth of the image.

Indeed, PSNR calculations typically use $n = 8$ for 8-bit images, and the denominator of MSE is 3 for color images. A higher PSNR indicates better algorithm performance, implying more effective removal of rain from the original images. However, it's important to note that PSNR values may not always align perfectly with human visual perception. Sometimes, a lower PSNR value can correspond to better visual results as perceived by humans.

Figure 6 illustrates the performance of three methods (a) CNN with GRU, (b) CNN with LSTM (depth five), and (c) CNN with LSTM (depth seven) under good

illumination conditions with small rain streaks against the original image for visual comparison. In such scenarios, all three methods perform similarly well, effectively recovering details from the original image without introducing blur. This is attributed to the sparse nature of rain streaks in these images, which contain minimal information. However, the LSTM-based approach preserves more background details compared to the GRU method, resulting in less blur in the final output under dark and high-contrast environments.

Table 3 indicates that the PSNR values, which signify image quality (higher values denote better performance), are consistently higher for LSTM compared to GRU, irrespective of whether it's the fifth or seventh depth. The overall average numbers suggest that LSTM outperforms GRU in this image group, demonstrating superior performance in terms of image quality.

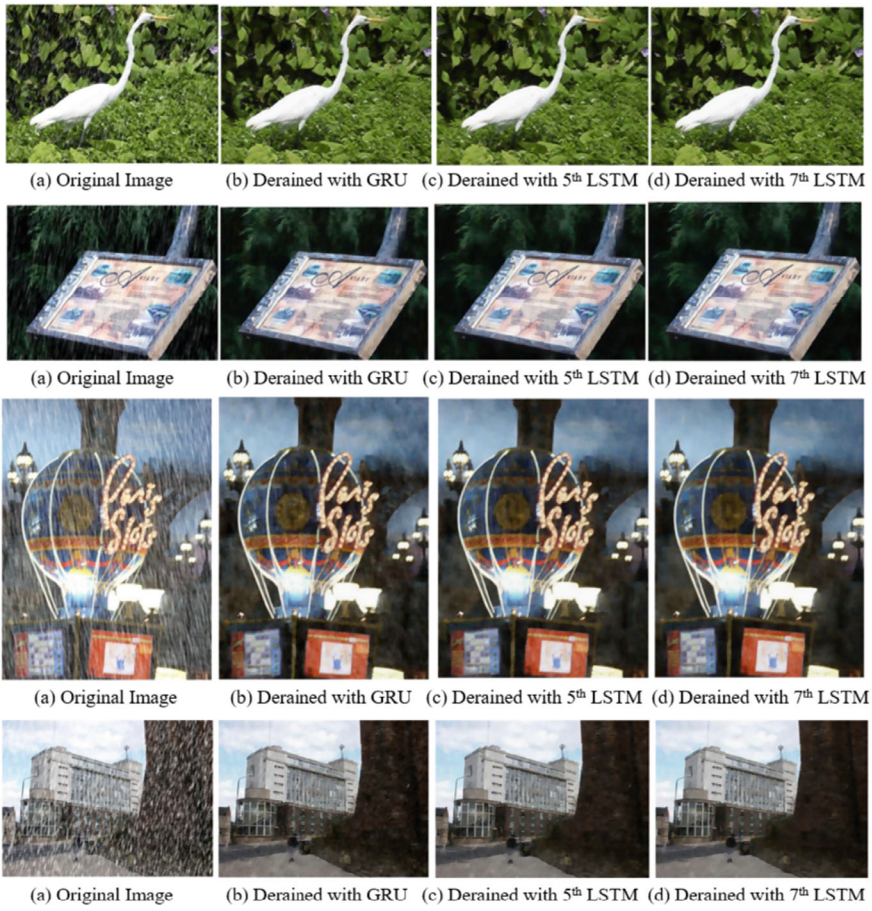

Fig. 6 Deraining using various methods

Figure 7 visually demonstrates the clearer depiction of lane markers post-rain removal, significantly aiding in their detection. Table 4 underscores this improvement by revealing higher PSNR values after rain streak removal, facilitating enhanced lane marker detection.

From the Google Vision platform test, as shown in Fig. 8, the original lane detection probability was 0.6552. After implementing the progressive derain method, the lane detection probability in the rain-free image increased to 0.7098. It is evident that after removing rain streaks, the Google Vision test results show a 9% increase in confidence for lane marker prediction and improvements in various other related metrics.

In our experiments, we observed significant improvements in both processing speed and algorithmic performance. Figure 9 demonstrates that the progressive algorithm accurately detects lane markers under various road and illumination conditions, including normal light, shadows, road repairs, false lane markings, curves, and strong light.

This shows the robustness of the algorithm, which can detect nearly all lane markers under diverse conditions. By applying different color space thresholding and validity tests, it successfully distinguishes real lane markers from false ones. By

Table 3 PSNR values of deraining among different images

Algorithm	1	2	3	4	Average
GRU	26.64	28.76	20.07	21.73	24.3
5th LSTM	26.97	28.93	20.56	21.80	24.56
7th LSTM	27.84	29.05	20.97	22.45	25.07

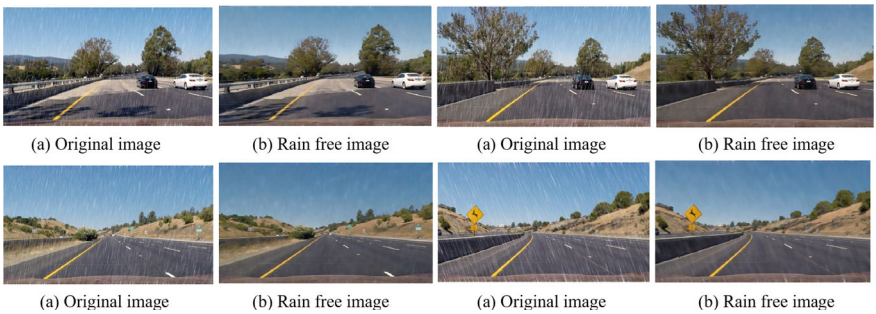

Fig. 7 Lane marker detection post image deraining

Table 4 PSNR values of lane marker detection in derained images

Algorithm	1	2	3	4	Average
GRU	33.31	30.41	31.80	27.17	30.67
5th LSTM	33.26	30.99	32.17	27.19	30.90
7th LSTM	33.76	31.01	32.71	27.76	31.31

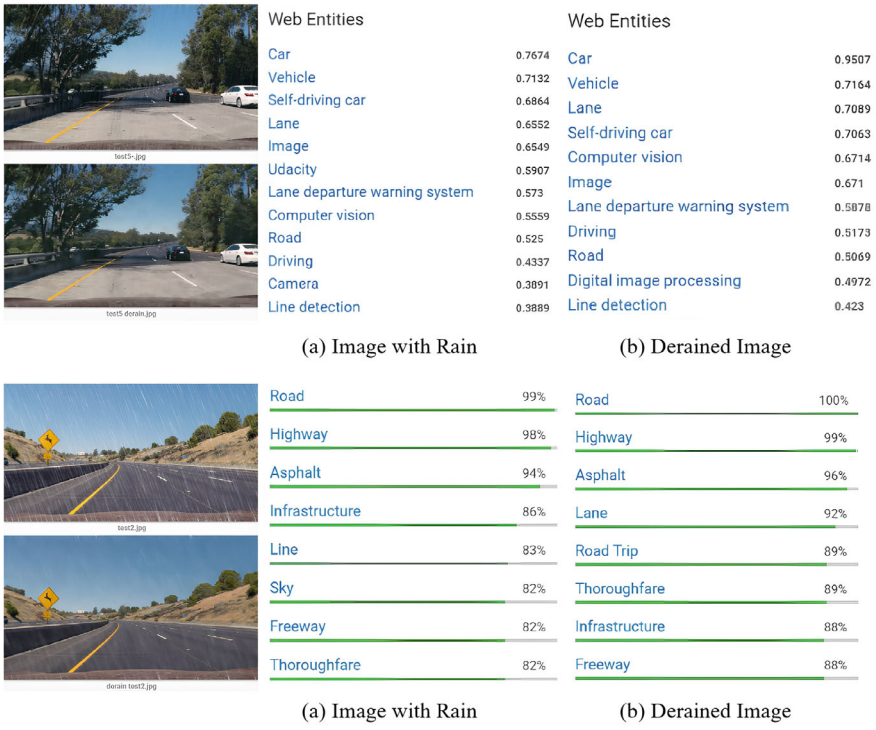

Fig. 8 Google vision test results for raining and deraining

incorporating the Kalman filter into the video processing, the video processing speed increased from 1.67 to 2.72 frames per second.

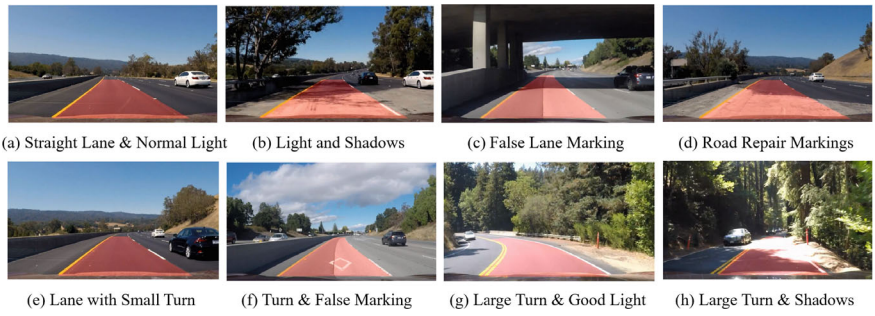

Fig. 9 Results for Progressive Lane Marker Detection Algorithm & use cases

5 Concluding Remarks and Future Outlook

In this chapter, we have presented novel approaches for lane marker detection and rain removal for AV navigation. The fusion of CNN with LSTM for rain removal has exhibited notable advancements in algorithmic performance and rain removal accuracy. Furthermore, the progressive lane marker detection algorithm, incorporating sliding windows and the Kalman filter, demonstrates robust and expedited performance, thereby improving overall operational efficiency. Simulation results conducted on diverse test datasets substantiate these findings. An average improvement of 2.3% in PSNR for rain removal and an 8% enhancement in Google Vision test results have been observed compared to existing techniques.

Although both algorithms perform well in most cases, there exist certain limitations and areas for improvement. In the case of rain removal, the presented approach is computationally expensive and requires significant GPU time for training the neural network. Similarly, in the lane marker detection method, the presented approach may not ensure accurate detection on mountain roads with continuous sharp curves. These limitations could be addressed by employing advanced GPU hardware for model training, increasing the number of epochs, and hybridizing various existing deep-learning models.

References

1. A. Abraham, P. Vyas, C.B. Math, J. Dauwels, Prioritized route patterns for autonomous vehicles using an IoT cloud in urban scenarios, in *Human-Machine Interaction and IoT Applications for a Smarter World* (CRC Press, 2022), pp. 221–234
2. V. Janakiraman, Why weather is a problem for autonomous vehicle safety (2022), https://www.geotab.com/blog/autonomous-vehicles-safety-2/. Geotab
3. L. Zhang, H. Ridehail, Crash rate benchmark (2023), https://www.getcruise.com/news/blog/2023/human-ridehail-crash-rate-benchmark/?ref=warpnews.org. Cruise
4. A. Abraham, S.C. Nagavarapu, S. Prasad, P. Vyas, L. K. Mathew, Recent trends in autonomous vehicle validation ensuring road safety with emphasis on learning algorithms, in *2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV)* (IEEE, 2022), pp. 397–404
5. National statistics (2021), https://www-fars.nhtsa.dot.gov/Main/index.aspx. FARS Encyclopedia
6. The dangers of driverless cars (2021), https://v2roads.com/businesses/ev. The National Law Review
7. D. Silver, How the udacity self-driving car works (2017), https://medium.com/udacity/how-the-udacity-self-driving-car-works-575365270a40. Medium
8. S. C. Nagavarapu, A. Abraham, N.M. Selvaraj, J. Dauwels, A dynamic object removal and reconstruction algorithm for point clouds, in *2023 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)* (IEEE, 2023), pp. 1–8
9. L.W. Kang, C.-W. Lin, Y.-H. Fu, Automatic single-image-based rain streaks removal via image decomposition. IEEE Trans. Image Process. **21**(4), 1742–1755 (IEEE, 2011)
10. K. Garg, S. K. Nayar, Detection and removal of rain from videos, in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1 (IEEE, 2004), pp. I–I

11. K. Garg, S. K. Nayar, Vision and rain. Int. J. Comput. Vis. **75**, 3–27 (Springer, 2007)
12. P. Barnum, T. Kanade, S.G. Narasimhan, Spatio-temporal frequency analysis for removing rain and snow from videos, in *Proceedings of the First International Workshop on Photometric Analysis For Computer Vision-PACV 2007* (INRIA, 2007), pp. 8-p
13. D.-Y. Chen, C.-C Chen, L.-W. Kang, Visual depth guided image rain streaks removal via sparse coding, in *2012 International Symposium on Intelligent Signal Processing and Communications Systems* (IEEE, 2012), pp. 151–156
14. B.A. Olshausen, D.J. Field, Sparse coding with an overcomplete basis set: a strategy employed by V1?. Vis. Res. **37**(23), 3311–3325 (Elsevier, 1997)
15. Q. Wu, W. Zhang, B.V.K.V. Kumar, Raindrop detection and removal using salient visual features, in *2012 19th IEEE International Conference on Image Processing* (IEEE, 2012), pp. 941–944
16. M. Bertalmio, G. Sapiro, V. Caselles, C. Ballester, Image in painting, in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (2000), pp. 417–424
17. A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks. Commun. ACM **60**(6), 84–90 (ACM, New York, NY, USA, 2017)
18. H. Zhang, V. Sindagi, V.M. Patel, Image de-raining using a conditional generative adversarial network. IEEE Trans. Circ. Syst. Video Technol. **30**(11), 3943–3956 (IEEE, 2019)
19. C.-W. Lin, H.-Y. Wang, D.-C. Tseng, A robust lane detection and verification method for intelligent vehicles, in *2009 Third International Symposium on Intelligent Information Technology Application*, vol. 1 (IEEE, 2009), pp. 521–524
20. V. Gioi, R. Grompone, J. Jakubowicz, J.-M. Morel, G. Randall, LSD: A fast line segment detector with a false detection control. IEEE Trans. Pattern Anal. Mach. Intell. **32**(4), 722–732 (IEEE, 2008)
21. T.-Y. Sun, S.-J. Tsai, V. Chan, HSI color model based lane-marking detection, in *2006 IEEE Intelligent Transportation Systems Conference* (IEEE, 2006), pp. 1168–1172
22. H. Kong, J.-Y. Audibert, J. Ponce, Vanishing point detection for road detection, in *2009 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2009), pp. 96–103
23. R.K. Satzoda, S. Sathyanarayana, T. Srikanthan, Hierarchical additive Hough transform for lane detection. IEEE Embed. Syst. Lett. **2**(2), 23–26 (IEEE, 2010)
24. S.P. Narote, P.N. Bhujbal, A.S. Narote, D.M. Dhane, A review of recent advances in lane detection and departure warning system. Pattern Recogn. **73**, 216–234 (Elsevier, 2018)
25. Y. Wang, E.K. Teoh, D. Shen, Lane detection and tracking using B-Snake. Image Vis. Comput. **22**(4), 269–280 (Elsevier, 2004)
26. G. Liu, F. Wörgötter, I. Markelić, Combining statistical Hough transform and particle filter for robust lane detection and tracking, in *2010 IEEE Intelligent Vehicles Symposium* (IEEE, 2010), pp. 993–997
27. M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, R. Urtasun, Multinet: real-time joint semantic reasoning for autonomous driving, in *2018 IEEE Intelligent Vehicles Symposium (IV)* (IEEE, 2018), pp. 1013–1020
28. J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 3431–3440
29. S. Hochreiter, J. Schmidhuber, Long short-term memory. Neural Comput. **9**(8), 1735–1780 (MIT Press, 1997)
30. P.C. Barnum, S. Narasimhan, T. Kanade, Analysis of rain and snow in frequency space. Int. J. Comput. Vis. **86**, 256–274 (Springer, 2010)
31. X. Li, J. Wu, Z. Lin, H. Liu, H. Zha, Recurrent squeeze-and-excitation context aggregation net for single image deraining, in *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 254–269
32. O. Chen, J. Xu, V. Koltun, Fast image processing with fully-convolutional networks, in *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 2497–2506
33. J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 7132–7141

34. K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation (2014). arXiv preprint arXiv:1406.1078
35. W. Rong, Z. Li, W. Zhang, L. Sun, An improved CANNY edge detection algorithm, in *2014 IEEE International Conference on Mechatronics and Automation* (IEEE, 2014), pp. 577–582
36. X. Li, Q. Wu, Y. Kou, L. Hou, H. Yang, Lane detection based on spiking neural network and Hough transform, in *2015 8th International Congress on Image and Signal Processing (CISP)* (IEEE, 2015), pp. 626–630
37. J. Matas, C. Galambos, J. Kittler, Robust detection of lines using the progressive probabilistic Hough transform. Comput. Vis. Image Underst. **78**(1), 119–137 (Elsevier, 2000)
38. Z. Ying, G. Li, Robust lane marking detection using boundary-based inverse perspective mapping, in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, 2016), pp. 1921–1925
39. L. Chen, Q.Q. Li, Q.Z. Mao, Lane detection and following algorithm based on imaging model. China J. Highway Transp. **24**(6), 96–102 (2011)
40. S. Kumar, M. Jailia, S. Varshney, An efficient approach for highway lane detection based on the Hough transform and Kalman filter. Innov. Infrastruct. Solut. **7**(5), 290 (Springer, 2022)
41. Udacity, Udacity self-driving car driving data 10/3/2016 (dataset-2-2.bag.tar.gz) (2016), https://github.com/udacity/self-driving-car
42. United Nations, Department of Economic and Social Affairs, Population Division (2018), World Urbanization Prospects 2018: The Revision. United Nations, https://population.un.org/wup/Publications/Files/WUP2018-Highlights.pdf
43. M. Cocchia, *Smart and Sustainable Cities: A Handbook for Theory* (Elsevier, Practice and Policy, 2017)
44. S. Yin et al., *Smart Cities: Technological Paradigms* (Springer, Framework Design and Applications, 2019)
45. Y. Fang, Z. Shan, W. Wang, Modeling and key technologies of a data-driven smart city system. IEEE Access, **9**, 91244–91258 (IEEE, 2021)
46. G. Halegoua, *Smart Cities* (MIT Press, 2020)