Wang *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2020) 9:27
https://doi.org/10.1186/s13677-020-00172-z

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

CNN based lane detection with instance segmentation in edge-cloud computing

Wei Wang[1], Hui Lin[2] and Junshu Wang[3,4*]

Abstract

At present, the number of vehicle owners is increasing, and the cars with autonomous driving functions have attracted more and more attention. The lane detection combined with cloud computing can effectively solve the drawbacks of traditional lane detection relying on feature extraction and high definition, but it also faces the problem of excessive calculation. At the same time, cloud data processing combined with edge computing can effectively reduce the computing load of the central nodes. The traditional lane detection method is improved, and the current popular convolutional neural network (CNN) is used to build a dual model based on instance segmentation. In the image acquisition and processing processes, the distributed computing architecture provided by edge-cloud computing is used to improve data processing efficiency. The lane fitting process generates a variable matrix to achieve effective detection in the scenario of slope change, which improves the real-time performance of lane detection. The method proposed in this paper has achieved good recognition results for lanes in different scenarios, and the lane recognition efficiency is much better than other lane recognition models.

Keywords: Double layer network, Lane line detection, Edge computing

Introduction

With the advent of autonomous driving technology, people can largely get rid of the safety problems caused by daily manual driving. Therefore, self-driving cars are sought after by many automobile consumers. In recent years, many researchers from academic institutions and industries have engaged in autonomous driving technology and these researches have promoted the development of image processing and computer vision technology. As a key part of the automatic driving system, lane detection technology is meaningful. At present, the difficulty in lane detection is how to deal with lane detection accuracy and real-time issues at the same time, so we need to improve the accuracy and efficiency of lane recognition between traditional and neural network-based lane recognition methods. The traditional computer vision-based lane detection technology is mainly based on image processing algorithms to extract the features of lane lines, reduce the image channels, perform gray processing on the original image, and then use Canny algorithm or Sobel algorithm to edge the grayed image, extract some features of the acquired image, and then perform lane line fitting after extracting the lane. Common lane fitting models include the cubic polynomial, the spline curve, and the arc curve [1–3]. At the same time, the quality of fitting is improved. Bertozzi [4] usually uses an inverse perspective transform to convert the image into a bird's-eye view, and finally performs lane tracking and other operations. Recently, deep learning techniques have been used for lane detection and can effectively detect the lane images. Based on large-scale data training, the probability of lane detection errors can be greatly reduced and detection accuracy can be improved.

Cloud computing, as a kind of distributed computing paradigm, can decompose large-scale data into submodules through the network center, allocate them to a system composed of multiple servers for processing and

*Correspondence: njnuwjs@njnu.edu.cn
[3]Key Laboratory for Virtual Geographic Environment, Ministry of Education, Nanjing Normal University, Nanjing, China
[4]Jiangsu Center for Collaborative Innovation in Geographical Information Resource Development and Application, Nanjing, China
Full list of author information is available at the end of the article

analysis, and finally feed the calculation results to the central node [5, 6]. Cloud computing system has a new resource scheduling method, and the use of the cloud computing system can effectively reduce the system computing load and improve the data processing efficiency with current computer systems, which is helpful to lane detection [7, 8]. At the same time, a dynamic resource allocation method for intensive data in the cloud, which has greatly improved the fault tolerance of cloud computing has been proposed [9]. These are the factors that cloud computing has become the current mainstream framework [10]. Based on the advantages of cloud computing technology in large-scale data processing, we decided to use cloud computing technology to improve data processing efficiency in lane detection.

Edge computing refers to a computing platform that is close to the object or data source and can integrate core capabilities such as networking, computing, and applications [11]. It has a very prominent advantage in the field of intelligent transportation [12]. To achieve safe and reliable operation, smart cars need to process a large amount of data collected by multiple sensor in real times [13]. Through edge computing, data processing will be made closer to the source [9]. Compared with cloud computing, centralizing data processing in the cloud can greatly reduce latency time and effectively improving the data processing efficiency of smart cars. In terms of security and privacy protection, edge computing solves the risk of transferring private data to the cloud and leaking user privacy under the cloud computing model, which can effectively protect data and privacy security using its distributed and mobile features [14–16]. A data privacy optimization scheme that supports edge computing [17, 18], which further strengthens the data security of edge computing has been proposed and we have combined this scheme in the lane detection process in order to further improve the efficiency of data processing.

However, it is difficult for cloud computing or edge computing to play a prominent role in lane recognition of smart cars. Current lanes processed by convolutional neural networks face the problem of low real-time performance, especially in complex scenarios [19]. Currently, few researchers use cloud computing or edge computing to analyze lane lines, while only use cloud computing needs to upload a large amount of data to the cloud for processing, which increases the load on the network and is difficult to obtain better results under the circumstances of the Internet. Therefore, we combine cloud computing and edge computing to process lane data, which greatly improves the real-time performance of lane detection during the driving process of smart cars.

The main contributions of this paper are as follows:

1) Inspired by cloud computing and edge computing, adopt more efficient data processing methods to deal with lane image recognition, the real-time performance of lane image detection is greatly improved, and the delay is reduced.

2) Based on convolutional neural network and semantic segmentation, the lane line problem in multi-lane scenes is effectively identified, and when vehicles change lanes, the current lane scene can be dynamically identified.

3) During the lane fitting process, the fixed inverse perspective transformation matrix is changed. The neural network was used to dynamically generate the inverse perspective transformation matrix, which effectively solved the problem of lane line deviation of the fixed inverse perspective transformation matrix fitting under the gradient road scene.

The remainder of this paper is organized as follows: In "Related work" section, we review the peer research and work. "Prior knowledge" section describes a priori knowledge of lane detection, including the two main frameworks of lane detection techniques in this article. "Method" section describes the specific steps of lane detection. The experimental environment and results analysis are included in "Experimental evaluation" section. Finally, we conclude our work in "Conclusion" section.

Related work

The current methods for lane detection can be divided into three main categories: road-based models, road-based features, and neural network-based models. The detection method based on the road model mainly abstracts the lane lines into geometric shapes such as straight lines, curves, parabolas, and splines, and uses different two-dimensional or three-dimensional models to determine each model parameter.

Wang et al. [20] first located the initial area of the lane line, and then converted the lane line detection problem into the problem of determining the spline model in the initial area. Tan et al. [21] proposed a robust curve lane detection method, which uses the improved river method to search for feature points in the far field of view, guided by a straight line detected in the near field or the curve of the last frame, and can connect dotted lane marks or fuzzy lane marks. Shin et al. [22] used parallel features with constant distance between the left and right lane lines in the top view, and use parallel lines to detect and track lane lines. The above method has high accuracy for detecting a specific lane line, but needs to select an appropriate model and parameters. One model is often difficult to cope with multiple road scenarios, and its generalization ability and real-time performance are poor. The detection method based on road features uses lane line and environment differences to extract lane line features for recognition and segmentation. The main features include lane line edges, color, and geometric features. Yoo et al.

[23] proposed a gradient-enhanced conversion method for robust lane detection, which converts RGB space images into grayscale images and combines Canny's algorithm to generate large gradients at the boundaries. Son et al. [24] realized lane line detection by extracting white and yellow lane lines respectively. Jung et al. [25] proposed an effective method for reliably detecting lanes based on spatiotemporal images, which improved the detection accuracy. Niu et al. [26] proposed a lane detection method with two-stage feature extraction in order to solve the problem of robustness caused by the inconsistency of lighting and background clutter, where each lane has two boundaries. The above-mentioned detection methods are easy to implement, have low complexity, and can obtain high real-time performance, but are extremely susceptible to environmental influences. In rainy days or under insufficient light, misjudgment of lane lines is prone to occur.

With the continuous development of science and technology and the continuous improvement of hardware equipment, neural network-based detection methods have made breakthrough progress. Using deep learning to detect lane lines can ensure good recognition accuracy in most scenarios [27]. Insteading of relying on highly specialized manual features and heuristics to identify lane breaks in traditional lane detection methods, target features under deep learning can automatically learn and modify parameters during the training process.

Liu et al. [28] used the mobile edge computing framework to greatly reduce the time loss of learning. Qi et al. [29, 30] significantly improved the data processing rate through the method of distributed location and multiple data sources. John et al. [31] used the learned CNN model to predict the lane line position in the image, especially in the case of occlusion and missing, CNN can extract robust features from the road image, and then train an additional tree-like regression model directly from the features estimated by lane line position. Kim et al. [32] used serial end-to-end transfer learning to directly estimate and separate the left and right lanes, and collects a dataset that includes multiple road conditions to train a CNN model. Pan et al. [33] proposed a spatial CNN (SCNN) and the convolution of the extended layer structure in the feature map is a slice structure convolution. In this way, the information transfer between pixels is performed between the rows and columns of each layer, which is suitable for strong spatial relationships, but long continuous-shaped structures or large targets lack apparent clues in this method, such as lane lines, walls, and pillars. Zhang et al. [34] applied deep learning-based convolutional neural networks to instance segmentation of monocular vision to achieve segmentation of different objects in actual scenes, and achieved better segmentation results, but less robust. The use of deep learning methods for lane line detection has greatly improved the robustness and accuracy compared with traditional detection methods, but in order to train the network, a huge data set is required as support, and the requirements for hardware facilities are correspondingly increased. Xu et al. [35, 36] proposed a video surveillance resource method when studying the Internet of Vehicles (IoV), which supports edge computing and can effectively use edge computing to solve the problem of lane image acquisition and training requirements and high network bandwidth, we combine this method in the data processing process to distribute computing power from the central node to devices with image acquisition and processing capabilities.

At the same time, we choose to convert the lane line detection problem into an instance segmentation problem, and combines edge computing to design a two-instance segmentation branch network, where the lane segmentation branch outputs the background or lane line, and the lane embedding branch separates the lane lines obtained by the segmentation branch into different lane examples to improve the accuracy of lane detection. At the same time, a custom network is designed, and the training data is marginalized to enhance the real-time nature of the data processing. A perspective transformation matrix is proposed to solve the problem of lane detection for classic neural networks when the road plane changes and helps us improve the real-time and accurate performance of the network used in this paper.

Prior knowledge

Lane prediction

Lane detection usually requires the use of relevant algorithms to extract the pixel features of the lane line, and then the appropriate pixel fitting algorithm is used to complete the lane detection. Traditional lane detection uses Canny edge extraction algorithm or Sobel edge extraction algorithm to obtain lane line candidate points and use Hough transform for lane feature detection, but most operations are based on manual feature extraction [37, 38]. The latest research is based on deep neural networks to make dense predictions instead of artificial feature extraction.

In order to enable the model to adapt to more road scenes, by analyzing the structure of classic convolutional neural networks and semantic segmentation, we use an improved two-branch network and custom function network to convert the lane line detection problem into an instance segmentation problem detection lane line, where each lane line forms its own instance. In this paper, we use a multi-branch network with two branches and the two-branch network contains a lane segmented branch and a lane embedded branch that can be trained end-to-end. Lane segmentation branches output backgrounds or lane lines, and lane embedding branches further decompose

the divided lane pixels into different lane proportions. By dividing the lane detection problem into the above two tasks, the function of lane segmentation can be fully utilized without having to assign different classes to different lanes. Lane embedding branches trained with the clustering loss function assign lanes to each pixel from the lane segmentation branch, ignoring background pixels. By doing so, the problem of lane change is solved, a variable number of lanes can be handled, and the practice of detecting a fixed lane line is improved.

At the same time, combined with edge computing, a blockchain-based edge offloading method is used to ensure real-timeness and integrity in the data transmission process [39, 40]. For real-time processing and feedback, edge computing, as a new computing paradigm, has a better ability to solve some problems such as high transmission delays, high bandwidth expenditures, and privacy leaks.

Lane fitting

To estimate the lane instance, determine which pixel belongs to which lane, we need to convert each of them into a parameter description. To this end, we use a widely used fitting algorithm. At present, the widely used lane fitting models are mainly cubic polynomials, spline curves or arc curves. Inverse perspective transformation is used to transform the image into a "bird's-eye view" to improve the quality of the fitting while maintaining the computational efficiency, and then adopt the method of curve fitting [41]. The fit lines in the "bird's-eye view" can be re-projected into the original image through an inverse transformation matrix. Generally, the inverse perspective transform calculates the transformation matrix on a single image and can remain fixed, but if the road plane changes, the fix is no longer valid, causing lane points near the horizon to be projected to infinity, which affects lane line simulation total accuracy. In order to solve this situation, we apply an inverse perspective transformation to the image before fitting the curve, and uses a loss function customized for the lane fitting problem to optimize it. A custom function network is used to generate a transformable matrix and transform the lane pixels, then use curve fitting polynomials to perform pixel fitting on the converted pixels, and finally convert the fitted pixels into the input image. The advantage of this network is that the detection algorithm can fit the pixels of distant lanes with good robustness when the road surface changes, and better adapt to lane changes.

The overall framework of the model is shown in Fig. 1. The camera on the car collects the image and transmits it to the cloud data processing center. After the image is subjected to binary segmentation and embedded segmentation, the clustering operation is performed and combined with the transformable matrix generated by the custom network to generate the final lane detection image.

Method

Binary segmentation

In lane detection, in order to save computing resources, we binarize the image. The segmented branch of the two-branch network outputs a binary segmentation map, which divides the lane line and non-lane line parts. In this paper, all ground-level car-to-car points are connected together to form the connecting line of each lane to avoid situations where lane lines are unpredictable due to lane line degradation or road obstacles blocking the lane lines. At the same time, due to the high imbalance between lane and background, we use bounded inverse class weighting for lane pixel segmentation.

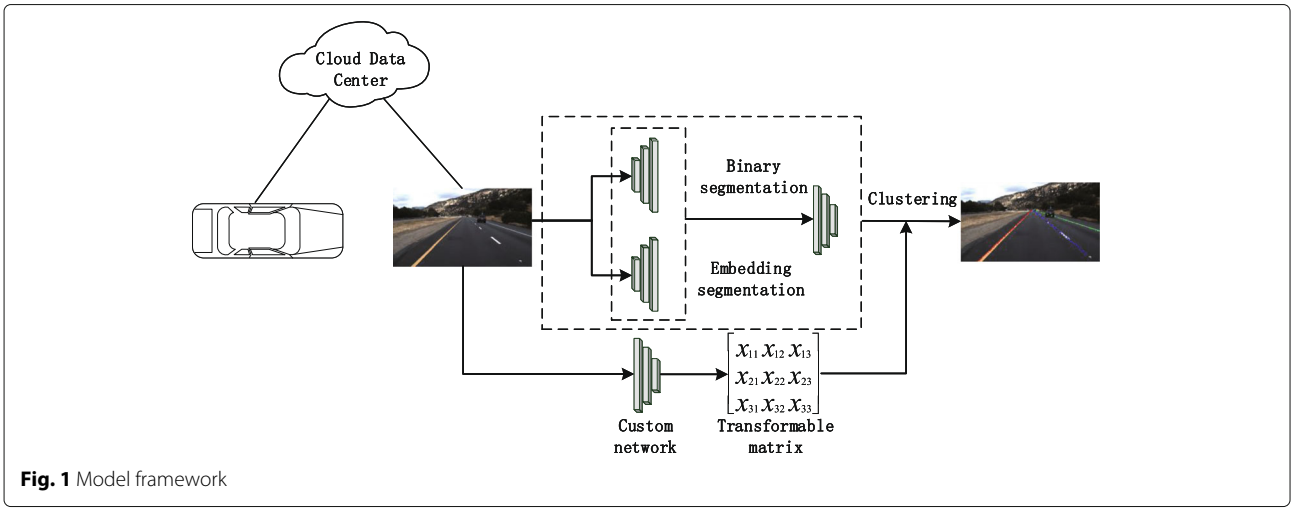

Fig. 1 Model framework

Lane instance embedding segmentation

The lane loss embedding branch function of the dual branch network is used to output the pixel embeddings of each lane, so that the pixel embeddings belonging to the same lane are clustered together, and the pixel embeddings belonging to different lanes have the largest distance. In the way, the pixel embeddings of the same lane will come together to form a unique cluster for each lane and this is achieved by introducing two terms, one is the variance term (L_{var}), which applies a pulling force to each embedding and tends to be closer to the average embedding of the lane. The other is the distance term (L_{dist}), which clusters the centers push away from each other. The pull force is active only when the distance from the embedding point to the cluster center is greater than δ_v. The thrust force is active only when the distance from the embedding point to the cluster center is less than δ_d.

$$\begin{cases} L_{var} = \frac{1}{C}\sum_{c=1}^{C}\frac{1}{N_C}\sum_{i=1}^{N_C}[\,\mu_c - \mu_i - \delta_v]_+^2 \\ L_{sum} = \sum_{CA,CB=1}^{C}[\,\delta_d - \mu_{CA} - \mu_{CB}]_+^2 \\ L_{dist} = \frac{1}{C(C-1)}L_{sum} \end{cases} \tag{1}$$

In (1), C is the number of lane lines, N_C is the number of elements in C, μ_i is the lane pixel embedding, μ_c is the average embedding of C, "$\parallel \mu_c - \mu_i \parallel$" represents the average embedding and pixel embedding distance, and CA and CB represent two lane line, $\mu_{CA} - \mu_{CB}$ represents the average embedding distance between CA lane line and CB lane line, $[\,\delta_d - \parallel \mu_{CA} - \mu_{CB} \parallel -\delta_v]_+^2$ means 0 and $\delta_d - \parallel \mu_{CA} - \mu_{CB} \parallel$ has the largest positive number value, $[\,\mu_c - \mu_i - \delta_v]_+^2$ represents the maximum value of a positive number between 0 and $\mu_c - \mu_i - \delta_v$, δ_v represents the threshold of the variance term, δ_d represents the threshold of the distance term, and the total loss $L = L_{var} + L_{dist}$. After the network converges, the lane pixel embeddings will be clustered together so that the distance between each cluster is greater than δ_d and the radius of each cluster is less than δ_v.

In Fig. 2a and b are the lane images collected, and the images generated by the example embedding segmentation are (c) and (d), respectively.

Clustering

For a large number of unlabeled data sets, the data set is divided according to the inherent similarity of the data, and the data set is divided into different categories with inherent differences. The similarity between the data in the same category is large and the similarity between the data in different categories is small. In order to improve the robustness of lane line detection in different environments, it is necessary to perform cluster analysis on the lane line pixels, embed the pixels of the same lane line together, and distinguish the pixels of different lane lines. In the future, clustering can ensure the recognition accuracy under the conditions of insufficient lighting conditions and poor traffic conditions.

In this paper, we complete clustering by iteration, combining the clustering algorithm with the loss function. In (1), by setting $6\delta_v < \delta_d$, we take a random lane with a radius of $2\delta_v$ and its surrounding threshold to select the ones that belong to the same lane all embedded. Repeat the above operation until all lanes are embedded and assigned to one lane. In order to avoid choosing the distance from the outlier to the threshold, we first use the

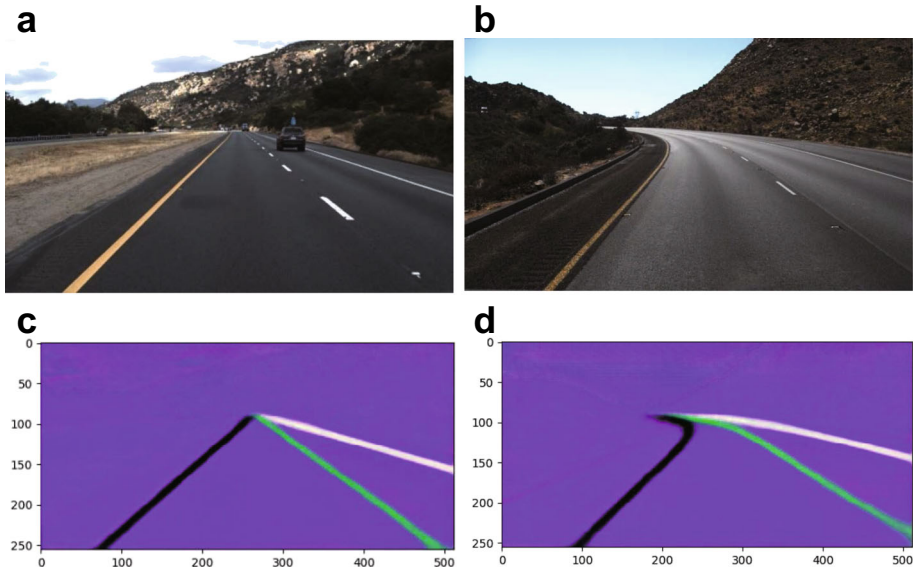

Fig. 2 Lane instance embedding segmentation graph

mean shift to approach the center of the cluster, and then sets the threshold.

Pixel fitting

The output of the two-branch network is a set of pixels for each lane line. It is not ideal to fit a polynomial with these pixels in the input image space, because a higher-order polynomial is needed to process the curve lane. Several commonly used fitting methods are RANSAC algorithm, Bezier curve fitting, curve fitting based on spline interpolation, and polynomial fitting based on least squares. In this paper, the image is first transformed by inverse perspective and projected into the "bird's eye view", where the lanes are parallel to each other, so the curve lane can be fitted by a second to third order polynomial. However, in these cases, the fixed transformation matrix H is calculated only once and applied to all images. This results in errors in the case of ground-level changes, where the vanishing point projected to infinity moves up or down as shown. To solve this problem, this paper trains a neural network with a custom loss function. The network is optimized end-to-end to predict the parameters of the perspective transformation matrix H. In the perspective transformation matrix H, the transformed lane points can be optimally fitted with second or third order polynomials. The prediction is based on the input image, allowing the network to adjust the projection parameters when the ground plane changes, so that the lane line fit is still correct. In (2), the transformation matrix H has 6 degrees of freedom, of which six variables a-f represent 6 degrees of freedom parameters. The purpose of placing 0 is to enforce the constraint and make the horizontal line remains horizontal under the transformation.

$$H = \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & f & 1 \end{bmatrix} \tag{2}$$

The network architecture of the custom function network remains small in size and consists of a continuous block of 3×3 convolutional layers, Batchnorm, and ReLU activation functions, and the largest pooling layer to reduce the network size, and finally add 2 fully connected layers. Before a pixel fits a lane line, a custom function network output transformable matrix is used to transform the lane pixels. The inverse perspective transformation is performed on the road image to a new top view to generate pixels in the top view lane after the inverse perspective transformation and the inverse perspective transformation formula is shown in (3).

$$P' = HP \tag{3}$$

In (3), $P'(P'_i = [X'_i, Y'_i, 1]^T \in P', i = 1, 2, \ldots)$ represents the transformed lane line pixels after inverse perspective transformation, $P(P_i = [X_i, Y_i, 1]^T \in P, i = 1, 2, \ldots)$ represents the pixels of real lane pixels, X_i, Y_i represents the horizontal and vertical coordinates of pixels of real lanes, X'_i, Y'_i represents the horizontal and vertical coordinates of lane pixel points after inverse perspective transformation. In order to obtain the position of the lane x at a given y position, the real lane pixel extraction point $P_i = [-, y_i, 1]^T$ is transformed into a transformed lane pixel point $P'_i = [-, y'_i, 1]^T$ after perspective transformation , predict lane pixel points and coordinates $P'^*_i = (x'^*_i, y'_i)$ at each y'_i position. A third-order polynomial is used to fit the predicted lane pixel point P'^*_i by the least square method. Equations (4) and (5) are third-order polynomials and least squares formulas, respectively.

$$f(y') = \alpha y'^2 + \beta y' + \gamma \tag{4}$$

$$w = \left(Y^T Y\right)^{-1} Y^T x' \tag{5}$$

In (4) and (5), α, β, and γ are the given optimal parameters, T is the transpose of the matrix, $w = [\alpha, \beta, \gamma]^T$is the least square method to represent the letter, $x' = [x'_1, x'_2, x'_3, \cdots x'_n]^T$represents the lane coordinate point P' abscissa combination, $y' = [y'_1, y'_2, y'_3, \cdots y'_n]^T$ represents the transformable parameter matrix transform lane pixel point P' ordinate combination, $Y = \begin{bmatrix} y'^2_1 & y'_1 & 1 \\ \cdots & \cdots & \cdots \\ y'^2_n & y'_1 & 1 \end{bmatrix}$ is the required matrix in the least squares formula. Fitting the predicted lane pixels $P'^*_i = [x'^*_i, y'_i, 1]^T$perspective transformation to the road lane image to obtain lane pixels P^*_i at different lane positions. The cluster loss function is used to segment the lane line instance into a pixel map, and the predicted pixels of the fitted lane are projected into the road lane image through perspective transformation to obtain the final detection result image, and the perspective transformation formula of the cluster loss function is shown in (6).

$$P^*_i = H^{-1} P'^*_i \tag{6}$$

In (6), $P^*_i = [x^*_i, y_i, 1]^T$ is a lane pixel projected into the input road lane image, $P'^*_i = [x'^*_i, y'_i, 1]^T$is the predicted lane pixel point, and H^{-1}represents the perspective transformation matrix. In order to train the output of the custom function network H, which is the most suitable transformation matrix H for pixel fitting polynomials, the following loss function is constructed. Since lane fitting is done by using the least squares method, the loss function is differentiable and the loss function of the custom function network is shown in (7).

$$Loss = \frac{1}{N} \sum_{i=1}^{N} (x^*_i - x_i)^2 \tag{7}$$

Experimental evaluation

Data set

This article uses the Tusimple dataset as a training and testing source, which includes lane line data at different times during the day, and also includes situations with 2 lanes, 3 lanes and 4 lanes.

For the lane segmentation task, the final segmentation result needs an evaluation index to measure the performance of the test. We choose F-Measure to measure the model's ability to predict the lane line because the F-Measure evaluation index can comprehensively consider the two measures of precision and recall in the experiment. All pixels of the lane line are defined as positive samples, which are denoted by P and background. A pixel is a negative sample, denoted as N. It is determined whether the pixel prediction is correct by judging whether the predicted image result and each pixel of the real label are equal. Formulas for evaluating image prediction results using the F-Measure index are shown in (8) and (9).

$$Precision = \frac{TP}{TP + FP}, Recall = \frac{TP}{TP + FN} \tag{8}$$

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{9}$$

Precision reflects the accuracy of the model and is expressed as the ratio of the number of correct predictions to the total number of predictions. Recall reflects the recall rate of the model and is a measure of coverage, expressed as the ratio of the number of targets detected in the data to the total number of targets that actually exist. TP represents the number of positive instances detected correctly, FP represents the number of instances detected as positive cases, FN represents the number of instances detected as negative cases and β is an adjustable parameter which role is to highlight the proportion of Recall and we set it to 0.8, which helps reduce the damage caused by misrecognizing lane areas.

Experimental results

The experimental environment of this article is Ubuntu 16.04 operating system, using NVIDIA GEFORCE 920M graphics processor to implement all code based on Tensorflow deep learning framework. During the training process, the input image is regularized and randomly cropped and randomly rotated. The size of the obtained pictures is uniformly set to 256 × 512, and the batch size is set to 6 so that 6 pictures can input at the same time for training. The gradient descent algorithm optimizes the training of network parameters. The optimizer has a momentum parameter of 0.9, a weight decay of 1×10^{-4}, and a learning rate update strategy of exponential decay:

$$lr = lr_{init} \times \left(1 - \frac{iter}{N}\right)^{power} \tag{10}$$

In (10), lr_{init} is the initial learning rate, $lr_{init} = 0.0005$, N is the total number of iterations for training, set to 80010, $iter$ is the current number of iterations, and $power$ is the attenuation coefficient, set to 0.9. We select some images from the Tusimple data set for testing. By comparing with the traditional lane detection algorithms and the currently used deep learning algorithms, it is found that the lane detection algorithm used in this paper has excellent performance in scenarios such as inadequate lighting, shadow occlusion, missing lane lines and curved lanes.

By using the model in this paper to detect under different road complex environments, we can get the following detection results with Table 1. We found that other models can achieve correspondingly good results when dealing with lanes in some scenes, but it is difficult to take into account all the scenes, while our proposed method is superior to other models in various scenarios, which is a benefit for lane recognition technology.

At the same time, we obtained the time information of the relevant model when processing the lane images. During the lane instance segmentation and clustering process, each frame image took 16.6ms and the FPS reached 60.2. During the lane fitting process, each frame took 2.4ms and the FPS reached 416.7. The total test efficiency reached 52.6 frames per second and achieved a good processing rate.

In Fig. 3a and b are the lane images collected in a poorly lit environment, (c) and (d) are the lane detection results. It can be seen that the lane line detection model used in this paper can still achieve better detection results under poor lighting conditions.

In Fig. 4a and b are the lane images collected in the road line degradation scene, (c) and (d) are the corresponding

Table 1 Comparison of test results of different models

Models	Insufficient light	Shadow occlusion	Missing lane line	Curve lane line	Normal road
VGG-FCN	0.9369	0.9311	0.8837	0.9259	0.9308
SCNN	0.9612	0.9713	0.9412	0.9315	0.9671
U-Net	0.9510	0.9613	0.9126	0.9168	0.9647
Tradition	0.9213	0.9122	0.8635	0.8874	0.9171
Ours	0.9662	0.9785	0.9588	0.9410	0.9757

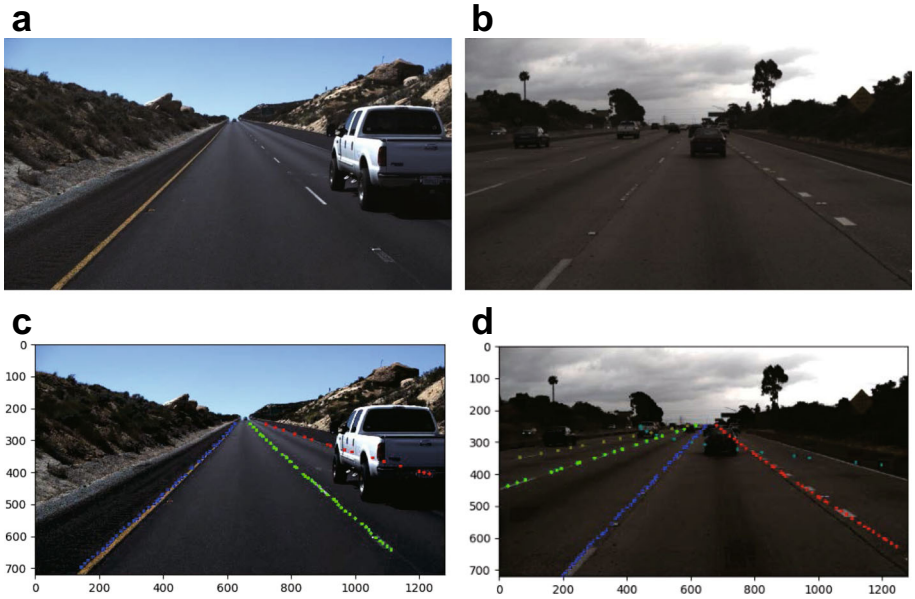

Fig. 3 Lane detection under poor lighting conditions

detection results. It can be seen that even in some cases the lane line has been degraded or blurred, the lane line can still be accurately detected under the deep training lane network model, which proves that the network model used in this paper has better robustness.

In Fig. 5a and b are lane images collected in a scene with shadows and obstacles, and (c) and (d) are corresponding detection results. It can be found that the lane detection model used in this paper can still detect lane lines well in the case of obstacles and shadow, and has strong robustness.

Conclusion

In this paper, the lane detection algorithm is carefully studied. The lane detection method based on traditional image processing and the lane detection method based on deep learning are analyzed and compared to solve the problem of lane detection under complex conditions

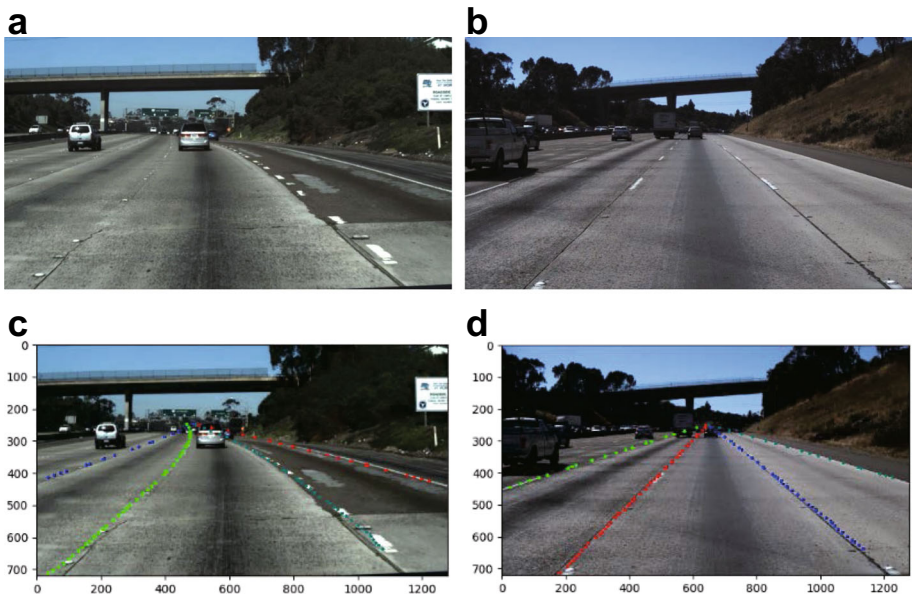

Fig. 4 Lane line detection under lane line edge degradation

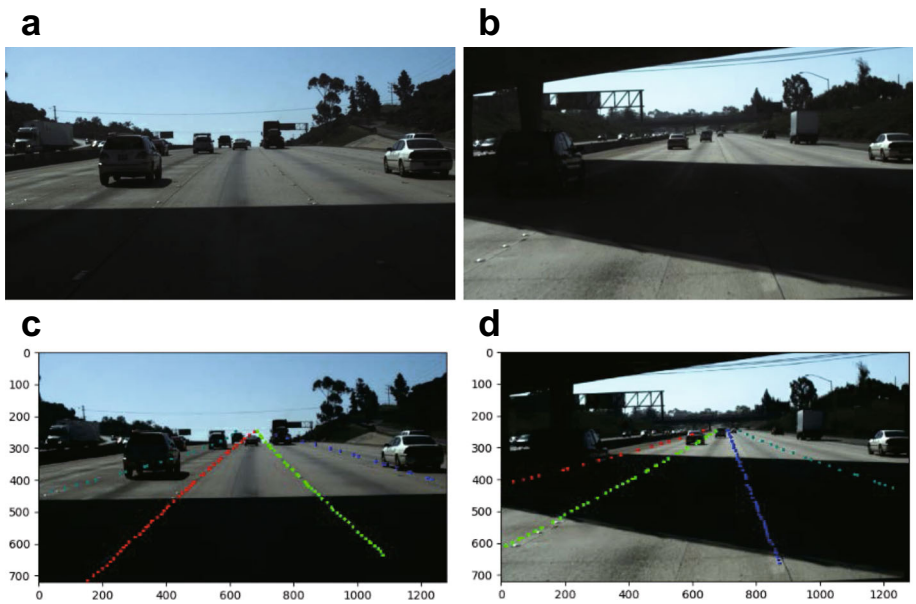

Fig. 5 Detection of lane lines under shadows and obstacles

such as shadows and obstacles. In order to improve the efficiency of central computing, a two-branch training network and a customized training network based on semantic segmentation are proposed, and the combination of vehicle edge calculation and actual engineering can improve the effect of lane line detection network.

In the inverse perspective transformation, the use of a fixed transformation matrix will cause errors when ground changes, which will cause the vanishing point projected to infinity to move up or down. We trained a neural network with a custom loss function that can dynamically predict the parameter values of the transformable matrix. The transformed lane points are fitted with second or third-order polynomials. The prediction is based on the input image, allowing the network to adjust the projection parameters when the ground plane changes, making the model excellent stickiness. The final tests show that the model in our paper has better performance in scenarios such as insufficient lighting and lane line degradation.

Authors' contributions

Wei Wang, Hui Lin, Junshu Wang conceived and designed the study. Wei Wang, Hui Lin and Junshu Wang performed the simulations. Wei Wang, Hui Lin and Junshu Wang wrote the paper. All authors reviewed and edited the manuscript. All authors read and approved the final manuscript.

Funding

This research is supported by National Natural Science Foundation of China under Grant No. 41671055, Jiangsu Natural Science Foundation of China under Grant No. BK20171037, the Program of Natural Science Research of Jiangsu colleges and universities under Grant No.17KJB170010.

Availability of data and materials

The raw data required to reproduce these findings cannot be shared at this time as the data also forms part of an ongoing study.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Neusoft Corporation, Shenyang, China. [2]Nanjing University of Information Science and Technology, China No.219 Ningliu Road, 210044 Nanjing, China. [3]Key Laboratory for Virtual Geographic Environment, Ministry of Education, Nanjing Normal University, Nanjing, China. [4]Jiangsu Center for Collaborative Innovation in Geographical Information Resource Development and Application, Nanjing, China.

Received: 4 March 2020 Accepted: 22 April 2020
Published online: 19 May 2020

References

1. Smuda P, Schweiger R, Neumann H, Ritter W (2006) Multiple cue data fusion with particle filters for road course detection in vision systems. In: 2006 IEEE Intelligent Vehicles Symposium. IEEE. pp 400–405. https://doi.org/10.1109/ivs.2006.1689661
2. Aly M (2008) Real time detection of lane markers in urban streets. In: 2008 IEEE Intelligent Vehicles Symposium. IEEE. pp 7–12. https://doi.org/10.1109/ivs.2008.4621152
3. Teng W, Wang Y, Yu B, Liu J (2020) Icciu: A new real-time lane-level positioning algorithm. IEEE Access. https://doi.org/10.1109/access.2020.2970503
4. Bertozzi M, Broggi A (1996) Real-time lane and obstacle detection on the gold system. In: Proceedings of Conference on Intelligent Vehicles. IEEE. pp 213–218. https://doi.org/10.1109/ivs.1996.566380
5. Xue C, Lin C, Hu J (2019) Scalability analysis of request scheduling in cloud computing. Tsinghua Sci Technol 24(3):249–261
6. Shen D, Luo J, Dong F, Zhang J (2019) Virtco: joint coflow scheduling and virtual machine placement in cloud data centers. Tsinghua Sci Technol 24(5):630–644
7. Xu X, He C, Xu Z, Qi L, Wan S, Bhuiyan M (2019) Joint optimization of offloading utility and privacy for edge computing enabled IoT. IEEE Internet of Things J. https://doi.org/10.1109/jiot.2019.2944007
8. Xu X, Zhang X, Khan M, Dou W, Xue S, Yu S (2017) A balanced virtual machine scheduling method for energy-performance trade-offs in cyber-physical cloud systems. Futur Gener Comput Syst
9. Xu X, Mo R, Dai F, Lin W, Wan S, Dou W (2019) Dynamic resource provisioning with fault tolerance for data-intensive meteorological

workflows in cloud. IEEE Trans Ind Inform. https://doi.org/10.1109/tii.2019.2959258
10. Yang S, Wu J, Shan Y, Yu Y, Zhang S (2019) A novel vision-based framework for real-time lane detection and tracking. Technical report. SAE Technical Paper. https://doi.org/10.4271/2019-01-0690
11. Xu X, Zhang X, Gao H, Xue Y, Qi L, Dou W (2020) Become: Blockchain-enabled computation offloading for iot in mobile edge computing. IEEE Trans Ind Inform 16(6):4187–4195
12. Ganin AA, Mersky AC, Jin AS, Kitsak M, Keisler JM, Linkov I (2019) Resilience in intelligent transportation systems (its). Transp Res Part C Emerg Technol 100:318–329
13. Kuo C, Lu Y, Yang S (2019) On the image sensor processing for lane detection and control in vehicle lane keeping systems. Sensors 19(7):1665
14. Xu Y, Qi L, Dou W, Yu J (2017) Privacy-preserving and scalable service recommendation based on simhash in a distributed cloud environment. Complexity 2017. https://doi.org/10.1155/2017/3437854
15. Xu X, Liu Q, Luo Y, Peng K, Zhang X, Meng S, Qi L (2019) A computation offloading method over big data for iot-enabled cloud-edge computing. Futur Gener Comput Syst 95:522–533
16. Chi X, Yan C, Wang H, Rafique W, Qi L Amplified locality-sensitive hashing-based recommender systems with privacy protection. Concurr Comput Pract Experience:5681. https://doi.org/10.1002/cpe.5681
17. Xu X, Chen Y, Zhang X, Liu Q, Liu X, Qi L (2019) A blockchain-based computation offloading method for edge computing in 5g networks. Softw Pract Experience. https://doi.org/10.1002/spe.2749
18. Xu X, Cao H, Geng Q, Liu X, Dai F, Wang C (2020) Dynamic resource provisioning for workflow scheduling under uncertainty in edge computing environment. Concurr Comput Pract Experience:5674. https://doi.org/10.1002/cpe.5674
19. Son Y, Lee ES, Kum D (2019) Robust multi-lane detection and tracking using adaptive threshold and lane classification. Mach Vis Appl 30(1):111–124
20. Wang Y, Teoh EK, Shen D (2004) Lane detection and tracking using b-snake. Image Vis Comput 22(4):269–280
21. Tan H, Zhou Y, Zhu Y, Yao D, Li K (2014) A novel curve lane detection based on improved river flow and ransa. In: 17th International IEEE Conference on Intelligent Transportation Systems (ITSC). IEEE. pp 133–138. https://doi.org/10.1109/itsc.2014.6957679
22. Shin B-S, Tao J, Klette R (2015) A superparticle filter for lane detection. Pattern Recogn 48(11):3333–3345
23. Yoo H, Yang U, Sohn K (2013) Gradient-enhancing conversion for illumination-robust lane detection. IEEE Trans Intell Transp Syst 14(3):1083–1094
24. Son J, Yoo H, Kim S, Sohn K (2015) Real-time illumination invariant lane detection for lane departure warning system. Expert Syst Appl 42(4):1816–1824
25. Jung S, Youn J, Sull S (2015) Efficient lane detection based on spatiotemporal images. IEEE Trans Intell Transp Syst 17(1):289–295
26. Niu J, Lu J, Xu M, Lv P, Zhao X (2016) Robust lane detection using two-stage feature extraction with curve fitting. Pattern Recogn 59:225–233
27. De Brabandere B, Van Gansbeke W, Neven D, Proesmans M, Van Gool L (2019) End-to-end lane detection through differentiable least-squares fitting. arXiv preprint. arXiv:1902.00293
28. Liu L, Chen X, Lu Z, Wang L, Wen X (2019) Mobile-edge computing framework with data compression for wireless network in energy internet. Tsinghua Sci Technol 24(3):271–280
29. Qi L, Zhang X, Dou W, Ni Q (2017) A distributed locality-sensitive hashing-based approach for cloud service recommendation from multi-source data. IEEE J Sel Areas Commun 35(11):2616–2624
30. Qi L, Zhang X, Dou W, Hu C, Yang C, Chen J (2018) A two-stage locality-sensitive hashing based approach for privacy-preserving mobile service recommendation in cross-platform edge environment. Futur Gener Comput Syst 88:636–643
31. John V, Liu Z, Guo C, Mita S, Kidono K (2015) Real-time lane estimation using deep features and extra trees regression. In: Image Video Technol. Springer. pp 721–733. https://doi.org/10.1007/978-3-319-29451-3_57
32. Kim J, Park C (2017) End-to-end ego lane estimation based on sequential transfer learning for self-driving cars. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp 30–38. https://doi.org/10.1109/cvprw.2017.158
33. Pan X, Shi J, Luo P, Wang X, Tang X (2018) Spatial as deep: Spatial cnn for traffic scene understanding. In: Thirty-Second AAAI Conference on Artificial Intelligence
34. Zhang Z, Schwing AG, Fidler S, Urtasun R (2015) Monocular object instance segmentation and depth ordering with cnns. In: Proceedings of the IEEE International Conference on Computer Vision. pp 2614–2622. https://doi.org/10.1109/iccv.2015.300
35. Xu X, Fu S, Yuan Y, Luo Y, Qi L, Lin W, Dou W (2019) Multiobjective computation offloading for workflow management in cloudlet-based mobile cloud using nsga-ii. Comput Intell 35(3):476–495
36. Xu X, Liu X, Xu Z, Dai F, Zhang X, Qi L (2019) Trust-oriented iot service placement for smart cities in edge computing. IEEE Internet Things J. https://doi.org/10.1109/jiot.2019.2959124
37. Aminuddin NS (2020) A new approach to highway lane detection by using hough transform technique. J Inf Commun Technol 16(2):244–260
38. Wang J, Kong B, Mei T, Wei H (2019) Lane detection algorithm based on temporal–spatial information matching and fusion. CAAI Trans Intell Technol 2(4):154–165
39. Xu X, Liu Q, Zhang X, Zhang J, Qi L, Dou W (2019) A blockchain-powered crowdsourcing method with privacy preservation in mobile environment. IEEE Trans Comput Soc Syst 6(6):1407–1419
40. Nguyen TNA, Phung SL, Bouzerdoum A (2020) Hybrid deep learning-gaussian process network for pedestrian lane detection in unstructured scenes. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/tnnls.2020.2966246
41. Barrera A, Guindel C, Beltrán J, García F (2020) Birdnet+: End-to-end 3d object detection in lidar bird's eye view. arXiv preprint. arXiv:2003.04188

Publisher's Note