



Lane Detection Based on Instance Segmentation of BiSeNet V2 Backbone Network

Sun Yang, Li Yunpeng & Liu Yu

To cite this article: Sun Yang, Li Yunpeng & Liu Yu (2022) Lane Detection Based on Instance Segmentation of BiSeNet V2 Backbone Network, Applied Artificial Intelligence, 36:1, 2085321, DOI: [10.1080/08839514.2022.2085321](https://doi.org/10.1080/08839514.2022.2085321)

To link to this article: <https://doi.org/10.1080/08839514.2022.2085321>



© 2022 The Author(s). Published with license by Taylor & Francis Group, LLC.



Published online: 15 Jun 2022.



Submit your article to this journal [↗](#)



Article views: 1510



View related articles [↗](#)



View Crossmark data [↗](#)

Lane Detection Based on Instance Segmentation of BiSeNet V2 Backbone Network

Sun Yang^a, Li Yunpeng^b, and Liu Yu^b

^aSchool of Mechanical and Equipment Engineering, Hebei University of Engineering, and the Handan Key Laboratory of Intelligent Vehicles, Handan, Hebei, China; ^bSchool of Mechanical and Equipment Engineering, Hebei University of Engineering, Handan, Hebei, China

ABSTRACT

Most lane line detection algorithms still have room for improvement in detection accuracy, speed, and robustness. Meanwhile, these algorithms only test the performance indicators through the test set of the open-source dataset rather than deploying them on actual vehicles and evaluating the performance indicators through road scenarios. Therefore, this paper proposes a lane detection algorithm based on instance segmentation. Firstly, a dual-branch neural network model for lane line image segmentation was designed based on BiSeNet V2. Then the discrete lane line feature points are operated through the clustering model. The corresponding feature points are selected for fitting by combining straight lines and curves to obtain the appropriate fitting parameter equation for the specific visual field area. Finally, the model is trained and verified based on the TuSimple dataset. The algorithm has a noticeable performance improvement under the two evaluation indicators of mIoU and FPS. Meanwhile, the model is integrated into the ROS task platform for intelligent vehicles. The results show that the algorithm's accuracy and detection speed are increased to about 3.9 and 2.9 times, respectively, that of the improved probabilistic Hough transform algorithm under the two evaluation indicators of lateral distance and the detection time of each image frame.

ARTICLE HISTORY

Received 31 March 2022



Revised 27 May 2022

Accepted 30 May 2022

Introduction

Solving the problem of environmental perception is one of the essential prerequisites for the realization of autonomous driving of intelligent vehicles, and research on lane line detection plays a crucial role in environmental perception.

Traditional lane line detection algorithms are easily affected by environmental conditions such as rain, snow, fog, and night. The parameters in the algorithm need to be manually adjusted based on experience to meet different driving scenarios. In contrast, lane line detection based on deep learning

CONTACT Liu Yu  2261180009@qq.com  School of Mechanical and Equipment Engineering, Hebei University of Engineering, Xiangyuan Jinwan, Changning Avenue, High-tech Industrial Development Zone, Shushan, Hebei, Anhui Province, 230031, China

© 2022 The Author(s). Published with license by Taylor & Francis Group, LLC.

This is an Open Access article distributed under the terms of the Creative Commons Attribution-NonCommercial License (<http://creativecommons.org/licenses/by-nc/4.0/>), which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

algorithms is usually more complex and generally consists of an input, a hidden, and an output layer. If the number of network layers is different, the lane line information extracted by the algorithm from the image is also different. For the underlying network layers, the extracted features are usually low-dimensional information. However, when the number of layers is progressive, the neural network model learns high-dimensional information such as the color, outline, or texture features of the lane lines from the image.

Two image segmentation techniques are based on deep learning, namely semantic and instance segmentation (Chang et al., 2019; Chen et al. 2018b; Hou et al. 2019; Lee et al. 2017; Liu et al. 2020; Mohsen et al. 2018; Pan et al. 2017; Shao-Yuan et al. 2019). In recent years, semantic segmentation networks have developed rapidly, and some researchers have used them for lane line detection. Li et al. (2019) proposed an improved encoder-decoder network with an instance batch normalization network and an attention mechanism. Pixel-level classification of scene labels using captured content image structure addresses the shortcomings of batch normalization for texture capture in end-to-end segmentation. IBN layers are applied to replace standard BN layers to use visual and appearance invariance of instance normalization. Then, an attention mechanism is added to the network to force the network to focus on the lane area. Liu, Deng, and Yang (2017) proposed a dilated feature pyramid network with feature aggregation, called DFFA, in which feature aggregation was used to combine multi-level features enhanced by dilated convolution operations and FPN under the ResNet framework.

Other research teams are using instance segmentation to detect lane lines. Researchers such as Neven et al. (2018) have established a neural network model for the problem of lane line detection as image instance segmentation, which can be trained end-to-end to detect each lane line instance. This approach can recognize and handle multi-lane situations while coping with lane changes. The algorithm does not use a fixed “bird’s-eye view” transformation but learns perspective transformation based on images, so that lane instances can be parameterized after lane line segmentation to complete lane line fitting. Wang et al. (2020) improved the traditional lane detection method and established a dual model using the currently popular convolutional neural network based on instance segmentation. In image acquisition and processing, the distributed computing architecture provided by edge cloud computing is used to improve data processing efficiency. In the case of slope changes, the lane fitting process generates a variable matrix to achieve adequate detection and improve the real-time performance of lane detection.

Most of the deep learning-based lane line detection algorithms mentioned above have not been tested on actual vehicles and are not practical. At the same time, their detection algorithm still has room for improvement in

detection accuracy, detection speed, and robustness. These inspired me to develop a better deep learning-based lane detection algorithm and apply it to actual vehicles to test its various performance indicators. Our contributions are summarized as follows:

- We propose a dual-branch neural network model based on BiSeNet V2 for lane line image segmentation.
- We apply our proposed approach to the existing vehicle platform using our proposed unified accuracy index of lateral offset and real-time index of detection time per frame. And we demonstrate that it has corresponding practical implications.

Method

Data Collection and Annotation

The training set used in this study contains 100,000 images, the validation set contains 9,754 images, and the test set contains 35,680 images. When manually labeling the data, if an obstacle occludes the lane line, the label passes through the obstacle and then labels the dataset. This labeling method enables the training model to have specific generalization ability to obstacles. When making the dataset, use labelme software to label the lane lines. This annotation tool can quickly generate the annotation file .json of the dataset. Considering the data set format required by the subsequent neural network detection model, the annotation file .json is converted into the TuSimple data set (TuSimple 2018) format as the neural network input. An example of labelme annotation and an image after data transformation is shown in [Figure 1](#).

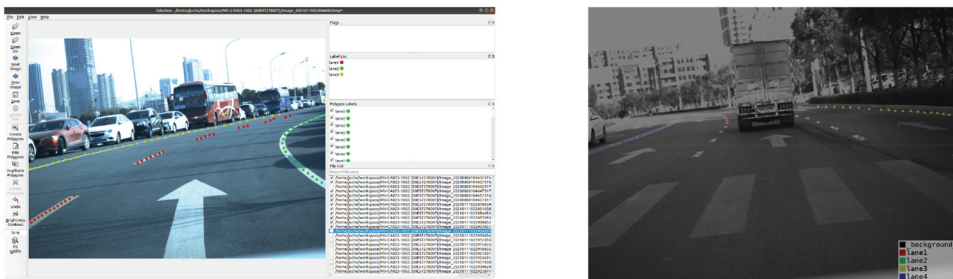


Figure 1. Labelme annotation and image data transformation.

Neural Network

This study designs a multi-task branching network to instance segmentation of lane lines. The network mainly includes lane line segmentation and lane line clustering. The lane line segmentation branch is mainly used to obtain binary segmentation results of lane line pixels. The lane line clustering branch is mainly used to determine which lane line category is the pixel after binary segmentation. These different classes are clustered by treating different lane lines as different classes. The problem of lane line detection is decomposed into the above two tasks, which improves the running speed of the network and realizes high-precision detection of lane lines.

The most significant advantage of the instance segmentation model for lane line detection using the deep learning training dataset is that it can solve the problem of detecting an arbitrary number of lanes to accommodate scenarios where vehicles change lanes. Instance segmentation of lane lines usually consists of two parts: lane line semantic segmentation and lane line clustering. The former is used to obtain the binary image of the lane line; the latter is used to determine whether the pixels belong to the same lane line (Libiao and Qilong 2019). The above two parts will share the previous encoding process to improve the accuracy and calculation speed of lane line segmentation when performing instance segmentation.

The lane line segmentation is to obtain the segment of the lane line pixels, which can be expressed as to whether the image pixel belongs to the lane line. This study uses the standard cross-entropy loss function (Farahnak-Ghazani and Baghshah 2016) for model training. However, the ratio of the number of pixels belonging to the lane line and the background to all the pixels in the image is highly unbalanced, so the weights δ are set for the lane line and background pixels, respectively. The formula L of loss function for lane line segmentation branch is as follows.

$$\delta_1 = \frac{1}{\ln(1.02 + n_1/m)} \quad (1)$$

$$\delta_2 = \frac{1}{\ln(1.02 + n_2/m)} \quad (2)$$

$$L = \begin{cases} -\delta_1 \ln(p^*) & p = 1 \\ -\delta_2 \ln(1 - p^*) & p = 0 \end{cases} \quad (3)$$

Among them, δ_1, δ_2 is the weight of the pixels belonging to the lane line and the background; p^* is the probability that the detected pixel is a lane line pixel. n_1, n_2 is the number of pixels belonging to the lane line and the background. m is the total number of pixels. When the pixel belongs to the lane line pixel, $p = 1$, otherwise $p = 0$.

The high accuracy of lane detection methods depends on their backbone networks. However, real-time semantic segmentation applications also demand an efficient inference speed. Facing this demand, based on both backbone networks, existing methods mainly employ two approaches to accelerate the model: (i) Restricting Input. Smaller input resolution results in less computation cost with the same network architecture. Many algorithms attempt to restrict the input size to reduce the whole computation complexity to achieve real-time inference speed; (ii) Channel Pruning. It is a straightforward acceleration method, significantly when pruning channels in the early stages to boost inference speed. Although both methods can improve the inference speed to some extent, they sacrifice the low-level details and spatial capacity, leading to a dramatic accuracy decrease. Therefore, to achieve high efficiency and high accuracy simultaneously, it is challenging and of great importance to exploit a specific architecture for the real-time semantic segmentation task. We observe that both the low-level details and high-level semantics are crucial to the semantic segmentation task. In the general semantic segmentation task, the deep and wide networks simultaneously encode both kinds of information. However, we can treat spatial details and categorical semantics separately in the real-time semantic segmentation task to achieve the trade-off between accuracy and inference speed. To this end, we propose a two-pathway architecture, termed Bilateral Segmentation Network (Yu et al. 2021) (BiSeNet V2), for real-time semantic segmentation. One pathway is designed to capture the spatial details with wide channels and shallow layers, called the Detail Branch. In contrast, the other pathway is introduced to extract the categorical semantics with narrow channels and deep layers, called the Semantic Branch. The Semantic Branch requires a sizable receptive field to capture semantic context, while the Detail Branch can supply detailed information. Therefore, the Semantic Branch can be lightweight with fewer channels and a fast-downsampling strategy. Both types of feature representation are merged to construct a more substantial and comprehensive feature representation. This conceptual design leads to an efficient and effective architecture for real-time semantic segmentation, as illustrated in [Figure 2](#).

[Figure 3](#) shows the network model instantiation of the detail branch and semantic branch of BiSeNet V2, which converts the input TuSimple image 1280×720 to 512×256 pixel format for feature extraction. Since the encoding process has less semantic feature information, the lane line position is accurate. In contrast, the semantic feature information in the decoding process is relatively affluent, and the lane line position is relatively rough, so the corresponding encoded information is fused during decoding to improve the accuracy of lane line segmentation. [Figure 4](#) is an effect diagram after image segmentation, and a binary lane line image is obtained after image segmentation.

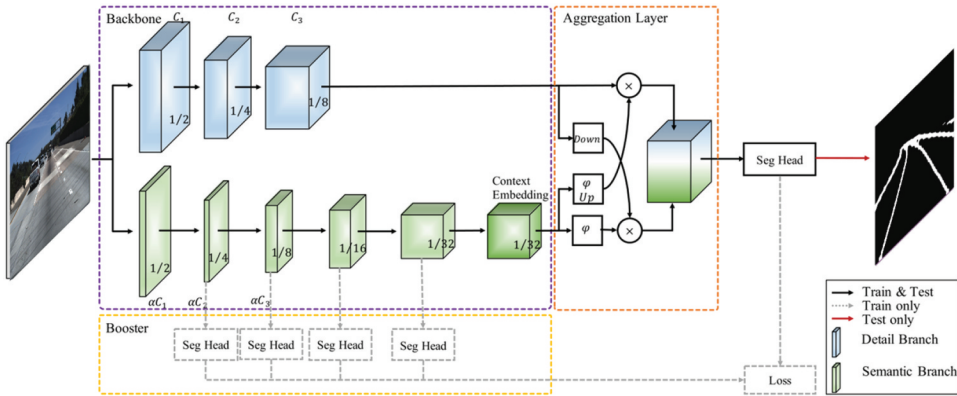


Figure 2. BiSeNet V2 network model architecture.

Stage	Encoder					Decoder						Output Size
	opr	k	c	s	r	opr	k	c	e	s	r	
Input												1920×1200
S ₁	Conv2d	3	64	2	1	Stem	3	16	-	4	1	521×256
	Conv2d	3	64	1	1							512×256
S ₂	Conv2d	3	64	2	1							256×128
	Conv2d	3	64	1	2							256×128
S ₃	Conv2d	3	128	2	1	GE	3	32	6	2	1	128×64
	Conv2d	3	128	1	2	GE	3	32	6	1	1	128×64
S ₄						GE	3	64	6	2	1	64×32
						GE	3	32	6	1	1	64×32
S ₅						GE	3	128	6	2	1	32×16
						GE	3	128	6	1	3	32×16
						CE	3	128	-	1	1	32×16

Figure 3. Detail and semantic branch instantiation.

Lane Cluster

The purpose of lane line clustering is to identify which pixels in an image belong to the same lane line. Through the vector values corresponding to all pixels in the output image of the encoder, it can be found that the vector distances between pixels of different lane lines are relatively far, while the vector distances between pixels of the same lane line are relatively close. Therefore, this feature is used to cluster pixels to identify pixels belonging to the same lane line. The clustering loss function consists of L1 and L2. The function of L1 is to cluster the pixels belonging to the same lane line to the center point according to the clustered pixel vector value. The role of L2 is to

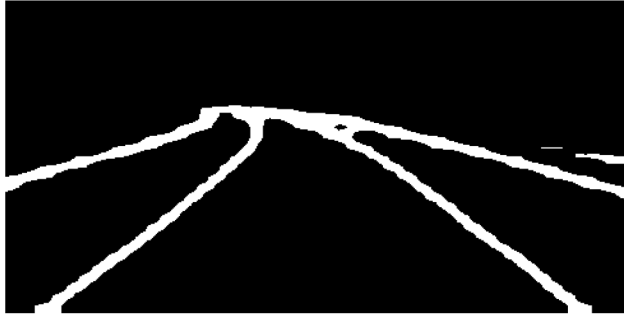


Figure 4. Image segmentation effect diagram.

keep pixels that do not belong to a specific lane line as far away as possible according to the clustered pixel vector values. There are $M(M-1)/2$ groups of different lane lines. The clustering loss function L (de Brabandere et al. 2017) is as follows.

$$L_1 = \frac{1}{M} \sum_1^M \frac{1}{M_c} \sum_1^{M_c} [|\mu_c - x_i| - 0.6]^2 \quad (4)$$

$$L_2 = \begin{cases} \frac{M(M-1)}{2} \sum_1^{M(M-1)/2} [6 - \mu_i - \mu_j]^2 & M \geq 2 \quad i \neq j \\ 0 & M = 1 \end{cases} \quad (5)$$

$$L = L_1 + L_2 \quad (6)$$

In the formula, M is the number of lane lines, M_c is the pixel number of the lane line, μ_c is the mean vector of the pixel clustering of the lane line, x_i is the clustering vector pixel point. μ_i and μ_j are the mean vectors of two different lane line pixel clusters (i, j) .

In the lane line clustering stage, the DBSCAN clustering algorithm is adopted to cluster each pixel belonging to the lane line. This method is suitable for high-density connected regions, dividing high-density regions into clusters and discovering clusters of arbitrary shapes from noisy data (Tran, Drab, and Daszykowski 2013). Figure 5 shows the change in loss during training. In this study, the center of clusters is the circle center, the radius is $0.3 m$, and the minimum number of points in the cluster is 180 for clustering until all lane line pixels are assigned to the corresponding lane lines.

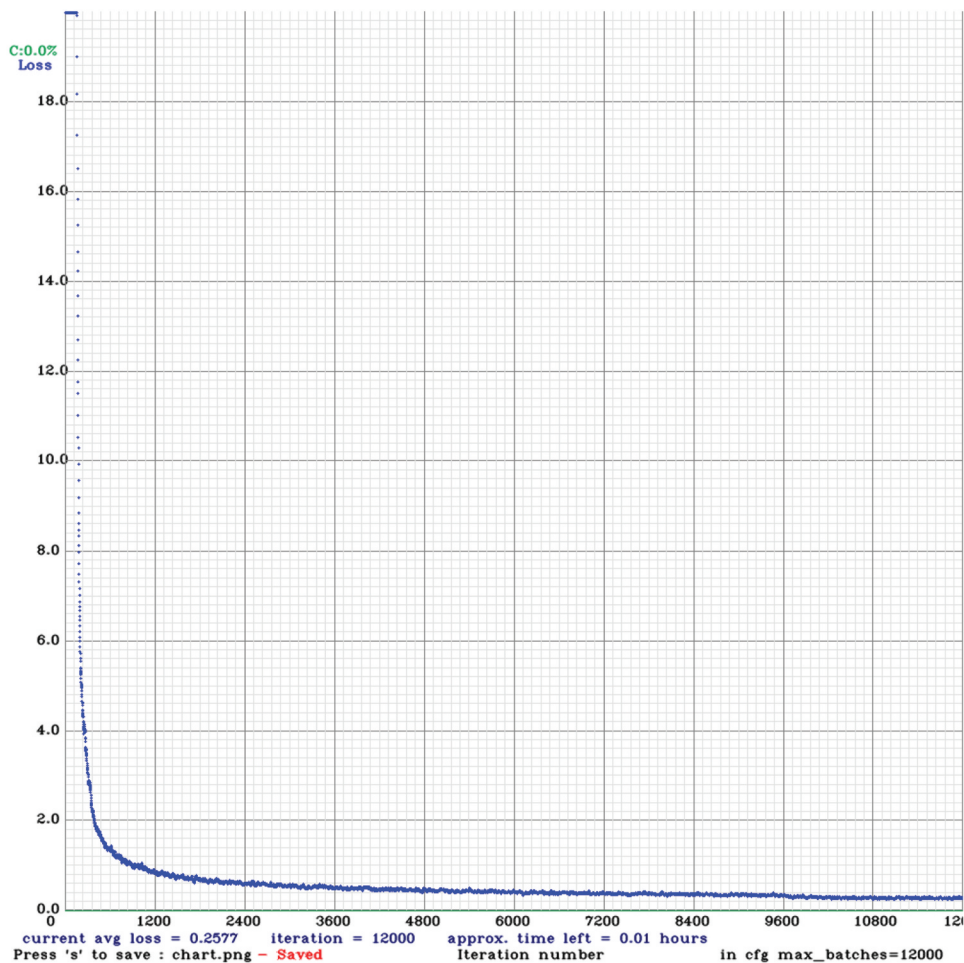


Figure 5. Change in loss during training.

Post-Processing

Before fitting the lane line, the driving area needs to be divided. Under normal circumstances, the most precise image that the onboard camera can obtain is about 100 m ahead of the current driving vehicle. In this study, the lane line image area collected by the vehicle camera is divided into three areas, A/B/C, as shown in Figure 6. Area A is the central part of the field of view and is set as a near-view area. The lane lines in this area can be assumed to be straight lines, and a simple fitting model can be selected for processing. Area B may have a curved part with a large radian, and the information of the lane line reflects the curvature of the lane line. It is set as a medium-view area. This area needs to be processed by a complex fitting model. Area C is the upper half of the image. This area does not contain lane marking information and is set as the far-field view area.



Figure 6. Lane line image area division.

According to the situation after the lane line image area is divided, when designing the fitting algorithm, it is only necessary to pay attention to the lane lines of the near-view and the medium-view area of the current image road surface area. Then, different fitting models are designed according to the linear characteristics of line segments in different regions to improve the accuracy of the fitting algorithm. Using straight-line fitting and cubic spline fitting algorithms to fit the lane lines of different fields of view in segments and finally generate the optimal parameter equation of lane line fitting.

Since the near-field area A is mainly composed of straight lines, a straight-line fitting model is used to process this area. Usually, the Hough transform or the least-squares method extracts or fits straight lines. By combining the above two algorithms, this study adopts a new fitting method that maximizes the advantages of Hough transform and least squares and avoids their disadvantages. First, the rough area of the lane line in the image is defined by the Hough transform algorithm, then the feature points in the area where the detected lane line is located are clustered, and finally, the parameters of the fitted line equation are determined by the improved least squares method.

For the case where the curvature of the lane line is slight, the lane line information obtained by the straight-line fitting model can meet the needs of lane line detection, but the straight-line fitting model cannot be applied to the lane line area with significant curvature. In this study, the straight-line fitting model is only used for the lane line fitting in the A area, and the curve model is designed for the B area with more complex linear features. Standard curve fitting models include Bayesian fitting (Denison et al. 1998), B-spline curve fitting (Zheng et al. 2012), and least-squares curve fitting (Kaibo, Ning, and Peishou 2015). When performing curve fitting at the junction of the straight

line and the curve of the lane line, the commonly used model does not have the adaptive ability. On this basis, this study uses a third-order linear equation to fit the model, and the specific formula is as follows.

$$f(x) = L_0 + L_1x + \frac{L_2}{2}x^2 + \frac{L_3}{6}x^3 \quad (7)$$

In this study, the cubic polynomial $f(x)$ is used to fit the clustered feature points, and L_0 , L_1 , L_2 , and L_3 represent the parameter information of the curve model. Among them, L_0 represents the yaw angle and heading, L_1 represents the slope, L_2 represents the curvature, and L_3 represents the rate of change of the curvature. In addition, each lane line is an independent cubic spline model during the fitting process.

Result

This study was completed based on the following hardware and software conditions. Hardware includes CPU: Intel(R) Core(TM) i7-9750 H CPU@2.60 GHz, RAM: 16 G; Camera: MV-EM200C; Software includes operating system: Ubuntu18.04, ROS: Melodic version; Programming language: Python.

Offline Experimental Verification and Results

The algorithm implemented in this paper is based on Python and uses the TensorFlow deep learning framework to build an instance segmentation network. Via the ADAM optimizer, the dataset is iteratively trained 12,000 epochs. The batch size is set to 8, the initial learning rate is set to 0.0005, and it decays exponentially with a decay exponent of 0.6.

The model's accuracy needs to be evaluated in object detection in image processing. The commonly used standard is IoU (Intersection over Union). The segmentation accuracy is involved in the segmentation process, and the evaluation standard is generally mIoU (Mean Intersection over Union); that is, the IoU value is obtained in all categories. The algorithm in this paper will also mIoU be used as the evaluation standard. For the detection results of lane lines, the corresponding ground truth values are marked and rasterized into point sets. After the points on the lane line are identified, the points in the set of ground truth points need to be considered, and the Euclidean distance between the above two is calculated. If it is not greater than 5, set it as a matching point, and its range is the true value area. The point outside the true value region, the false point, the aggregate length of this point is FP. The points in the true value region, namely the positive points, have a total length of TP. The difference between the sum of the lengths of the marked ground-

truth point set and TP, that is, the missed detection value, is set to FN, and $k + 1$ is the number of categories. The precision standard mIoU obtained from the above definition can be expressed as:

$$mIoU = \frac{1}{k + 1} \sum_{i=0}^k \frac{TP}{TP + FP + FN} \quad (8)$$

The differences between different models are mainly reflected in the backbone network. Therefore, the proposed algorithm is compared with the algorithms of different backbone networks in the selected datasets in this study. The content of the comparison is accuracy and real-time. Figure 7 shows the effect of the model after lane line detection on the test set of the TuSimple dataset.

The algorithm in this paper verifies the speed and accuracy of the model on the TuSimple dataset. The proposed algorithm is compared with existing models such as FCN (Zhang, Koubia, and Mohammed 2020), SegNet (Ambar 2019), LaneNet (Neven et al. 2018), and DeepLabV2 (Chen et al. 2018a) in terms of model detection accuracy on the TuSimple dataset, as shown in Table 1. We can see from Table 1 that the segmentation accuracy of the algorithm model in this paper is higher than that of most other

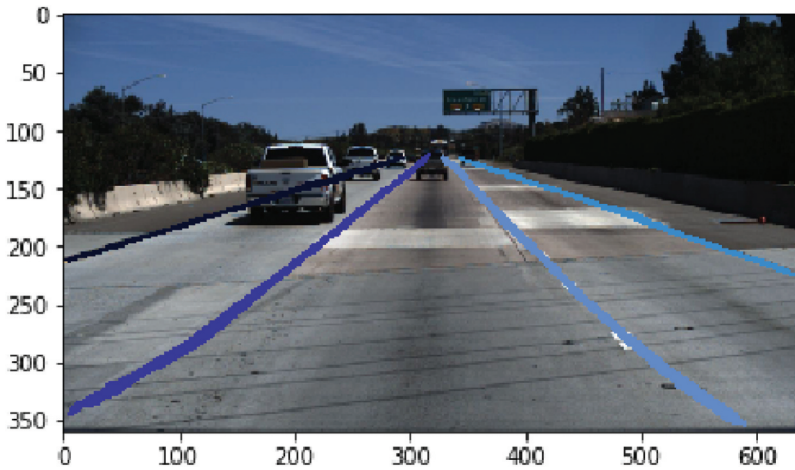


Figure 7. Lane detection on TuSimple test set of network model.

Table 1. Segmentation accuracy of network model.

Model Name	Backbone	mIoU (%)
FCN	VGG16	65.2
SegNet	VGG16	68.1
LaneNet	ENet	69.6
DeepLabV2	ResNet101	65.8
Ours	BiSeNetV2	71.2

segmentation models. The lane line detection accuracy of the algorithm in this paper on the TuSimple dataset is about 2% higher than that of the LaneNet model and about 6% higher than most other models, with better robustness.

Table 2 compares the processing speed of the neural network model of the algorithm in this paper with the LaneNet model. As shown in Table 2, on the TuSimple data set, the neural network model of the algorithm in this paper has dramatically improved the processing speed compared with the LaneNet model. The detection speed is 1.3 times that of the LaneNet.

Real Vehicle Verification under ROS

ROS (Robot Operating System) is often used in robot development and is one of its application platforms. It can carry out message passing, operate across platforms, and have the advantages of distributed computing. In addition, ROS is highly integrated, can be programmed in modules, and the community is more active than other open source communities.

The visualization of lane line detection can be realized by calling the rviz module in the ROS. A node is a process that performs computing tasks. A system can consist of many nodes. These nodes can communicate through published or subscribed topics, services (Saad and Liljenquist 2014). This study establishes a lane line detection node, as shown in Figure 8. First, build a node /lane for subscribing to the topic /cam/image about lane line images published in real-time by the camera driver. At the same time, convert the subscribed image information into a format recognized by OpenCV. This step can be completed by calling the cv_bridge module in the ROS library. Then use the lane line detection module to process the image data, and finally publish the lane line detection results in real-time in the form of a session, and the detection results can be obtained by subscribing to the session.

The camera is installed on the intelligent vehicle, and its parameters such as its height and inclination angle are determined. The driving speed is about 30 km/h, the image captured by the camera is about 60 FPS, and the image data format is

Table 2. Processing speed of network model.

Model Name	FPS
LaneNet	46
Ours	60

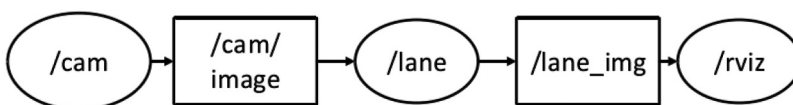


Figure 8. Lane line detection ROS node.

RGB format. In order to verify the robustness of the lane line fitting algorithm, this study performed current lane line detection and multi-lane line detection in multiple datasets. [Figure 9](#) shows the lane line detection results in different scenarios during the experiment. [Figure 10](#) is a time record of detecting lane lines.

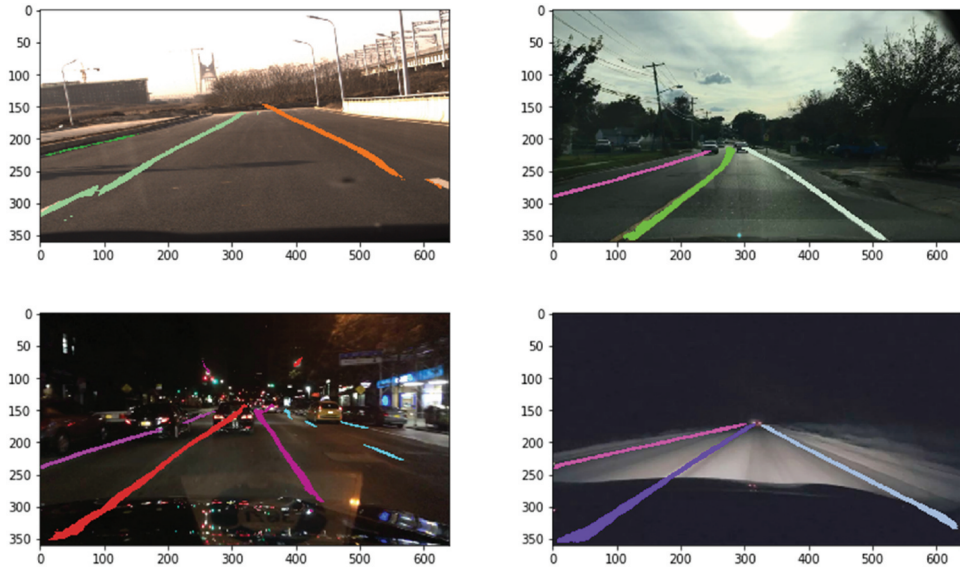


Figure 9. Lane line detection results in different scenarios.

```

liuyv@liuyv: ~/catkin_ws
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
2021-06-01 13:27:31.974 | INFO | __main__.work_thread:137 - Single imgae inference cost time: 0.01119s
get one frame: Width[1920], Height[1200], PixelType[0x2100032], nFrameNum[5940]
2021-06-01 13:27:32.152 | INFO | __main__.work_thread:137 - Single imgae inference cost time: 0.00982s
get one frame: Width[1920], Height[1200], PixelType[0x2100032], nFrameNum[5946]
2021-06-01 13:27:32.318 | INFO | __main__.work_thread:137 - Single imgae inference cost time: 0.01157s
get one frame: Width[1920], Height[1200], PixelType[0x2100032], nFrameNum[5952]
2021-06-01 13:27:32.513 | INFO | __main__.work_thread:137 - Single imgae inference cost time: 0.01062s
get one frame: Width[1920], Height[1200], PixelType[0x2100032], nFrameNum[5958]
2021-06-01 13:27:32.682 | INFO | __main__.work_thread:137 - Single imgae inference cost time: 0.01077s
get one frame: Width[1920], Height[1200], PixelType[0x2100032], nFrameNum[5964]
2021-06-01 13:27:32.828 | INFO | __main__.work_thread:137 - Single imgae inference cost time: 0.00991s
get one frame: Width[1920], Height[1200], PixelType[0x2100032], nFrameNum[5968]
2021-06-01 13:27:32.976 | INFO | __main__.work_thread:137 - Single imgae inference cost time: 0.01017s
get one frame: Width[1920], Height[1200], PixelType[0x2100032], nFrameNum[5973]
2021-06-01 13:27:33.112 | INFO | __main__.work_thread:137 - Single imgae inference cost time: 0.01124s
get one frame: Width[1920], Height[1200], PixelType[0x2100032], nFrameNum[5978]
2021-06-01 13:27:33.232 | INFO | __main__.work_thread:137 - Single imgae inference cost time: 0.00979s
get one frame: Width[1920], Height[1200], PixelType[0x2100032], nFrameNum[5982]
2021-06-01 13:27:33.350 | INFO | __main__.work_thread:137 - Single imgae inference cost time: 0.01089s
get one frame: Width[1920], Height[1200], PixelType[0x2100032], nFrameNum[5985]
2021-06-01 13:27:33.486 | INFO | __main__.work_thread:137 - Single imgae inference cost time: 0.01024s
get one frame: Width[1920], Height[1200], PixelType[0x2100032], nFrameNum[5990]
2021-06-01 13:27:33.633 | INFO | __main__.work_thread:137 - Single imgae inference cost time: 0.01106s
get one frame: Width[1920], Height[1200], PixelType[0x2100032], nFrameNum[5995]
(tf_gpu) liuyv@liuyv:~/catkin_ws$
    
```

Figure 10. Time record of detecting lane lines.

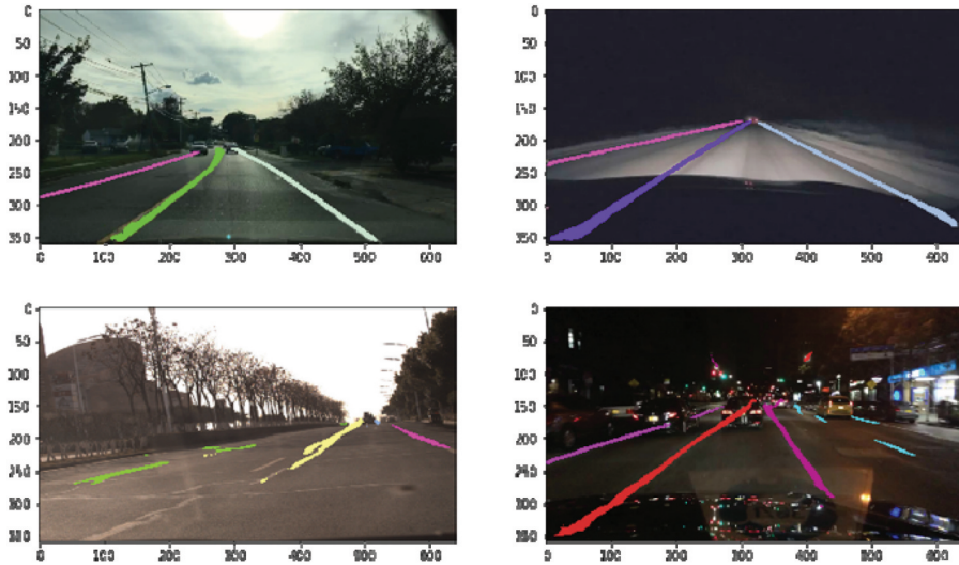


Figure 11. Detection results based on BiSeNet V2.

Lane Line Detection Evaluation Method

For the lane line detection algorithm, there is no standardized index of its pros and cons in the existing literature. Therefore, after referring to the literature (Borkar, Hayes, and Smith 2010), in terms of accuracy, this paper unifies the evaluation index: lateral offset; in addition, a real-time standard is also proposed. The position marked by the detection result may be different from the actual lane line position during the experiment. Therefore, the above difference value can be used as a standard for evaluating the accuracy of the results, so the related formula of lateral distance (LD) is defined, its unit is pixel.

After the detection of the lane line is completed, the lateral offset calculation can be performed on the detection result, and the formula is as follows.

$$LD = \frac{1}{N} \sum_{i=1}^N |T_{(i,f)} - P_{(i,f)}| \quad (9)$$

In the formula, T is the calibrated truth point; f is the number of video frames; P is the algorithm identification point; N is the number of lane line. It can be seen from the above formula that for the actual value and the detection result, the lateral offset only measures the difference between the two and has nothing to do with the width of the lane line. This criterion is used to evaluate the accuracy of the results in pixels.

Since the above algorithms have been integrated into ROS, they can be used for real-time lane line detection of intelligent vehicles. Therefore, a real-time standard can be proposed, the average time for detecting each frame of lane line images.

Table 3. Four representative scenes.

Scenes	Periods	Road Conditions
One	Daytime	Lane Intact
Two	Daytime	Lane Broken or Obscured
Three	Night	Lane Intact
Four	Night	Lane Broken or Obscured

In reality, the driving scene is complex and changeable. Therefore, in the field of lane line detection, the standard test database has not yet been unified. In addition, some references are detected on the original image, while others are detected after processing the image using an inverse perspective transformation. By considering the above reasons, part of the data is taken from the road video during the cycling process to evaluate the algorithm in this paper.

The Algorithm Detection Result

This section uses the above evaluation metrics to measure the algorithm's performance. Four representative scenes are selected, at different periods, including various road conditions and various road disturbances, such as a tree or building shadows, lane line damage, vehicle obscuring, etc. The details are shown in Table 3.

Figures 11 and 12 show the fitting results of the detection method based on BiSeNet V2 and the relevant results of the detection method based on the improved PHT. The comparison between the two is as follows.

The specific evaluation data is shown in the following table. Table 4 shows the fitting results of the detection method based on BiSeNet V2 and the relevant results of the detection method based on the improved PHT. The comparison between the two is as follows.

According to the above table, no matter in which scenario, the lane line detection method based on BiSeNet V2 is more accurate than the detection method based on PHT.

In different scenarios, the detection time of each frame of lane line image is different. By adding timestamps before and after the algorithm and subtracting the start time point from the algorithm end time point, the algorithm time-consuming can be calculated. The specific real-time evaluation data is shown in Table 5, which is mainly about the time-consuming comparison of lane line detection during accurate vehicle testing. Different scenarios, i.e., standard lighting, night road sections, shadow occlusion, intense light irradiation, curved roads, etc., record the time spent detecting lane lines and calculate the average detection time per frame, which is convenient for intuitive comparison



Figure 12. Detection results based on PHT.

Table 4. LD value of fitting results (pixel).

Scenes	BiSeNet V2	Improved PHT
One	2.1566	8.3625
Two	2.2568	9.5541
Three	2.6860	9.8361
Four	2.7015	10.36

Table 5. Detection time per frame (ms).

Scenes	BiSeNet V2	Improved PHT
Normal Lighting	10	35
Night Road Sections	12	40
Shadow Occlusion	16	38
Strong Light Irradiation	18	39
Curved Roads	13	46
AVERAGE TIME	13.8	39.6

Combining the two criteria, the comparison results of the lane line detection based on the method based on BiSeNet V2 and the improved probabilistic Hough transform algorithm show that the algorithm based on BiSeNet V2 has higher performance than the improved probabilistic Hough transform algorithm. The accuracy is about 3.9 times that of the other algorithm, and the speed is 2.9 times.

Conclusion

Most deep learning-based lane line detection algorithms have not been tested on actual vehicles and are not practical. At the same time, their detection algorithm still has room for improvement in detection accuracy, detection speed, and robustness. These inspired me to propose a dual-branch neural network model based on BiSeNet V2 for lane line image segmentation and apply it to actual vehicles to test its various performance indicators. The model is trained and verified based on the TuSimple data set. The algorithm has a noticeable performance improvement compared with other algorithms under the two evaluation indicators of mIoU and FPS. In the actual vehicle test, the accuracy and real-time performance of the two algorithms are evaluated. Finally, the following conclusions are drawn: the algorithm's accuracy and detection speed are increased to about 3.9 and 2.9 times, respectively, that of the improved PHT algorithm under the two evaluation indicators of LD and the detection time of each image frame, therefore the method proposed in this paper has practical application value.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

Research supported by the Natural Science Foundation of Hebei Province, China [No. F2021402011].

References

- Ambar, M. K. 2019. *Road segmentation in segnet*. Nicosia: Near East University.
- Borkar, A., M. Hayes, and M. T. Smith. 2010. An efficient method to generate ground truth for evaluating lane detection systems. In *IEEE International Conference on Acoustics Speech & Signal Processing IEEE*, Dallas, Texas, U.S.A.
- Chang, D., V. Chirakkal, S. Goswami, M. Hasan, T. Jung, J. Kang, S.-C. Kee, D. Lee, and A. Pratap Singh. 2019. Multi-lane detection using instance segmentation and attentive voting. In *2019 19th International Conference on Control, Automation and Systems (ICCAS)*, International Convention Center Jeju, Korea, 1538–42. IEEE.
- Chen, L.-C., G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. 2018a. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40 (4):834–48. doi:10.1109/tpami.2017.2699184.
- Chen, P.-R., L. Shao-Yuan, H.-M. Hang, S. Wei Chan, and J.-J. Lin. 2018b. Efficient road lane marking detection with deep learning. In *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, Shanghai, China, 1–5. IEEE.
- de Brabandere, B., D. Neven, and L. V. Gool. 2017. Semantic instance segmentation with a discriminative loss function.

- Denison, D. G., B. K. Mallick, A. F. M. Smith, et al. 1998. Automatic Bayesian Curve Fitting. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 60 (2):333–50. doi:10.1111/1467-9868.00128.
- Farahnak-Ghazani, F., and M. S. Baghshah. 2016. Multi-label classification with feature-aware implicit encoding and generalized cross-entropy loss. In *2016 24th Iranian Conference on Electrical Engineering (ICEE)*. IEEE.
- Hou, Y., M. Zheng, C. Liu, and C. Change Loy. 2019. Learning lightweight lane detection cnns by self-attention distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, 1013–21.
- Kaibo, Q., J. Ning, and D. Peishou. 2015. Curve fitting based on least squares method. *Business* 3:1.
- Lee, S., J. Kim, J. Shin Yoon, S. Shin, O. Bailo, N. Kim, T.-H. Lee, H. Seok Hong, S.-H. Han, and I. So Kweon. 2017. Vpgnet: Vanishing point guided network for lane and road marking detection and recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, 1947–55.
- Li, W., et al. 2019. A lane detection network based on IBN and attention. *Multimedia Tools and Applications*
- Libiao, J., and T. Qilong. 2019. The lane line detection in complex scene based on instance segmentation. *Machine Design and Manufacturing Engineering* 48 (5):6.
- Liu, X., Z. Deng, and G. Yang. 2017. Drivable road detection based on dilated FPN with feature aggregation. In *IEEE International Conference on Tools with Artificial Intelligence IEEE*.
- Liu, T., Z. Chen, Y. Yang, W. Zehao, and L. Haowei. 2020. Lane detection in low-light conditions using an efficient data enhancement: Light conditions style transfer. *arXiv preprint arXiv:2002.01177*.
- Mohsen, G., N. Cedric, B. Nora, O. 'Booij, and H. Michael. 2018. El-gan: embedding loss driven generative adversarial networks for lane detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*.
- Neven, D., et al. 2018. *Towards end-to-end lane detection: an instance segmentation approach*. IEEE
- Pan, X., J. Shi, P. Luo, X. Wang, and X. Tang. 2017. Spatial as deep: Spatial cnn for traffic scene understanding. *arXiv preprint arXiv:1712.06080*.
- Saad, A., and J. Liljenquist. A multi-robot testbed for robotics programming education and research. In *Proceedings of the 2014 ACM Southeast Regional Conference*. 2014. 10.1145/2638404.2675737
- Shao-Yuan, L., H.-M. Hang, S.-W. Chan, and J.-J. Lin. 2019. Multi-class lane semantic segmentation using efficient convolutional networks. In *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, 1–6. IEEE.
- Tran, T. N., K. Drab, and M. Daszykowski. 2013. Revised DBSCAN algorithm to cluster data with dense adjacent clusters - ScienceDirect. *Chemometrics and Intelligent Laboratory Systems* 120:92–96. doi:10.1016/j.chemolab.2012.11.006.
- TuSimple. 2018. Accessed September 8. 4325. <https://github.com/TuSimple/tusimple-benchmark/issues/3>.
- Wang, W., et al. 2020. CNN based lane detection with instance segmentation in edge-cloud computing. *Journal of Cloud Computing* 9(1). doi: 10.1186/s13677-020-00172-z.
- Yu, C., C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang. 2021. BiSeNet V2: Bilateral network with guided aggregation for real-time semantic segmentation. *International Journal of Computer Vision* 129 (11):3051–68. doi:10.1007/s11263-021-01515-2.
- Zhang, S., A. E. Koubia, and K. Mohammed. 2020. *Traffic lane detection using FCN*.
- Zheng, W., P. Bo, Y. Liu, W. Wang, et al. 2012. Fast B-spline curve fitting by L-BFGS. *Computer Aided Geometric Design*. 29(7):448–62. doi:10.1016/j.cagd.2012.03.004.