Neurocomputing 413 (2020) 328–338

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Brief papers

Deep reinforcement learning based lane detection and localization

Zhiyuan Zhao, Qi Wang, Xuelong Li *

School of Computer Science and Center for OPTical IMagery Analysis and Learning (OPTIMAL), Northwestern Polytechnical University, Xi'an 710072, Shaanxi, PR China

ARTICLE INFO

Article history:
Received 16 January 2020
Revised 28 April 2020
Accepted 28 June 2020
Available online 09 July 2020
Communicated by Yongsheng Dong

Keywords:
Lane detection
Deep reinforcement learning
Lane localization
Q-Learning

ABSTRACT

Recently, deep-learning based lane detection methods effectively boost the development of Advanced Driver Assistance Systems (ADAS) and Self-Driving Systems. However, these methods only detect lane lines with sketchy bounding boxes while ignore the shape of specific curved lanes. To address the above problems, this paper introduces deep reinforcement learning into cursory lane detection models for accurate lane detection and localization. This model consists of two stages, namely the bounding box detector and landmark point localizer. To be specific, a bounding box level convolution neural network lane detector outputs the preliminary location of lanes in the form of bounding boxes. Then, a reinforcement based Deep Q-Learning Localizer (DQLL) accurately localizes the lanes as a group of landmarks to achieve better representation of curved lanes. Moreover, a pixel-level lane detection dataset named NWPU Lanes Dataset is constructed and released. It contains a variety of real traffic scenes and accurate masks of the lane lines. This approach achieves competitive performance in the released dataset and TuSimple Lane dataset. Furthermore, the codes and dataset will be released on https://github.com/tuzixini/DQLL.

1. Introduction

Advanced Driver Assistance Systems become more and more popular in recent years. Avoiding accidents and guiding the vehicles travel along appropriate lanes are two essential tasks of those auxiliary systems. Several technical essentials are introduced to realize these two goals, such as lane detection [1–3], road detection [4–8], forward vehicle collision warning [9,10], traffic sign detection [11,12], traffic congestion detection [13,14], and road marking detection [15,16]. Lane detection plays an irreplaceable role in above tasks and other advanced driving assist goals, like drivable area detection [17,18] and automatic parking [19,20], *etc.* As shown in Fig. 1, there are different ways to represent lanes, which include straight lines, bounding boxes, landmarks, and pixel masks. No matter what lane representation methods are used, the core target is to get precise locations of the lanes.

Before the extensive study and application of Deep Convolutional Neural Networks (DCNN), lots of works use low-level feature extractors to detect lane lines [21–23] and use several straight lines to represent lanes [24]. The straight lines work fine with straight lanes but fail for the curved one. To remedy the representation problems of the curved lanes, bounding boxes [25] and pixel-level masks [26] are introduced into lane detection methods. However, bounding boxes are not accurate enough, and pixel-level masks' prediction requires complex computation.

To address the problems mentioned above, we propose a deep reinforcement learning based network for lane detection and localization. It consists of a deep convolutional lane bounding box detector and a Deep Q-Learning localizer. The structural diagram of the proposed network is shown in Fig. 2. It is a two-stage sequential processing architecture. To be specific, the first stage is a modified Faster R-CNN [27], which detects the lanes in the form of bounding boxes. The second stage is a light weighted deep Q-learning landmark localizer, which consists of five convolution layers and three fully connected layers.

After getting bounding boxes from the detection stage, the initial landmarks uniformly distributed along the diagonal line of the bounding boxes. Then the lane localization task becomes a point moving game. The lane localizer plays the role of agent in this game. What the agent needs to do is moving the landmarks toward specific directions according to the current environment state, which consists of present point position, action history vector, and encoded image features. Finally, when the agent decides not to move any landmark points anymore, the position of all landmark points is output as the location of lanes.

To verify the effectiveness of the proposed method, we build a pixel-level lane dataset named NWPU Lanes Dataset, which contains 1964 traffic scenes images with well-labeled pixel-level lane masks.

* Corresponding author.
E-mail addresses: tuzixini@gmail.com (Z. Zhao), crabwq@nwpu.edu.cn (Q. Wang), li@nwpu.edu.cn (X. Li).

https://doi.org/10.1016/j.neucom.2020.06.094
0925-2312/© 2020 Elsevier B.V. All rights reserved.

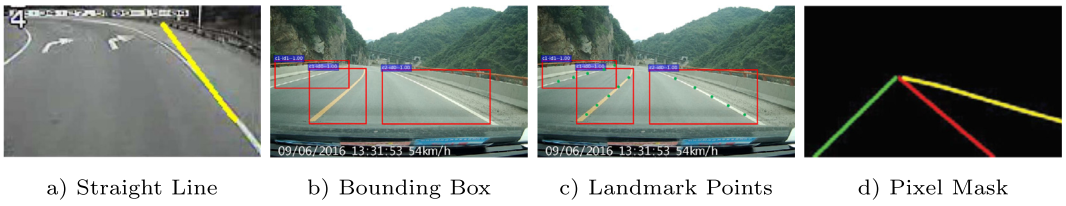


Fig. 1. Different representations of lanes. From left to right, the accuracy of the representation increase, and the amount of computation required for detection also becomes higher.

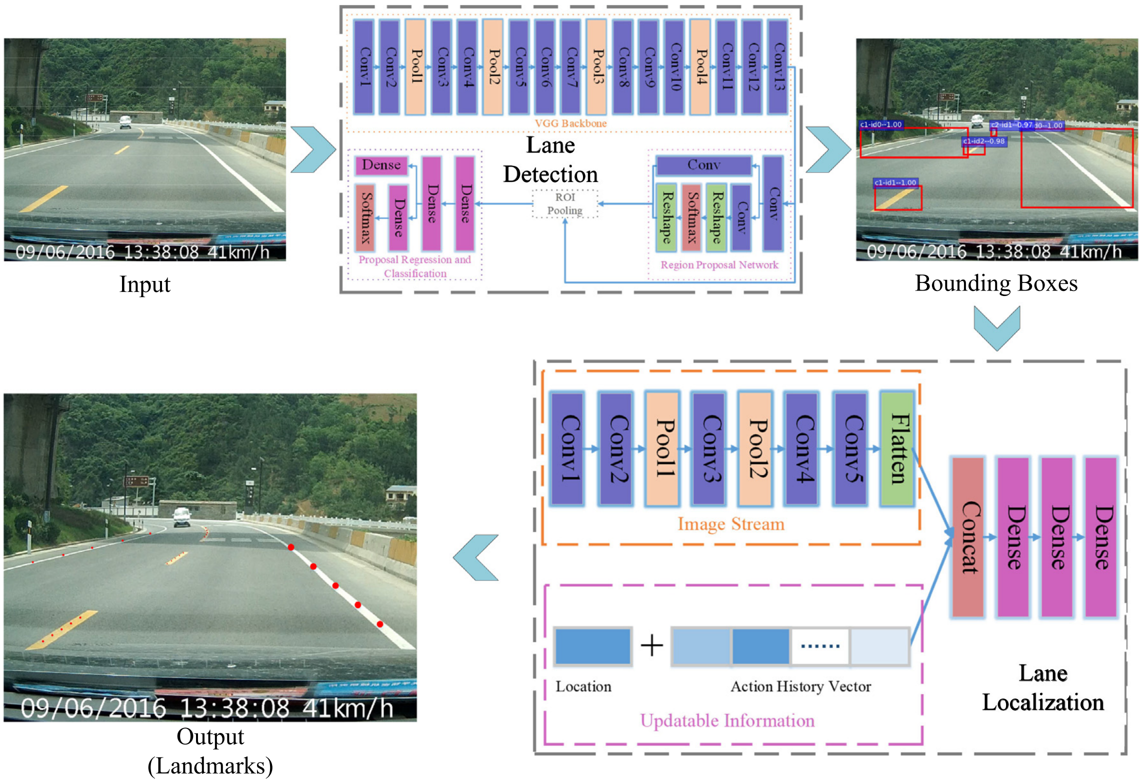


Fig. 2. The flowchart of the proposed lane detection and localization framework.

The main contributions of this work are summarized as follows:

- Define a new representation method for lane detection and localization, which reaches the balance of precision and computation.
- Deep Q-Learning Localizer (DQLL) accurately localizes the lanes as a group of landmarks, which achieves better representation for curved lanes.
- Build a pixel-level lane dataset NWPU Lanes Dataset, which contains carefully labeled urban images and contributes to the development of traffic scenes understanding.

The rest sections of this paper are arranged as follows: Section 2 introduces some related works. Section 3 focuses on the detailed implementation and structure of the proposed methods. The NWPU Lanes Dataset will be introduced in Section 4. The experiment results and further analysis are given in Section 5. Finally, Section 6 concludes the proposed method and discusses future works.

2. Related work

This section focuses on review serval previous works related to lane detection and reinforcement learning.

2.1. Traditional lane detection methods

Before the fast development of deep learning on the computer version, to construct easily identified features for lane, feature representations are manually designed base on its identical characteristics. Common feature extractors like Hough Transform [21] and Dark-Light-Dark (DLD) [22] are effective in simple conditions, but the performance suffers rapid decrease in complex scenes. Their sensitivity to noise causes this problem. Besides construct powerful feature extractors, some researchers first use Inverse Perspective Mapping (IPM) [28] to transfer the original image to the bird's-eye view. Then use the above feature extractors to generate features under this view. The view transform helps reduce redundant information and enhance the target representation. Thus, it improves the lane detection accuracy, but its effects drop a lot under extremely complex scenarios. The essential reason is that the features extracted by the low-level extractors like DLD and Hough Transform are not powerful enough.

2.2. Deep learning lane detection methods

Due to the rapid evolution of Deep Learning, the inefficient hand-crafted features are replaced by deep features extracted by

DCNN among many computer vision tasks. Compare to the above traditional feature extractors, for one specific task, DCNN can generate features with sufficient high-level semantic information from the input image. Moreover, its automatic fitting characteristics save lots of feature design works. Due to the high efficiency of DCNN, lots of works [29–32] try to use DCNNs to improve lane detection performance. Lee et al. [33] and Umar *et al.* [34]use vanishing point to guid network for lane and road marking detection. Some researchers focus on combine DCNNs with traditional methods. Van et al. [35] integrate deep neural networks with differentiable least-squares fitting. Neven et al. [36] use small convolutional network to estimate the parameters of IPM transformation.

2.3. Reinforcement learning

Forty years after Bellman proposes the theory of dynamic programming [37], Peter et al. put forward the value-based reinforcement method Q-Learning [38]. Mnih et al. [39] combine Q-Learning with deep learning methods from the Deep Q-Learning Network (DQN), which means use serval neural networks to replace the Q-table.

3. Methodology

In this section, we explain the detailed implementation and working principle of the proposed method.

3.1. Overview

As shown in Fig. 2, the proposed lane detection and localization framework consists of the detection stage and localization stage. The purpose of the detect part is to obtain the preliminary bounding box locations of the lanes. In order to cooperate with the next stage's localization process, we carefully consider the characteristics of lanes, through observing a variety of lanes like shown in Fig. 3 , we summarize them as follows:

- The lanes represented by bounding boxes always near the diagonal line of the rectangle.
- The lanes may go along different diagonals of the bounding boxes, which means from left top to right bottom or from left bottom to right top. The former one always appears on the left side of the field of views, while the latter one generally appears on the right side.
- The diagonals of the bounding boxes can be used to represent the position of the straight lane lines roughly. As for the curved lanes, it fails because of the massive difference in lane shape.

According to those unique characteristics of the lanes, we manually split it into two types. The first one is the lanes go along with the diagonals from the left bottom to the right top, like shown in Fig. 3a). The other one is the lanes go along with the other diagonal, like the one in Fig. 3c). After the lane detector gets the bounding boxes of lanes, the diagonal line which the lane goes along with will be a critical factor in determining the initial position of landmark points. The detailed landmark initialization process will be introduced in Section 3.3. After that, the Deep Q-Learning Localizer tries to move the landmark points to the correct position, respectively.

3.2. Lane detection

The modified common object detector is first retrained on the lane dataset, and it is applied here to obtain the preliminary position of the lanes. Technically speaking, almost all typical object detector like [40–43] can be used here. We adopt Faster R-CNN as the base-line lane bounding boxes detector in the first stage. The lanes are detected and classified into different types according to their slops, and the lane types are unified with what we discuss in Section 3.1. The intact working flow of the detection stage is shown in Fig. 4.

Faster R-CNN is the following-up development of RCNN [44] and Fast R-CNN [45]. Compared with the work of the previous two generations, Faster R-CNN uses CNN to complete proposal generation, regression, and classification. An input image only spread once in the network, which improves the efficiency of the network.

As shown in Fig. 4, the backbone CNN used here is VGG [46], it extracts the convolutional feature maps from the input three-channel RGB image. The whole image feature vector and the image information is then sent to RPN to generate region proposals. ROI pooling layer helps force the region proposal feature vector into a fixed size. The proposal regression network and proposal classification network use serval fully connected layers to get the bounding box bias and class probabilities, respectively. The detail network structure is shown in Fig. 4, where *Conv* means convolutional layers, and *Dense* means fully connected layers.

The final output of the lane detection stage is the bounding box locations and lane type of all lanes inside the input image, as shown in Figs. 3 and 4.

3.3. Lane localization

After getting the general location information of lanes in the detection stage, the following step is to localize them accurately. To this end, we use five landmark points to positioning the lanes. The landmarks are uniformly initialized inside the bounding box. Thus, the localization stage becomes a point moving game which aims to shift all landmarks to the correct positions. A reinforcement learning based deep Q-learning lane localizer is applied to play this game. Compare to bounding boxes, landmarks improve the representation ability for curved lanes effectively and provide more precise position information. The precise localization method will be introduced in this section.

3.3.1. Game definition

As shown in Fig. 3, each bounding box from the detection stage is divided into six equal areas by five cut-off lines in the horizontal direction together with the box. The diagonal line in which the lane line goes along with intersects these five divider lines at serval points. Those points are used as the initial position of the landmark points.

In the following stage, we try to tackle the point localization game via a deep reinforcement learning method. The learning policy used here is Q-Learning [38] method. In the original Q table, it recodes for each different environment states, which action choice leads to the highest reward. The initial Q table gives random action decisions, and it updates with the training process according to the following formula:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a} Q(s',a) - Q(s,a)\right], \tag{1}$$

where s is environment state, a is the action choice under environment state s. α and γ represent the learning rate and attenuation coefficient during the learning process. After take action a, the current environment state s changes into state s' and the reward r will be observed. $Q(s,a)$ is the current expectation reward of action a in state s. The observed reward plus the possible maximum reward of next state multiply by attenuation coefficient $r + \gamma \max_a Q(s',a)$

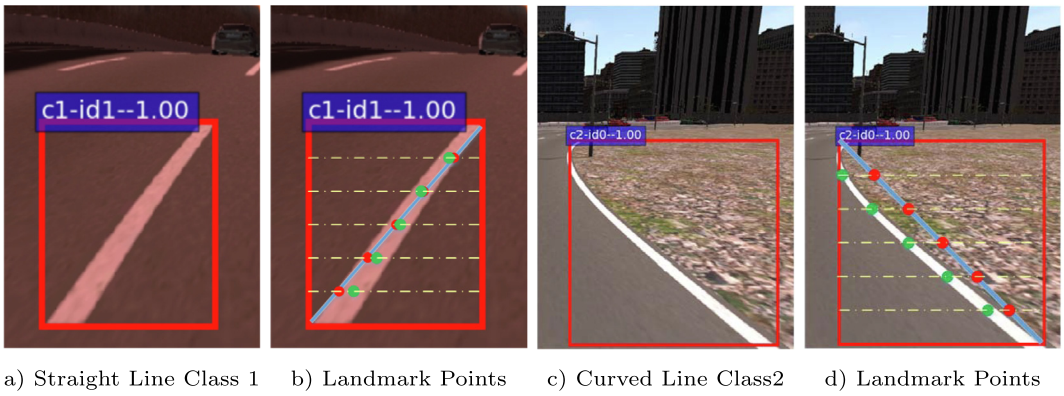


Fig. 3. Different kinds of lanes, straight or curved lanes, lane go from left bottom to right top or from right bottom to left top. The red points are the initial landmark positions of lane localization stage, and the green points are the final landmark positions after the point moving game. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

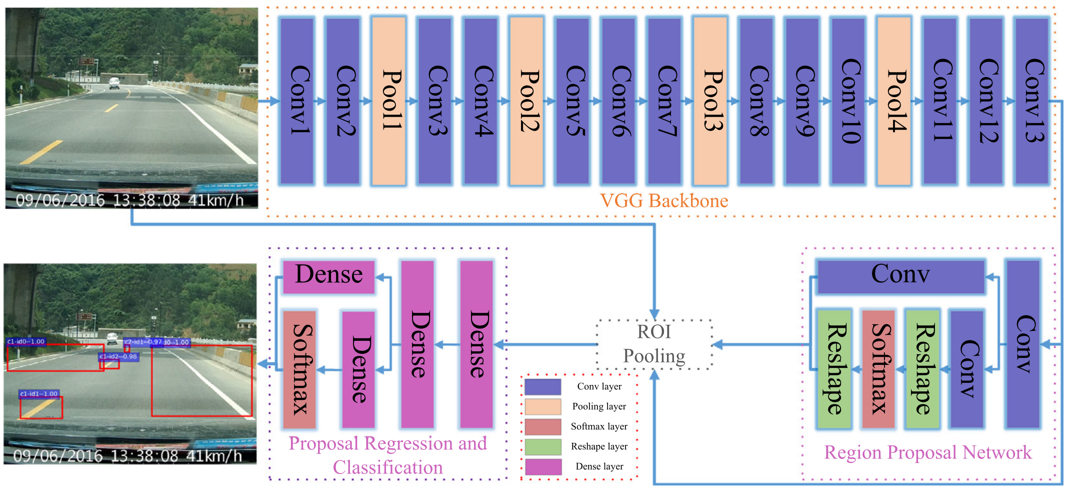


Fig. 4. The working flow of the Faster R-CNN detector. It contains a convolutional feature extractor, a region proposal network, a region of interest pooling module, a region proposal classifier, and a bounding box regression layer.

together forms the ideal real reward solved by Bellman Equation and Greedy Strategy.

The agent only takes movements among the given action options, and it makes action choice according to the current environment state and the learned Q table. The algorithm updates the Q table base on the rewards of different actions under the current environment state. The reward score reflects whether the action leads to a better result or worse, and it is calculated by the reward functions. Besides the Q table, the environment state, action options, and reward functions together constitute the deep Q learning process. Those three key components will be detail introduced in the following subsections.

3.3.2. Environment state

The environment state contains the factors that could influence the action decision consequence. For this point moving game, the current location information of the selected landmark point, and the image block all help to find the right position. We also take the actions that have been done before into consideration, which we call it Action History Vector. Thus, the current environment state can be represented by Eq. (2):

$$\mathcal{S} = \mathcal{E}(\mathcal{I}_b) \oplus x \oplus \bar{\mathcal{H}}, \tag{2}$$

where $\mathcal{S}$ is the current environment state, $\mathcal{E}(\cdot)$ means the feature extractor that encodes the input color images into feature vectors, which leads to a better representation ability of the environment. The $\mathcal{I}_b$ represents the bounding box image blocks truncated from the complete original figure. x is the position of the landmark point under the current state. $\bar{\mathcal{H}}$ is the action history vector composed of serval previous action records, and $\oplus$ serve as the concatenate procedure.

Fig. 5 demonstrates the composition of environment state. The feature extractor $\mathcal{E}(\cdot)$ consists of serval small convolutional neural networks. Therefore, we no longer need to construct features manually, and the extractor itself will automatically adjust during the training and learning process to extract efficient features.

3.3.3. Action options

The set of all optional actions constitutes the agent's action space, which means the agent can only make action decisions among the scheduled action set. For this situation, the landmark point's vertical coordinate is a fixed value, so it can only move horizontally. Thus, we have artificially defined three alternative action types.

We call the first action type delete action. It means the agent decides to delete the current point or take other actions. The points which locate out of the appropriate range or too far away from the actual lane position may be deleted.

For the landmark points under normal state, the agent tries to move the point toward the correct direction. The points have two

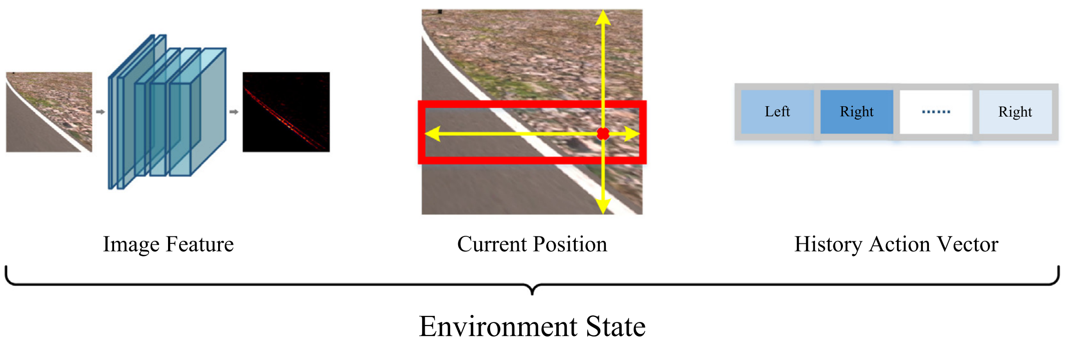


Fig. 5. The environment state of the landmark moving game contains encoded image feature vectors, action history vectors, and the current position of the moving landmark point.

moving directions along the horizontal line, so the moving action contains movements toward the left or right side.

Finally, when the point is close enough to the expected position, the agent has to decide whether the current position is the final position or not. The terminal action decides to cut off the point moving process or run into the next action choice.

All the action choices and there corresponding actual pixel level point movements are shown in Table 1. Where x means the position of the current landmark point.

3.3.4. Reward functions

After the agent takes action a under environment state s, the state changes into s'. Base on these three observations $s' \leftarrow s \otimes a$, we can calculate the reward according to reward functions. Different action choices lead to a different state. It can be a better state or a worse one. Therefore, we divide the action choices into three types on the basis of the results they lead to. Those action choices are defined and analyzed in the following paragraphs.

Invalid Action Choices: If action a moves the landmark point out of the appropriate image range, deletes the points that should be kept or keeps the points should be deleted, we call it the invalid action choice. The reward score will be given by the following formula:

$$R_i(s,s') = -5, \tag{3}$$

which means all action choices that lead to particularly wicked results are severely forbidden.

Regular Action Choices: If the action choice is not the invalid one mentioned previously, and it is a moving action, we call this choice a regular action choice. We define the distance between the current point position and the ground truth under environment state s as $d(s)$. Considering the regular action choices, it may make the distance longer or shorter, so the rewards are given by the relation between original distance $d(s)$ and the new one $d(s')$, it is defined by Eq. (4):

$$R_n(s,s') = \begin{cases} +1, & if \quad d(s') < d(s) \\ -1, & otherwise \end{cases}, \tag{4}$$

this reflects that the action choices which take the point moving toward the ground truth's direction earn one score otherwise lose one score.

Terminal Action Choices: If the agent makes decisions that terminate the point moving process, we call those decisions terminal action choices. The agent earns positive scores by terminal action choices when the current point and ground truth position is near enough. Otherwise, the agent gets negative scores if it stops the moving process at an inappropriate time. Following equation gives the reward function of terminal action choices:

$$R_t(s,s') = \begin{cases} +3, & if \quad d(s) <= 5 \\ -3, & otherwise \end{cases}. \tag{5}$$

3.3.5. Overview

The complete lane localization working flow is demonstrated in Fig. 6. All detected bounding boxes are firstly truncated from the complete image and then resized into unified size $[100, 100, 3]$ before send into Deep Q-Learning Localizer(DQLL). The localizer initialize five landmark points according to the type of the bounding box, which means the horizontal and vertical coordinates of the five landmarks are evenly distributed between 0 and 100. Moreover, these five points may go along different diagonal lines for different lanes. After initialization, all five landmark points are localized separately.

Different from traditional Q-Learning methods, DQN uses neural networks to take the place of Q-Tables. The network gives action choice decisions according to the environment state and its parameter values. The learning process updates all parameters of the network and forces it makes the right decisions. Fig. 6 illustrates the structure of the decision-making network on the right side. The detailed network architecture is given in Table 2.

The "Conv1: (k(3,3),c(3->48),s(1),NoPadding)" means the convolutional layer named Conv1 uses 48 convolution kernels of size $3 \times 3 \times 3$ with strid length 2 and nopadding. "FC: 5393 -> 512" is fully connected layer with input size 5393 and output size 512. The length of the action history vector influences the shape of the first fully connected layer's input side. Here we use four past actions to form the action history vector. Thus the input size of the first fully connected layer is $1 \times 21 \times 256 + 1 + 4 \times 4 = 5393$. $1 \times 21 \times 256$ is the output shape of the feature encoder,$1 + 4 \times 4$ contains four step action history vector.

The loss function used for the training process of Deep Q-Learning Network is Mean Squared Error (MSE) loss, and the following equation defines it:

Table 1
Optional action choices and their corresponding operations. All movements are in the unit of pixels.

Action	Delete action	Moving Action		Terminal action
		Left	Right	
Operation	Delete Point	$x' = x - 5$	$x' = x + 5$	$x' = x$, End Game

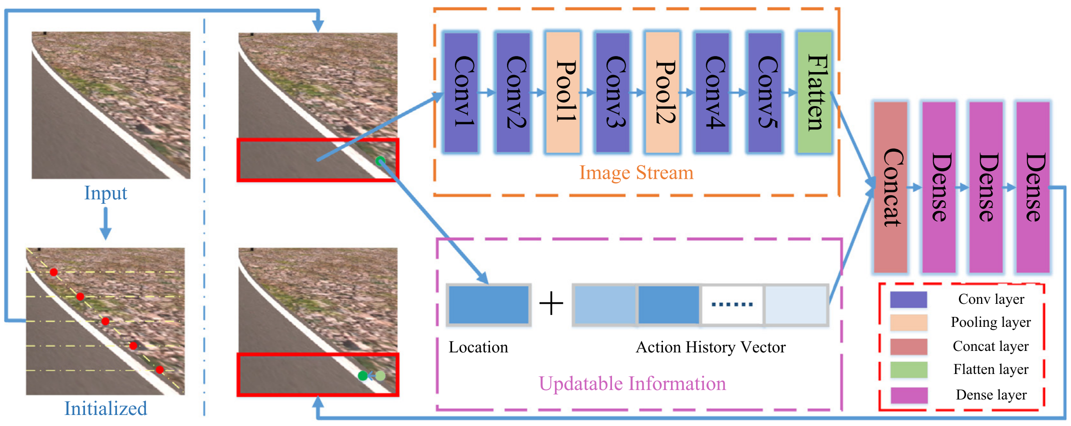


Fig. 6. The complete working process of the lane localization stage. The truncated lane line blocks are resized, and initialized according to the type of bounding box. All points are localized one by one using the DQLL to achieve better representation of curved lanes.

Table 2
This table shows the architecture of the environment state convolutional feature extractor and the decision maker.

Feature Extractor	Decision Maker (Hist:4)
Conv1: (k(3,3),c(3->48),s(1),NoPadding)	FC: 5393 ->512
Conv2: (k(3,3),c(48->96),s(1),NoPadding)	
Max Pooling	
Conv3: (k(2,2),c(96->128),s(1),NoPadding)	FC: 512 ->256
Max Pooling	
Conv4: (k(2,2),c(128->192),s(1),NoPadding)	
Conv5: (k(2,2),c(192->256),s(1),NoPadding)	FC: 256 ->4
Flatten Layer	
Concatenation	

$$MSE = \frac{1}{N}\sum_{a=1}^{N}\left(Q_a - \widehat{Q}_a\right)^2, \tag{6}$$

where N is the total number of all action choices, $\widehat{Q}_a$ is the estimate Q value of the action a while Q_a is the real Q value.

4. Dataset

On account of the lack of lane detection datasets, we build a lane dataset named NWPU Lanes Dataset with pixel-level labels. It comes from the recorded videos under real driving scenes.

Due to the difficulty of manually collecting and labeling the real scenes data, we select some virtual data from the dataset [47] to assist our training process. Those images and label masks are synthetic data generated by software with precise annotations.

4.1. NWPU lanes dataset

The automobile video data recorder collects thirteen videos during real driving scenes. Twelve of the clips are three minutes long, and the remaining one is one minute and 38 s long. After sampling one frame per second, a total of 2258 initial images are obtained. Through the observation of these pictures, we found that there are still many images that could not be used in the preliminary samples due to the problems such as vehicle stop, congestion, and shielding in the real driving process. So we did another manual image deletion and kept 1964 images after the deletion. Next, we use the marking tool developed by ourselves to carry out accurate pixel-level lane line marking. After the labeling is completed, bounding boxes can be generated through pixel-level marking.

Fig. 7's part (A) in the left side, demonstrates the driving scenes and their corresponding masks from the NWPU Lanes Dataset. All the data finally obtained are divided into the train set and the test set, of which test set accounts for 20%. The data distribution is given in Table 3.

4.2. Synthetic dataset

For the manually collected data, there will be inevitable errors during the labeling work. In the process of lane line localization, the input image is a small-size image that only contains the lane line area truncated from the original image. Therefore, the error at the original image-level may be magnified during the localization process. Compared with manual annotation, the virtual data generated by the software will have completely accurate pixel-level annotations. So, we use not only real dataset built by ourselves, but also virtual data under appropriate scenarios selected from other datasets for train and test. The SYNTHIA dataset contains a large amount of generated data, which are constructed in different scenarios, time, seasons, and weather settings. We manually select some scenarios that are close to our NWPU Lanes Dataset. The addition of these virtual scene data effectively boosts the learning process of the DQLL. We also split those data into the train set and test set, the split rule is consistent with the self-build dataset, and the detail information is shown in Table 3.

5. Experiments and discussion

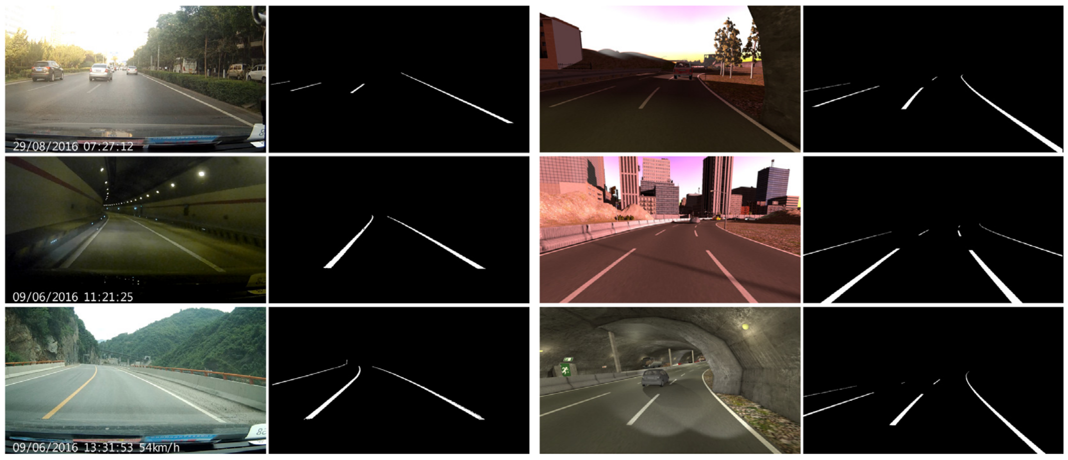
5.1. Evaluation metrics

To verify the effectiveness of the proposed method, we define two evaluation metrics to measure the experiment results from different perspectives. The definitions of these two parameters are closely related to the specific realization of lane line localization in Section 3. The first one we call it hit rate reflects the localization accuracy, while the second one average step measures the localization speed. These two variable both defined during a complete test process. Here are the definitions:

Hit Rate: If the final terminated point locates near the ground truth position enough, which means the distance between two points less than 5 pixels, we name this point "Hit Point". Thus we define the hit rate α by the following formula:

$$\alpha = \frac{N_h}{N}, \tag{7}$$

where N_h is the total number of the hit points during this specific test period, and N is the number of all landmark points it also equals to five times the number of the bounding boxes. Naturally, the hit



a) Manually Collected Data b) Virtual Data

Fig. 7. This figure shows the manually collected images, and their label masks compared with the virtual data.

Table 3
The distribution of the number of images used in the experiment.

Data Part	Train Set	Test Set	Total
Real	490	124	614
Syn	1572	392	1964

rate is a number no less than 0 and not greater than 1, the higher the value of α, the better the model we have get.

Average Step: On account of the diverse distance between initial landmark point and the ground truth, the DQLL needs to take a different number of actions to achieve the final point location. Thus we define average step β use the following equation:

$$\beta = \frac{S}{N}, \tag{8}$$

where S is the total number of action steps for all landmark points under this test period, and N have the same meaning as the one in Eq. (7). β tells us how many steps needed for localizing one landmark point. It is also the embodiment of DQLL's localization speed. In this case, it is a small positive number (usually less than 5).

5.2. Experiment setting

All the training and evaluation process are construct with one NVIDIA GTX 1080Ti GPU and Inter Core i7-6800K@3.4 GHz CPU under Ubuntu 14.10 operation system and TensorFlow [48] or PyTorch [49] framework. Two stages rely on quite independent experiment settings. This section introduces those settings in detail, respectively.

Detection Stage: The bounding boxes needed for the detection stage are generated base on the pixel level lane dataset by a simple connected component detection algorithm. During the training process, the Faster R-CNN runs with a batch size of 1, and the bounding box classification gets the batch size of 300. The learning rate and weight decay are set to 0.001 and 0.0005, respectively.

Localization Stage: For the training process of DQLL, it needs different ground truth data from the detection stage. We further process the bounding box data of the first step. Firstly, each rectangle box is truncated from the original image independently. Then, every box is divided into six equal-area horizontal rectangles by five dividers. The dividers and the lane lines intersect in a short section. Finally, the midpoint of the short section is treated as the ground truth of the corresponding landmark point. It is not until this moment that the data for training the DQLL is ready. The learning rate of the localization stage is set to 0.0001, and it is trained with batch size 1024.

5.3. Experiment results

In order to verify the effectiveness of the DQLL method, we carry out experimental tests on NWPU Lanes and TuSimple Lane [50] datasets, detail test results and analysis are given below.

Experiments on NWPU Lanes Dataset: Table 4 shows the test results when we combine DQLL with detection, segmentation, or other lane detection methods. As we expected, the initial accuracy of the segmentation-based method is higher than that of the detection-based method.

Experiments on TuSimple Lane Dataset: TuSimple Lane dataset contains 3626 images for training and 2782 for testing. We convert the labeled data into the form we need for the training and evaluation of DQLL. Table 5 shows the results of different methods.

The overall Hit Rate on the TuSimple Lane dataset is higher than the NWPU Lanes dataset. This means that the former has a more massive amount of straight lines. No matter what lane detection method is used in the first stage, DQLL can effectively boosts the

Table 4
Hit rate and average step of different methods on NWPU lanes dataset.

Method	Hit Rate	Avg. Steps
Faster R-CNN	37.55%	–
Faster R-CNN + DQLL	**75.51%**	**3.1830**
DeepLabV3P	81.36%	–
DeepLabV3P + DQLL	**86.05%**	**1.5408**
SCNN [51]	78.32%	–
SCNN + DQLL	**84.68%**	**1.7583**

Table 5
Hit rate and average step of different methods on TuSimple lane dataset.

Method	Hit Rate	Avg. Steps
Faster R-CNN [27]	71.85%	–
Faster R-CNN + DQLL	**86.96%**	**1.8370**
SCNN [51]	84.69%	–
SCNN + DQLL	**91.98%**	**1.5702**
PINet[52]	86.09%	–
PINet + DQLL	**93.36%**	**1.5311**

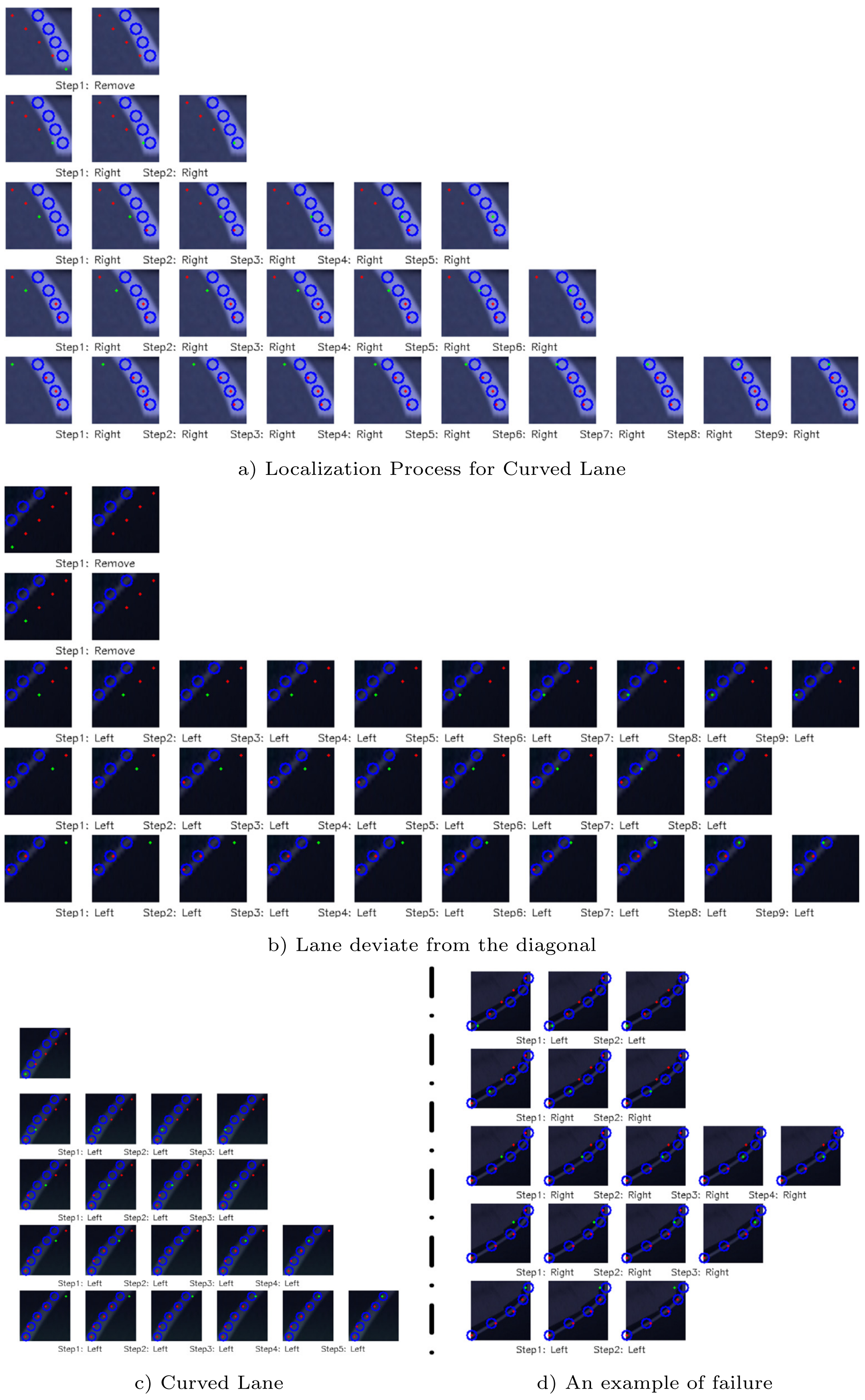

Step1: Remove
Step1: Right Step2: Right
Step1: Right Step2: Right Step3: Right Step4: Right Step5: Right
Step1: Right Step2: Right Step3: Right Step4: Right Step5: Right Step6: Right
Step1: Right Step2: Right Step3: Right Step4: Right Step5: Right Step6: Right Step7: Right Step8: Right Step9: Right
a) Localization Process for Curved Lane
Step1: Remove
Step1: Remove
Step1: Left Step2: Left Step3: Left Step4: Left Step5: Left Step6: Left Step7: Left Step8: Left Step9: Left
Step1: Left Step2: Left Step3: Left Step4: Left Step5: Left Step6: Left Step7: Left Step8: Left
Step1: Left Step2: Left Step3: Left Step4: Left Step5: Left Step6: Left Step7: Left Step8: Left Step9: Left
b) Lane deviate from the diagonal
Step1: Left Step2: Left Step3: Left
Step1: Left Step2: Left Step3: Left
Step1: Left Step2: Left Step3: Left Step4: Left
Step1: Left Step2: Left Step3: Left Step4: Left Step5: Left
c) Curved Lane
Step1: Left Step2: Left
Step1: Right Step2: Right
Step1: Right Step2: Right Step3: Right Step4: Right
Step1: Right Step2: Right Step3: Right
Step1: Left Step2: Left
d) An example of failure

Fig. 8. Visualized precise localization process. The first three are positive examples, and the last one fails while localizing the last landmark point.

Table 6
Results of DQLLs with different action history vector length.

Length of Hist	Hist:0	Hist:2	Hist:4	Hist:6	Hist:8	Hist:10
Hit Rate	69.00%	72.48%	**75.51%**	74.01%	69.10%	69.81%
Avg. Steps	3.524	3.364	3.183	**3.163**	3.467	3.461

Table 7
Comparison of DRL and DSL.

Method	DRL(Ours)	DSL	DSL	DSL
Loss Function	MSE	MSE	L1 Loss	Smooth L1 Loss
Hit Rate	**75.51%**	51.99%	64.47%	64.79%

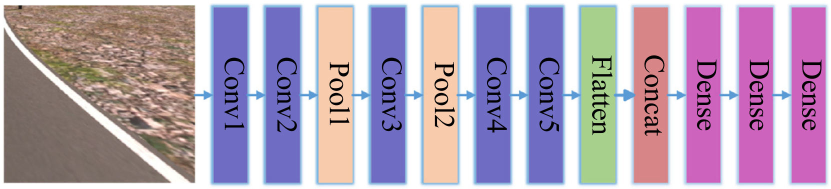


Fig. 9. The network architecture of the deep supervised learning method used to compare with DQLL.

performance of the original results on both lane datasets. The experimental results show that the number of average steps decreases with the improvement of the initial detection effect of the first stage. Because more reliable detection results can provide better guidance when initializing landmark points.

From the results of the experiment shown in Tables 4–6, it can be seen that no matter use what kind of experimental settings, DQLL can complete the localization process of one landmark point within four steps, which requires only about three steps on average for NWPU Lanes dataset, and less than two steps for TuSimple Lane dataset. Obviously, the length of the action history vector will influence the hit rate and average steps. The choice of appropriate length becomes important, and we put the concrete analysis in subsection 5.4.

DQLL Visualization: Fig. 8 demonstrates the visualization of DQLL localization process. All subfigures consist of five rows. Each row corresponds to one landmark point's localization process. The blue circles are the confidence interval of the ground truth landmarks, and the green dots are the current moving landmark point. Thus, for every subfigure, the upper left corner is the initial point location, and the lower right corner is the final output of the localization process. Subfigure (A) and (C) demonstrates the detail localization process of the normal curved lane lines. Compare to the initial landmarks' position, DQLL effectively moves the landmarks to fit the curved lines better. Subfigure (B) is a straight lane line but deviate from the center of the bounding box, therefore, two landmarks are removed and only three points left. The fourth subfigure gives a negative example, the initial position of the fifth landmark point is within the expected range, but it is wrongly moved out of the expected range by DQLL.

5.4. The influence of action history vector

Considering all components of the environment state, the action history vector is one of the easily changed variable factors. We manually design experiments to verify the influence of the action history vector and its length. The length of the action history vector is set into range $[0, 10]$ with a step length of two. All the experiment results are shown in Table 6.

Compare to DQLL without action history vector, no matter how long the history is, the DQLL with this vector improves the localization accuracy and reduces the average step needed. This proves that the action selections in the past help making decisions in the current state to some extent. Nevertheless, would a longer history vector lead to a better result? The experimental results show that the length of the history vector is not proportional to the performance of the model. The best record of hit rate occurs when the length of the history vector is 4, while the best average step appears when the length of the history vector is 6. Both of these two evaluation metrics tend to change from increase to decrease as the length of the action history vector rises. Thus, it is vital to choose an appropriate length of the history steps.

5.5. Comparison with supervised learning methods

To further verify the effectiveness of the proposed DQLL, we manually design serval Deep Supervised Learning (DSL) methods, those models have exactly the same network structures with DQLL. The only difference is that those DSL methods try to regress the five landmark points' positions directly, and they may train with different loss functions. Here, we use MSE, L1, and smooth L1 loss to train the DSL model, respectively. The experiment results are shown in Table 7. Fig. 9 illustrates the network architecture of the compared DSL model.

Loss functions can not make up for the performance gap between DQL and DSL. No matter under what conditions, DQL performs much better than the DSL methods. This comparison results prove that learning how to move the landmark point into the correct position is more efficient than directly regress the position of those points.

6. Conclusion

This paper proposes a two-stage lane detection and localization method to predict the precise location of the lanes effectively. To be specific, the deep convolutional detection model first detects the lane lines in the form of bounding boxes. Then the DQLL model accurately localizes the lanes as a set of landmark points base on the output of the previous stage. Furthermore, the pixel-level lane dataset named NWPU Lanes Dataset is released to help improve the lane detect performance. The experiment results demonstrate that the proposed method effectively improves the detection precision, especially for the curved lane lines. In the future, we will try to use more prior knowledge to improve detection performance.

CRediT authorship contribution statement

Zhiyuan Zhao: Software, Validation, Writing - original draft, Writing - review & editing. **Qi Wang:** Supervision, Writing - original draft, Writing - review & editing. **Xuelong Li:** Methodology, Writing - original draft, Writing - review & editing, Project administration.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant U1864204, 61773316, U1801262, and 61871470.

References

[1] Z. Kim, Robust lane detection and tracking in challenging scenarios..
[2] Y. Hou, Z. Ma, C. Liu, C.C. Loy, Learning lightweight lane detection cnns by self attention distillation, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 1013–1021.
[3] Z. Wang, W. Ren, Q. Qiu, Lanenet: Real-time lane detection networks for autonomous driving, arXiv preprint arXiv:1807.01726..
[4] H. Kong, J.-Y. Audibert, J. Ponce, General road detection from a single image, IEEE Transactions on Image Processing 19 (8) (2010) 2211–2220.
[5] Y. Yuan, Z. Jiang, Q. Wang, Video-based road detection via online structural learning, Neurocomputing 168 (2015) 336–347.
[6] Q. Wang, J. Fang, Y. Yuan, Adaptive road detection via context-aware label transfer, Neurocomputing 158 (2015) 174–183.
[7] Q. Wang, J. Gao, X. Li, Weakly supervised adversarial domain adaptation for semantic segmentation in urban scenes, IEEE Transactions on Image Processing 28 (9) (2019) 4376–4386.
[8] F. Pizzati, F. García, Enhanced free space detection in multiple lanes based on single cnn with scene identification, in: 2019 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2019, pp. 2536–2541..
[9] Y. Lu, Y. Yuan, Q. Wang, Forward vehicle collision warning based on quick camera calibration, in: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2018, pp. 2586–2590.
[10] T. Chen, K. Liu, Z. Wang, G. Deng, B. Chen, Vehicle forward collision warning algorithm based on road friction, Transportation Research Part D: Transport and Environment 66 (2019) 49–57.
[11] Y. Yuan, Z. Xiong, Q. Wang, Vssa-net: vertical spatial sequence attention network for traffic sign detection, IEEE Transactions on Image Processing 28 (7) (2019) 3423–3434.
[12] Y. Yuan, Z. Xiong, Q. Wang, An incremental framework for video-based traffic sign detection, tracking, and recognition, IEEE Transactions on Intelligent Transportation Systems 18 (7) (2016) 1918–1929.
[13] Q. Wang, J. Wan, Y. Yuan, Locality constraint distance metric learning for traffic congestion detection, Pattern Recognition 75 (2018) 272–281.
[14] M. Cao, J. Wang, Obstacle detection for autonomous driving vehicles with multi-lidar sensor fusion, Journal of Dynamic Systems, Measurement, and Control 142 (2)..
[15] X. Zhang, Y. Yuan, Q. Wang, Roi-wise reverse reweighting network for road marking detection, in: BMVC, 2018, p. 219..
[16] T.M. Hoang, S.H. Nam, K.R. Park, Enhanced detection and recognition of road markings based on adaptive region of interest and deep learning, IEEE Access 7 (2019) 109817–109832.
[17] Q. Wang, J. Gao, Y. Yuan, Embedding structured contour and location prior in siamesed fully convolutional networks for road detection, IEEE Transactions on Intelligent Transportation Systems 19 (1) (2017) 230–241.
[18] S. Lee, J. Hyun, Y.S. Kwon, J.H. Shim, B. Moon, Vision-sensor-based drivable area detection technique for environments with changes in road elevation and vegetation, Journal of Sensor Science and Technology 28 (2) (2019) 94–100.
[19] C.-C. Huang, S.-J. Wang, Y.-J. Change, T. Chen, A bayesian hierarchical detection framework for parking space detection, in: 2008 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2008, pp. 2097–2100.
[20] B. Suri, P.K.D. Pramanik, S. Taneja, An efficient parking solution for shopping malls using hybrid fog architecture, in: International Conference on Innovative Computing and Communications, Springer, 2020, pp. 313–320.
[21] M. Aly, Real time detection of lane markers in urban streets, in: 2008 IEEE Intelligent Vehicles Symposium, IEEE, 2008, pp. 7–12..
[22] M. Felisa, P. Zani, Robust monocular lane detection in urban environments, in: 2010 IEEE Intelligent Vehicles Symposium, IEEE, 2010, pp. 591–596..
[23] G. Kaur, A. Chhabra, Curved lane detection using improved hough transform and clahe in a multi-channel roi, International Journal of Computer Applications 122 (13)..
[24] Z. Kim, Robust lane detection and tracking in challenging scenarios, IEEE Transactions on Intelligent Transportation Systems 9 (1) (2008) 16–26, https://doi.org/10.1109/TITS.2007.908582, URL:http://ieeexplore.ieee.org/document/4459093/.
[25] J. Dai, Y. Li, K. He, J. Sun, R-fcn: Object detection via region-based fully convolutional networks, Advances in Neural Information Processing Systems (2016) 379–387.
[26] F. Pizzati, M. Allodi, A. Barrera, F. García, Lane detection and classification using cascaded cnns, arXiv preprint arXiv:1907.01294..
[27] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, Advances in Neural Information Processing Systems (2015) 91–99.
[28] M. Bertozzi, A. Broggi, Real-time lane and obstacle detection on the gold system, in: Proceedings of Conference on Intelligent Vehicles, IEEE, 1996, pp. 213–218..
[29] D. Liang, Y. Guo, S. Zhang, S.-H. Zhang, P. Hall, M. Zhang, S. Hu, Linenet: a zoomable cnn for crowdsourced high definition maps modeling in urban environments, arXiv preprint arXiv:1807.05696..
[30] N. Garnett, R. Cohen, T. Pe'er, R. Lahav, D. Levi, 3d-lanenet: end-to-end 3d multiple lane detection, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 2921–2930.
[31] P.-R. Chen, S.-Y. Lo, H.-M. Hang, S.-W. Chan, J.-J. Lin, Efficient road lane marking detection with deep learning, in: 2018 IEEE 23rd International Conference on Digital Signal Processing (DSP), IEEE, 2018, pp. 1–5.
[32] Q. Zou, H. Jiang, Q. Dai, Y. Yue, L. Chen, Q. Wang, Robust lane detection from continuous driving scenes using deep neural networks, IEEE Transactions on Vehicular Technology..
[33] S. Lee, J. Kim, J. Shin Yoon, S. Shin, O. Bailo, N. Kim, T.-H. Lee, H. Seok Hong, S.-H. Han, I. So Kweon, Vpgnet: Vanishing point guided network for lane and road marking detection and recognition, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1947–1955.
[34] U. Ozgunalp, R. Fan, X. Ai, N. Dahnoun, Multiple lane detection algorithm based on novel dense vanishing point estimation, IEEE Transactions on Intelligent Transportation Systems 18 (3) (2016) 621–632.
[35] W. Van Gansbeke, B. De Brabandere, D. Neven, M. Proesmans, L. Van Gool, End to end lane detection through differentiable least squares fitting, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2019.
[36] D. Neven, B. De Brabandere, S. Georgoulis, M. Proesmans, L. Van Gool, Towards end-to-end lane detection: an instance segmentation approach, in: 2018 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2018, pp. 286–291..
[37] R. Bellman, On the theory of dynamic programming, Proceedings of the National Academy of Sciences of the United States of America 38 (8) (1952) 716.
[38] C.J.C.H. Watkins, P. Dayan, Q-learning, Machine Learning 8 (3–4) (1992) 279–292.
[39] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, arXiv:1312.5602 [cs]..
[40] Y. Pang, L. Ye, X. Li, J. Pan, Incremental learning with saliency map for moving object detection, IEEE Transactions on Circuits and Systems for Video Technology 28 (3) (2016) 640–651.
[41] Y. Shen, R. Ji, C. Wang, X. Li, X. Li, Weakly supervised object detection via object-specific pixel gradient, IEEE Transactions on Neural Networks and Learning Systems 29 (12) (2018) 5960–5970.
[42] R. Quan, J. Han, D. Zhang, F. Nie, X. Qian, X. Li, Unsupervised salient object detection via inferring from imperfect saliency models, IEEE Transactions on Multimedia 20 (5) (2017) 1101–1112.
[43] L. Han, X. Li, Y. Dong, Salnet: Edge constraint based end-to-end model for salient object detection, in: Chinese Conference on Pattern Recognition and Computer Vision (PRCV), Springer, 2018, pp. 186–198.
[44] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587.
[45] R. Girshick, Fast r-cnn, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448..
[46] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556..
[47] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, A.M. Lopez, The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 3234–3243.
[48] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: A system for large-scale machine learning, in: 12th {USENIX} Symposium on Operating Systems Design and Implementation (OSDI 16), 2016, pp. 265–283..
[49] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch..
[50] The tusimple lane challenge, URL:https://github.com/TuSimple/tusimple-benchmark/tree/master/doc/lane_detection..
[51] X. Pan, J. Shi, P. Luo, X. Wang, X. Tang, Spatial as deep: Spatial cnn for traffic scene understanding, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[52] Y. Ko, J. Jun, D. Ko, M. Jeon, Key points estimation and point instance segmentation approach for lane detection, arXiv preprint arXiv:2002.06604..

Zhiyuan Zhao received the B.E. degree in electronics and information engineering from the Northwestern Polytechnical University, Xi'an 710072, Shaanxi, P. R. China, in 2018. He is currently pursuing the Ph.D. degree from Center for Optical Imagery Analysis and Learning, Northwestern Polytechnical University, Xi'an, China. His research interests include computer vision and pattern recognition.

Qi Wang (M'15-SM'15) received the B.E. degree in automation and the Ph.D. degree in pattern recognition and intelligent systems from the University of Science and Technology of China, Hefei, China, in 2005 and 2010, respectively. He is currently a Professor with the School of Computer Science, and with the Center for OPTical IMagery Analysis and Learning (OPTIMAL), Northwestern Polytechnical University, Xi'an, China. His research interests include computer vision and pattern recognition.

Xuelong Li (M'02-SM'07-F'12) is a full professor with the School of Computer Science and the Center for OPTical IMagery Analysis and Learning (OPTIMAL), Northwestern Polytechnical University, Xi'an 710072, Shaanxi, PR China.