Computational Visual Media
https://doi.org/10.1007/s41095-019-0160-1 Vol. 5, No. 4, December 2019, 417–428
Research Article

# SpinNet: Spinning convolutional network for lane boundary detection

**Ruochen Fan[1], Xuanrun Wang[1], Qibin Hou[2], Hanchao Liu[1], and Tai-Jiang Mu[1] (✉)**



**Abstract** In this paper, we propose a simple but effective framework for lane boundary detection, called SpinNet. Considering that cars or pedestrians often occlude lane boundaries and that the local features of lane boundaries are not distinctive, therefore, analyzing and collecting global context information is crucial for lane boundary detection. To this end, we design a novel spinning convolution layer and a brand-new lane parameterization branch in our network to detect lane boundaries from a global perspective. To extract features in narrow strip-shaped fields, we adopt strip-shaped convolutions with kernels which have $1 \times n$ or $n \times 1$ shape in the spinning convolution layer. To tackle the problem of that straight strip-shaped convolutions are only able to extract features in vertical or horizontal directions, we introduce the concept of feature map rotation to allow the convolutions to be applied in multiple directions so that more information can be collected concerning a whole lane boundary. Moreover, unlike most existing lane boundary detectors, which extract lane boundaries from segmentation masks, our lane boundary parameterization branch predicts a curve expression for the lane boundary for each pixel in the output feature map. And the network utilizes this information to predict the weights of the curve, to better form the final lane boundaries. Our framework is easy to implement and end-to-end trainable. Experiments show that our proposed SpinNet outperforms state-of-the-art methods.

**Keywords** object detection; lane boundary detection; autonomous driving; deep learning

1 Tsinghua University, Beijing, 100084, China. E-mail: R. Fan, frc16@mails.tsinghua.edu.cn; X. Wang, xuanrun-16@mails.tsinghua.edu.cn; H. Liu, liuhc17@mails.tsinghua.edu.cn; T.-J. Mu, mmmutj@gmail.com (✉).

2 Nankai University, Tianjin, 300350, China. E-mail: andrewhoux@gmail.com.

Manuscript received: 2019-12-20; accepted: 2019-12-24

## 1 Introduction

Object detection and segmentation are two of the most widely investigated areas of computer vision in recent decades. Generally speaking, advances in these domains are mostly driven by step changes made by classic baseline systems, such as Fast/Faster R-CNN [1, 2] and its follow-up architecture Mask R-CNN [3]. These methods provide a general and conceptional platform which is both flexible and robust.

Autonomous driving is a concept that has the ability to define future transportation. Fully autonomous cars are currently a major topic for computer vision and robotics research, both academically and industrially. The goal of autonomous driving requires full control of the car, which in turn necessitates understanding the surroundings of the car by gathering information from sensors attached to it. One of the most challenging subtasks for such perception is lane boundary detection for car positioning. Lane boundaries differ from typical objects in having stronger structural features but fewer appearance clues [4]. For instance, lane boundaries often have long, continuous shapes, and are located at the bottom of the scene. Because of the lack of apperance clues, lane boundaries may be incorrectly detected if they mainly rely on local features. Additionally, occlusion is a significant difficulty for lane boundary detection, as in many scenarios, some parts of lane boundaries may be occluded by other objects, especially vehicles.

Given the above issues, it is essential to utilize global information for lane boundary detection so that lane boundaries can be predicted in a holistic way. One natural idea is to attempt to enlarge the receptive field of the network. Spatial CNN (SCNN) [4], inspired by RNN [5], proposed a slice-by-slice convolution method that is able to transfer

information between neurons in the same layer. This architecture is also able to segment objects with long shapes. However, due to signal decay in the SCNN layer, distant information in the feature map is still hard to collect.

In order to gather and analyse features in large fields, in this paper, we present the idea of a spinning convolution and apply it within neural networks. The idea of a spinning convolution is to extract features along a narrow strip using a $1 \times n$ or $n \times 1$ convolution layer. "Spinning" means that long kernel convolution can run at arbitrary angle. Because of the long kernel used in a strip convolution, features covered by the kernels can be equally collected without signal fading caused by long distances. Generally speaking, a straightforward strip convolution operation is performed only along a vertical or horizontal direction; however, the directions of lane boundaries or other slender detection targets can vary. As rotating strip convolution kernels to lie along arbitrary directions is difficult, we propose to instead rotate the feature maps to increase the number of ways of collecting features. In this way, a vertical strip convolution with a large kernel can process features along any direction.

Furthermore, most existing methods consider lane boundary detection as a segmentation problem, that is, predicting which pixels are covered by lane boundaries, and the final parameterized lane boundary curves are produced by a hand-crafted algorithm which takes the segmentation maps predicted by the neural network as inputs. But this sort of method may be confused when the lane boundary mask is irregular in shape. For instance, they have problems processing masks with "holes". Instead, our approach carries out lane boundary detection by introducing a parameterization branch. To explicitly force the network to predict lane boundaries from a global perspective, our parameterization branch predicts the coefficients of lane boundary curves rather than segmentation masks. The final predicted lines are weighted combinations of the curves produced by the parameterization branch. To demonstrate the effectiveness of our proposed framework, we evaluate our results on the CULane dataset. We show that our SpinNet improves all prior lane boundary detection methods. To further verify the importance of our proposed spinning convolution and parameterisation branch, adequate ablation analysis is conducted. It reveals that both of the new operations contribute evidently to the overall performance.

In summary, our proposed approach contains the following contributions:

- A spinning convolution operation which can extract long range features running in arbitrary directions, without information decay.
- A novel method of determining parametric lane boundaries using information from all the pixels in a feature map.
- An end-to-end lane boundary detection framework, called SpinNet, which achieves state-of-the-art performance.

## 2 Related work

In this section, we briefly introduce some previous approaches closely related to our work.

### 2.1 Traditional methods

Traditional lane boundary detection algorithms are based on highly-specialised and hand-crafted low level features [6–10]. Methods exploiting structure tensors [11], color-based detectors [12], and bar filters [13] have achieved reasonable results; they are typically combined with Hough transforms [14] or Kalman filters [13]. When using such traditional techniques, post-processing is needed to weed out false detections, and to cluster lane boundary segments into final lines [7]. The weakness of traditional methods is that the models cannot handle complex circumstances. For example, occlusion, which is ever-present in real data [4, 15], presents difficulties for pattern-recognition-based methods: filters extracting image features fail. Traditional methods are insufficient to process the complex situations required in real driving conditions.

### 2.2 Neural-network-based methods

Deep learning has recently become mainstream in this domain due to its better results [16, 17]. The perception module in traditional approaches is replaced by a neural network, to overcome problems mentioned above. Huval et al. [18] proposed a deep learning based work, but lacked suitable large scale datasets for training. Spatial CNN [4] is now the state-of-the-art solution. It uses a specially designed architecture of convolution layers, enabling message

passing between pixels across rows and columns in a layer. This paper also provided a brand-new large scale dataset. However, the limited number of message passing directions resulted in problematic lane boundaries with unsatisfactory directions.

Methods predicting lane boundaries with the assistance of other information are now a new trend. For instance, some research [19, 20] has indicated that continuous driving scenes could provide information to constrain the predicted lane boundaries. However, these works place significant demands on dataset preparation. 3D vision has also been introduced to the lane boundary detection domain [21, 22] with reasonable results, but a major problem remains the costliness and rarity of both devices and datasets. Lee et al. [23] proposed a new way of predicting vanishing points to guide lane boundary prediction, giving a new approach to object-guided lane boundary segmentation.

### 2.3 Lane boundary parameterization

LaneNet [24] proposed a lane boundary detection framework that tackles the lane boundary detection task as an instance segmentation problem and also parametrizes the lane boundary by training an H-Net for view transformation. However, this work does not make good use of the predicted parameters, which are only used to generate the output, and are not capable of end-to-end training. Other works like Refs. [25, 26] utilized similar methods, fitting a parabola or spline for each lane boundary, but some lane boundaries are so irregular that such curves cannot provide a good fit.

### 2.4 Feature map deformation

A common method [27] is to rotate the convolution kernel, making the output of the neural network rotationally invariant. However, features of long strip shape are needed to detect the lane boundary, rather than rotational invariance, in our approach. Rotational invariance is typically of benefit in limited specific situations, like fingerprint recognition [28], galaxy morphology prediction [29], etc. When using the message passing technique of Spatial CNN [4], we have to find another solution.

## 3 SpinNet

In this section, we propose a framework for lane boundary detection and positioning, called SpinNet. SpinNet introduces two new ideas that are easy to implement.

### 3.1 Overall structure

SpinNet is an end-to-end trainable neural network, which employs VGGNet [30] as its backbone model. The structure is shown in Fig. 1. To ensure high resolution in the output feature map, we discard all fully-connected layers and the last two pooling layers (`pool4` and `pool5`) and change the dilation rates of all convolution layers in stage 5 from 1 to 2 to enlarge the receptive field. The proposed spinning convolution layer, whose purpose is to extract features along a long narrow field in a series of directions, is deployed after conv5 in VGGNet. After the spinning convolution layer, we also use a Spatial CNN layer, following Ref. [4], to further enlarge the receptive field. A fully-connected layer, followed by a softmax layer, is then used to predict a lane boundary presence vector $\boldsymbol{e}_i$, representing the probability that the $i$-th lane boundary is present in the image. After upsampling the feature map to the same size as the input image, a convolution layer predicts a map giving the probability of pixels being covered by lane boundaries, a coefficient matrix determining the parameters for the curves belonging to each pixel, and a coefficient confidence map.

### 3.2 Spinning convolution layer

Classical convolution kernels are usually square, e.g., 3×3, 5×5, etc. Using larger square kernels can enlarge the receptive field of a network. However, using huge square convolution kernels would lead to unacceptable computational costs in our specific task if we wish to cover a lane boundary. Furthermore, square convolution layers would have so many parameters in the convolution kernel, making them difficult to train. In practice, a lane boundary only occupies a small proportion of a square area, as they are narrow strips. A better approach to detecting these objects is to use strip-shaped narrow convolution kernels, e.g., $1 \times n$ or $n \times 1$.

To collect information about objects, Spatial CNN [4] borrowed ideas from RNN, passing messages inside its layers. However, its critical weakness is the message decay occurring in its message passing procedure.

Our specially-designed convolution kernels possess three clear advantages compared to square-shaped

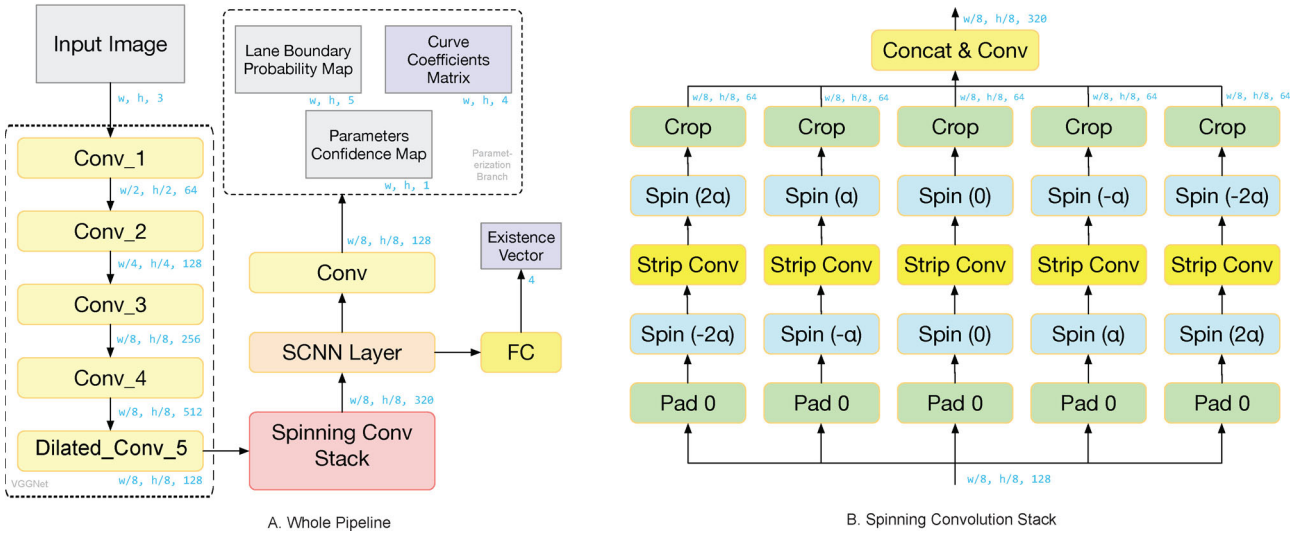



**Fig. 1** The network structure of SpinNet and an illustration of the proposed spinning convolution. (A) shows the whole network structure of SpinNet, in which Conv_1 to Dilated_Conv_5 are VGGNet backbone and the elements in dotted box and "Existence Vector" are the output of SpinNet.

convolution kernels and the SCNN technique. First, strip-shaped kernels provide a set of long narrow receptive fields in a series of directions. Second, these kernels require fewer network parameters: while a convolution layers with $3 \times 3$ and $9 \times 1$ kernels have the same number of network parameters, the latter has a much bigger receptive field than the former for the specific task of lane boundary detection. Third, our kernels are decay-free so that no information will be easily lost.

But lane boundaries are not just in these two specific directions, namely horizontal and vertical—they are arbitrary in direction. To overcome this problem, we have designed a method of rotating feature maps: the tilted lane boundaries become vertical or horizontal in one of the rotated feature maps, allowing them to be detected by vertical or horizontal convolution kernels. Although lane boundaries may not always be straight, some certain long strip kernels possessing different angles in variant positions of the feature map can fit a curve properly. Combining the above two concepts in a single layer gives our spinning convolution layer. Sample outputs of our spinning convolution are shown in Fig. 2.

The architecture of this spinning convolution layer is based on an original convolution layer, in which we replace the square convolution kernels by our specially designed strip kernels, and initially, we get the feature map generated by the backbone network. We first pad the feature map, and the rotation operation is then applied to the padded feature map, using a series of angles. Each different angle gives a new padded and rotated feature map. The strip convolution layer is then connected to each of these feature maps, which generates a series of new feature maps. We then perform an inverse rotation operation on each, and crop out the padding. Finally, we concatenate these outputs to form the final output of our spinning convolution layer. The process described above is shown in Fig. 1(B), in which $\alpha$ is a super parameter determining the spinning angle. We use five spinning convolutions with different $\alpha$ to form a spinning convolution stack.

Before our approach, there are many traditional approaches that aim to collect straight lane boundary information. However, there are several lane boundaries in one input image, and the boundaries may have some spatial information, or even correlations with each other. Thus, cropping the input to detect a lane boundary object is improper, for it will omit spatial information and correlations. Additionally, if we perform featuremap/input-picture rotation at the beginning of the network, we will not be able to finish the strip-shaped convolution, thus not be able to collect information on various directions explicitly. As we mentioned above, the directions of our strip-shaped kernels are fixed into 2, namely, horizontal and vertical. Thus, rotating feature maps is the very method to perform convolution using strip-shaped non-horizontal/vertical kernels, and this is

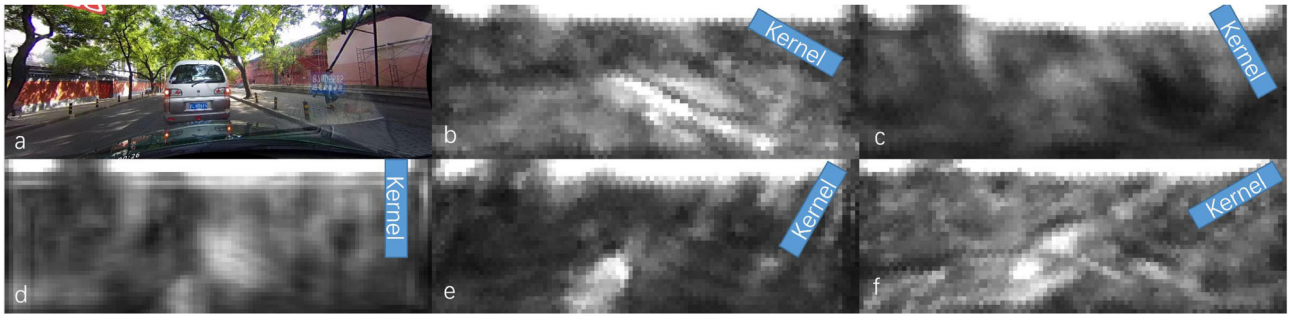



**Fig. 2** The result feature maps of spinning convolution. Given the strip-shaped kernels, performing convolution in traditional way can only use horizontal and vertical kernels. But in our purposed method, kernels can be designed to spin at arbitrary angles, as shown in the upper-right corner of each feature map. (a) is the original image input. (b–f) are the output feature maps of the spinning convolution stack. Each of the feature maps has 64 channels and the mean values of them are shown. The kernels are rotated with −60, −30, 0, 30, 60 degrees respectively. Obviously, a kernel with a certain rotation angle can collect more edge information of lane boundaries in that direction.

our original purpose to perform this rotation.

It seems that our method, rotating the feature maps, will receive a better result compared with these traditional techniques, since the spatial information and the correlation between lane boundaries are properly preserved, while the increase in calculation just arises a little.

### 3.3 Parameterization branch

Various works [24–26] consider lane boundary parameterization as a problem of fitting the lane boundary with a spline or parabola. However, all of the above methods share two common defects. Firstly, lane boundary parameterization is treated as a separate process from lane boundary semantic segmentation. In fact, the existing parameterization process is a hand-crafted algorithm only using the segmentation mask as input—it is unable to exploit the rich information in network feature maps. Instead, the spatial information from these two tasks can be exploited simultaneously inside the network, so that the tasks of lane boundary prediction and coefficient prediction can work together to provide mutual assistance, and the network will be an end-to-end task in this way. In our method, we combine these two tasks into one end-to-end network. Secondly, all previous works predict only a single set of coefficients, i.e., a single curve, for a lane boundary in an image. Commonly, lane boundaries are long smooth shapes, but in complex situations, their trajectories may be highly curved or irregular. Fitting a single curve with simple few global coefficients, typically a low order curve, will not give satisfactory results. In these situations, it is necessary to predict lane boundary curves locally, and then concatenate these local curves into the final complex lane boundary. We demonstrate later how we solve this problem.

We mitigate the first defect by combining parameterization and segmentation into one organic system: we generate two results using two branches attached to the same backbone. During training, gradient back-propagation influences the backbone through two paths: a semantic segmentation branch and a parameterization branch, allowing it to utilize the rich information included in the feature map for both tasks.

To alleviate the second defect, we force the network to predict not only local probability information, but global curve coefficients in each pixel of the feature map. By doing so, we explicitly make every pixel predict the curve from a global perspective.

After computing curve coefficients pixelwisely, the next step is to acquire global lane boundary information from this information. First, we know curve could be represented by a set of coefficients. Let these coefficients be $P_0, \cdots, P_n$. The curve

$$y = f(x) = \sum_{i=0}^{n} P_i x^i \tag{1}$$

then represents that lane boundary.

Let us first define the local curve. Unlike the global curve whose origin is at the upper left of the image, the origin of the local (relative to a certain pixel) curve is at the corresponding pixel in the image itself.

Overall, we first treat the whole lane boundary as a combination of several local curves, which is simple to generate from the dataset by polynomial fitting. As the dataset provides some key points on the lane boundary, we choose one key point and its neighbors, to determine the local curve coefficients near that key

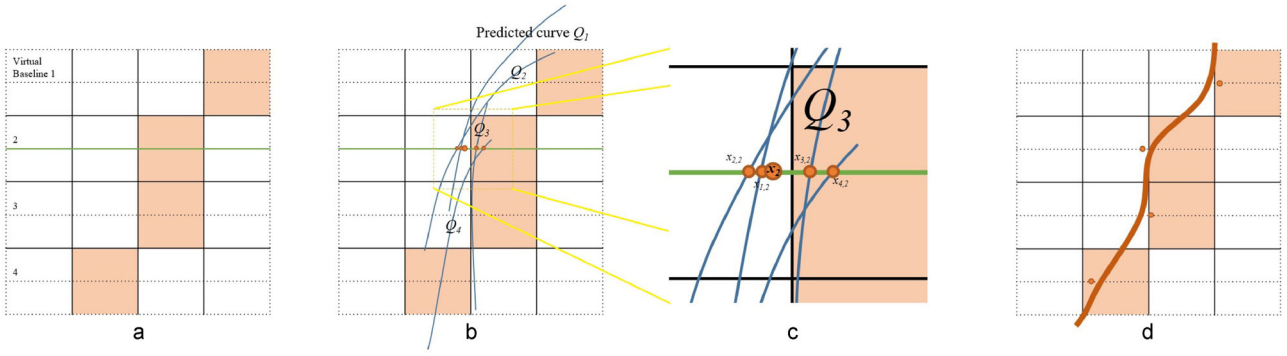



**Fig. 3** The curve aggregation pipeline. (a) shows a part of an input picture. There are $4 \times 4$ grids, and each grid indicates one pixel in the output feature maps and a square area in the original picture. The green horizontal line is the virtual baseline we are trying to solve the intersection point on. A grid is in orange only if the grid has the maximum $M_i$ on a horizontal vertical baseline. (b) shows the separate curves $Q_i$ generated from the four orange grids. These curves have intersections $x_{i,2}$ with the green horizontal baseline, which are shown in (c). These intersections are summed after being weighted by their confidence $M$ and the distance between their baselines and the green one, generating the final answer $x_2$ on the green baseline. Repeat this process, we get all the $x_j$s, and then we perform a simple post-process to draw the predicted lane boundary in orange shown in (d).

point by polynomial fitting. As for other key points, the method is similar. This is a simple pre-processing task by which we generate the ground truth of the lane boundaries coefficients. Then, we add a branch in parallel with the existing segmentation branch. Its aim is to generate the local curve coefficients at each pixel.

The method used to generate the set of coefficients of a lane boundary is regular, and could be performed by simply adding several dense layers after the output. But this method will let spatial information from the feature map be lost. Instead, for each pixel, the coefficients of the local lane boundary are easy to predict as this prediction only require adjacent information. Furthermore, the accuracy of local prediction is much better since one pixel only needs to predict the lane boundary nearby. We can thus use a feature-map-based pixelwise coefficient computation branch.

As the network only predicts local coefficients and the global coefficients are easy to acquire from the dataset, we need a simple global–local transformation. And this can be performed by

$$y - y_k = f_k(x - x_k) \tag{2}$$

where $f_k$ is the local curve function whose origin is at pixel $k$:$(x_k, y_k)$. Solving the equation above, we can easily get the new lane boundary curve coefficients $P_{k,i}$, with origin at pixel $k$. Preliminary experiments showed that using $n = 1$, i.e., fitting straight lines to lane boundaries, is not sufficient, but on the other hand, using $n > 2$ makes imperceptible further improvement. Thus we set $n = 2$, i.e., fit parabolas. In our approach, a parameterization branch is connected after the output layer of our backbone to produce a pixelwise coefficient matrix.

Some may believe there is a paradox that our output has passed the spinning pipeline, and thus only straight edge information was preserved, which is a misunderstanding. The shape of the convolution kernels only indicates the shape of the sampling points, and the ability to predict objects in other shapes will not be lost. In Section 4, we proved by experiments that strip-shaped convolution kernels have a better capability of detecting lane boundaries.

### 3.4 Curve aggregation

In the above introduction, we could get the local coefficient matrices generated by the neural network. That is, for each pixel of the feature map, there are a set of coefficients indicating the local lane boundary curve. To merge the set of local curves into final lane boundaries, we design the curve aggregation algorithm.

Curve aggregation is a weighted averaging process. For a point $(x, y_i)$ which lies in the virtual horizontal baseline $y = y_i$ in the final lane boundary, the target coordinate $x$ is the weighted average of the horizontal ordinates of the intersections of the local curves and the virtual baselines. The weights are positively correlated with the confidence of local curves predicted by the network and are negatively correlated with the distance between the pixel and the virtual baselines. Therefore, the more accurate the curve is, the more contribution the curve makes

to the final lane boundary. Besides, every pixel is mainly responsible for the section of the final lane boundary close to the pixel. Therefore, we proposed the following method to generate the key points of a lane boundary.

So far, the parameterization branch has provided: a probability map $\bar{M} \in \mathbb{R}^{4\times N_C}$, where $M_i$ is the probability that pixel $i$ is covered by the final lane boundary and $N_C$ is the number of pixels, a coefficient map $P \in \mathbb{R}^{N_C\times 3}$ where $P_i$ is the vector of coefficients of the curve at $i$, and a confidence map $F \in \mathbb{R}^{N_C}$, where $F_i$ is the confidence in the curve coefficients at $i$ as predicted by the network. Detailed defination and implementation of confidence map $F_i$ could be found in Section 3.5. We process the four lane boundaries separately, so from now on, we split $\bar{M}$ into four slices, process each slice $M \in \mathbb{R}^{N_C}$ separately. As we can exploit $\bar{M}$ to achieve the availability of reusing $F$ and $P$ in different lane boundaries, so it is unnecessary for $F$ or $P$ to possess four channels.

We first set $N_b$ virtual horizontal baselines evenly located in the image, at $y = j \times C,\ j = 0, \ldots, N_b - 1$ where $C$ is the baseline interval. The result of our algorithm is several points lying on the virtual horizontal baselines. For each baseline, we find the maximum $M_i$. The corresponding values of $i$ indicate the pixels most likely to be in the lane boundary, so we call them key-points from now on. However, we ignore any such points for which $M_i$ is below a threshold, as they are unlikely to be covered by a lane boundary. Let $K_i$ as the $i$-th key-point, with $y$ value $y_{K_i}$. For each key-point $K_i$, we can get the curve $Q_{K_i}$ assembled by $P_{K_i}$. These curves in general have several intersections with at least some of the virtual horizontal baselines. Let the $x$-coordinate of the intersection of $Q_{K_i}$ and $y = j\times C,\ j = 0, 1, \ldots, N_b-1$ be $x_{K_i,j}$. The coordinate $x_j$ of the result for $y = j\times C$ is given by

$$x_j = \frac{\sum_i x_{K_i,j} F_{K_i} \exp(-\alpha|jC - y_{K_i}|)}{\sum_i F_{K_i} \exp(-\alpha|jC - y_{K_i}|)} \tag{3}$$

where $\alpha$ is a constant. Thus, $(x_j, jC)$ is one of our result points. After finding such points for all virtual horizontal baselines, the set of $(x_j, jC)$ is then the final output.

### 3.5 Loss function

The loss function $L$ should have the ability to refine the probability maps and the coefficient maps simultaneously. To do so, it must be a weighted sum of multiple parts:

$$Loss = C_{\text{bin}} Loss_{\text{bin}} + C_{\text{ex}} Loss_{\text{ex}} + C_{\text{par}} Loss_{\text{par}} \tag{4}$$

where the weighting coefficients are hyperparameters which must be tuned.

The first part of the loss is the lane boundary object binary segmentation loss $Loss_{\text{bin}}$. To compute it, we can easily calculate a softmax cross-entropy between prediction and the ground truth in the semantic segmentation task. The second part of the loss is the lane boundary existence loss $Loss_{\text{ex}}$. This essentially concerns a binary classification task, so again a cross-entropy computation suffices. The third part of the loss is the lane boundary coefficients loss $Loss_{\text{par}}$. In one step of forward propagation, after finding $\bar{M}$, we are able to categorize each pixel. Thus, if a pixel $k$ belongs to *lane boundary* $j$, its coefficients $P_i$ should be as close to the ground truth $P_{i,\text{gt},j,k}$ as possible. Here, $P_{i,\text{gt},j,k}$ stands for the ground truth coefficients $P_i$ for lane $j$, transformed to origin at $k$. We may then calculate the loss as follows:

$$Loss_{\text{par},P_i} = \sum_{j=1}^{4} \sum_{k\in lane_j} |P_{i,\text{gt},j,k} - P_{i,j,k}| \tag{5}$$

where $|\cdot|$ denotes the $L_1$ loss. As we treat the curves as parabolas, in the following we simply replace $P_2$ by $a$, $P_1$ by $b$, and $P_0$ by $c$.

There are two tasks at each pixel to perform, to predict the semantic segmentation of the lane boundary, and to predict the confidence in the coefficients of the lane boundary passing through it. While it seems obvious that these two tasks are connected, they do not in fact constitute a sound causal relationship, as pixels with precise coefficients prediction may lie outside any lane boundary. So, we need a measurement of the confidence of coefficients accuracy as the contribution factor mentioned in the previous section. At pixel $k$, the ground truth value $F_{\text{gt},j,k}$ for lane boundary $j$ and origin at pixel $k$ is defined as follows:

$$F_{\text{gt},j,k} = \exp(-|b_{\text{gt},j,k} - b_k| - |c_{\text{gt},j,k} - c_k|) \tag{6}$$

Our experiments indicate that $b$ and $c$ are essential to lane boundary prediction, and the loss for $b$ and $c$ is key to determining the confidence.

The loss function for $F$ is easy to compute:

$$Loss_{\text{par},F} = \sum_{j=1}^{4} \sum_{k\in lane_j} |F_{\text{gt},j,k} - F_k| \tag{7}$$

We can then get the total loss:

$$\begin{aligned} Loss_{\text{par}} = \; & C_a Loss_{\text{par},a} + C_b Loss_{\text{par},b} \\ & + C_c Loss_{\text{par},c} + C_F Loss_{\text{par},F} \end{aligned} \tag{8}$$

where $C$s are the hyperparameters to finetune.

## 4 Experiments

In this section, we show the efficacy of SpinNet on the large-scale CULane dataset. Comparisons with state-of-the-art methods show that our proposed framework outperforms all prior lane detection methods. To further analyze the effectiveness of each component of SpinNet, we have performed a series of ablation experiments, and the effects of some key hyper-parameters also are shown.

### 4.1 Implementation details

#### *4.1.1 Training and testing*

Images provided by datasets only have points located evenly on each of the lane boundaries. To generate ground truths for lane boundary coefficients, we performed parabola fitting to these point sets.

During training, the original image, binary masks of corresponding lane boundaries, lane boundary existence vectors, and curve coefficients sets are fed into our end-to-end network. During testing, in order to judge if a lane boundary is correctly detected, we regard the lane boundaries as polyline segments with a width of 30 pixels, and then calculate the intersection-over-union (IoU) between the ground truth and the prediction. Predictions whose IoUs are greater than a certain threshold (0.5) are regarded as true positives (TP). This enables us to assess our method using the F1 measure.

#### *4.1.2 Hyper-parameters*

Our proposed network was implemented on TensorFlow [31]. The input images were not augmented. The hyper-parameters are set as follows: momentum (0.9), learning rate (0.05), weight decay (0.0001). We trained our network on a single Nvidia GTX1080 Ti GPU for 180k iterations.

### 4.2 Ablation study and setup

#### *4.2.1 The effect of spinning convolution and parameterization branch*

To evaluate the effectiveness of spinning convolution, we replaced this operation by a traditional convolution stack. The results in line "w/o Spin-Conv" of Table 1 indicate that spinning convolution contributes 1.1 percentage points to the overall performance.

To evaluate the effectiveness of our lane boundary parameterization branch, we disabled local curve prediction in our model and extracted curves from lane boundary segmentation masks, following existing methods such as SCNN [4]. The results in line "w/o Parameterization" of Table 1 indicate that the parameterization branch benefits evidently to lane boundary detection performance.

We also disabled both operations; the inferior experimental results further demonstrate the effectiveness of our proposed operations.

#### *4.2.2 Spinning angles*

To better understand our spinning convolution layer and suggest the best way to use it, we determined how choice of spinning angles affected the results. We used 3, 5, 7 simultaneous spinning convolutions in each experiment, with angles chosen as in Table 2. The experiment indicates that rotation angles affect lane boundary detection performance. Only a small spinning angle cannot give us the best result. Moreover, too many or too few directions of rotations will also harm the performance. The experiments show that $(\pm 60°, \pm 30°, 0°)$ is the optimal angle settings we have found, implying that making the spinning angles evenly distributed is a good strategy.

**Table 1** Ablation study of our proposed SpinNet. "w/o Spin-Conv" represents the performance when spinning convolution is replaced with traditional convolution stack. "w/o Parameterization" means disabling parameterization branch and generates lane boundary prediction results from segmentation mask. The experiments show that SpinNet achieves best performance when both proposed branches are used

| Strategy | Precision | Recall | F1-score |
|---|---|---|---|
| w/o Spin-Conv | 0.856 | **0.638** | 0.731 |
| w/o Parameterization | 0.853 | 0.635 | 0.728 |
| w/o both | 0.846 | 0.628 | 0.721 |
| w/ all branches | **0.879** | 0.628 | **0.742** |

**Table 2** Performance of SpinNet when using different rotation angles in spinning convolution. $(\pm 20°, \pm 10°, 0°)$ means that there are five sub-branches in spinning convolution stack, with rotation angles from $-20°$ to $20°$

| Angles | Precision | Recall | F1-score |
|---|---|---|---|
| $(\pm 20°, \pm 10°, 0°)$ | 0.878 | 0.632 | 0.735 |
| $(\pm 40°, \pm 20°, 0°)$ | **0.878** | 0.635 | 0.737 |
| $(\pm 60°, \pm 30°, 0°)$ | 0.852 | **0.657** | **0.742** |
| $(\pm 80°, \pm 40°, 0°)$ | 0.869 | 0.641 | 0.738 |
| $(\pm 45°, 0°)$ | 0.847 | 0.615 | 0.713 |
| $(\pm 60°, \pm 40°, \pm 20°, 0°)$ | 0.850 | 0.651 | 0.737 |

#### *4.2.3 Spinning convolution kernel size*

A kernel of size $1 \times n$ is used in the spinning convolution, and its size $n$ controls the range of the receptive field of this convolution layer. In theory, a large receptive field usually benefits performance, but requires more parameters which may increase training difficulty. We performed an experiment to find the optimal size of spinning convolution kernel. The results are shown in Table 3. The results reveal that a kernel of size $12 \times 1$ achieves the best result. A shorter or longer kernel harms the overall performance.

**Table 3** Performance of SpinNet when using different size of strip-shaped large kernel in spinning convolution. The size of the kernel we use is $1 \times n$. The experiment reveals that too long or too short kernels harm the performance, and a $1 \times 12$ kernel is most appropriate for our task

| No. | Kernel size ($n$) | Precision | Recall | F1-score |
|---|---|---|---|---|
| 1 | 7 | 0.839 | 0.650 | 0.733 |
| 2 | 9 | 0.851 | 0.656 | 0.740 |
| 3 | 12 | 0.852 | **0.657** | **0.742** |
| 4 | 15 | 0.847 | 0.652 | 0.737 |
| 5 | 18 | **0.879** | 0.628 | 0.732 |

### 4.3 Comparison with state-of-the-art

We compared our proposed method with existing lane boundary detection methods using the CULane dataset. Table 4 lists the results; performance is measured by F1-score. This table also gives efficacy comparison in some particular scenario. It is clear that our new approach, SpinNet, achieves the best overall results, improving on the baseline result presented in Zhang et al. [34] or LineNet [35] by 1.1 percentage.

In addition to quantitative evaluation, some visualization results are shown in Fig. 4. We label ground truth lines in green, true positive predictions in blue, and incorrect predictions in red. Our SpinNet is able to handle crowded road with evident occlusion. For failure cases, we find that it is difficult to detect lane boundaries fully occluded by cars.

## 5 Conclusions

This paper has presented a lane boundary detection framework, including a novel spinning convolution

**Table 4** Lane boundary results on CULane dataset (F1-measure). The columns from "Normal" to "Curve" show the effectiveness comparison in some particular scenario. The column "Total" shows the overall performance in the whole test set of CULane dataset, indicating that our SpinNet achieves new state-of-the-art result

| Method | Normal | Crowded | Night | No line | Shadow | Arrow | Dazzle light | Curve | Total |
|---|---|---|---|---|---|---|---|---|---|
| SegNet [32] | 0.792 | 0.617 | 0.496 | 0.145 | 0.294 | 0.710 | 0.389 | 0.456 | 0.572 |
| SegNet-Ego-Lane [33] | 0.754 | 0.620 | 0.578 | 0.177 | 0.310 | 0.714 | 0.476 | 0.430 | 0.584 |
| SCNN [4] | 0.883 | 0.753 | 0.686 | 0.365 | 0.593 | 0.821 | 0.531 | 0.594 | 0.720 |
| Zhang et al. [34] | 0.897 | **0.765** | **0.687** | 0.351 | 0.655 | 0.822 | **0.674** | **0.632** | 0.731 |
| LineNet [35] | — | — | — | — | — | — | — | — | 0.731 |
| **SpinNet (ours)** | **0.905** | 0.717 | 0.684 | **0.432** | **0.729** | **0.850** | 0.620 | 0.507 | **0.742** |

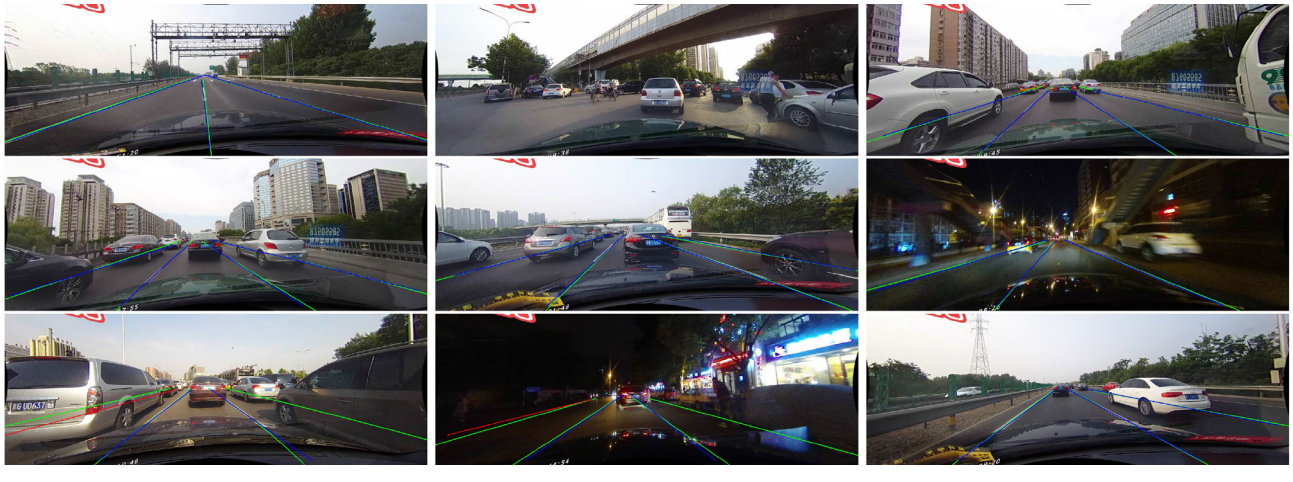


**Fig. 4** Selected examples produced by our SpinNet. Green lines are the ground truths. Blue lines are the predicted true positive line boundaries, while red lines are the incorrect prediction results. From this figure, we can see that our framework works well even in crowded road with obvious occlusion and scenes with low contrast.

layer and a new lane boundary parameterization branch. The spinning convolution layer with strip kernel is able to extract features in long narrow fields along a series of directions. The parameterization branch predicts a whole curve for each pixel in the output feature map. Experiments show that both operations improve the performance of a lane detection network, and that the proposed framework outperforms prior methods on the CULane dataset.

## Acknowledgements

This work was supported by the National Natural Science Foundation of China (Project No. 61572264), Research Grant of Beijing Higher Institution Engineering Research Center, and Tsinghua–Tencent Joint Laboratory for Internet Innovation Technology.

## References

[1] Girshick, R. Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, 1440–1448, 2015.

[2] Ren, S.; He, K.; Girshick, R.; Sun, J. Faster RCNN: Towards real-time object detection with region proposal networks. In: Proceedings of the Advances in Neural Information Processing Systems 28, 91–99, 2015.

[3] He, K. M.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, 2961–2969, 2017.

[4] Pan, X.; Shi, J.; Luo, P.; Wang, X.; Tang, X. Spatial as deep: Spatial CNN for traffic scene understanding. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, 7276–7283, 2018.

[5] Lipton, Z. C.; Berkowitz, J.; Elkan, C. A critical review of recurrent neural networks for sequence learning. *arXiv preprint* arXiv:1506.00019, 2015.

[6] Aly, M. Real time detection of lane markers in urban streets. In: Proceedings of the IEEE Intelligent Vehicles Symposium, 7–12, 2008.

[7] Bar Hillel, A.; Lerner, R.; Levi, D.; Raz, G. Recent progress in road and lane detection: A survey. *Machine Vision and Applications* Vol. 25, No. 3, 727–745, 2014.

[8] Son, J.; Yoo, H.; Kim, S.; Sohn, K. Real-time illumination invariant lane detection for lane departure warning system. *Expert Systems with Applications* Vol. 42, No. 4, 1816–1824, 2015.

[9] Jung, S.; Youn, J.; Sull, S. Efficient lane detection based on spatiotemporal images. *IEEE Transactions on Intelligent Transportation Systems* Vol. 17, No. 1, 289–295, 2016.

[10] Borkar, A.; Hayes, M.; Smith, M. T. A novel lane detection system with efficient ground truth generation. *IEEE Transactions on Intelligent Transportation Systems* Vol. 13, No. 1, 365–374, 2012.

[11] Loose, H.; Franke, U.; Stiller, C. Kalman Particle Filter for lane recognition on rural roads. In: Proceedings of the IEEE Intelligent Vehicles Symposium, 60–65, 2009.

[12] Chiu, K. Y.; Lin, S. F. Lane detection using color-based segmentation. In: Proceedings of the IEEE Intelligent Vehicles Symposium, 706–711, 2005.

[13] Teng, Z.; Kim, J.-H.; Kang, D.-J. Real-time lane detection by using multiple cues. In: Proceedings of the International Conference on Control, Automation and Systems, 2334–2337, 2010.

[14] Liu, G. L.; Wörgötter, F.; Markelić, I. Combining statistical Hough transform and particle filter for robust lane detection and tracking. In: Proceedings of the IEEE Intelligent Vehicles Symposium, 993–997, 2010.

[15] TuSimple. Tusimple dataset. Available at http://cvpr2017.tusimple.ai/.

[16] Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognitionm, 3431–3440, 2015.

[17] Chen, L. C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A. L. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 40, No. 4, 834–848, 2018.

[18] Huval, B.; Wang, T.; Tandon, S.; Kiske, J.; Song, W.; Pazhayampallil, J.; Andriluka, M.; Rajpurkar, P.; Migimatsu, T.; Cheng-Yue, R. et al. An empirical evaluation of deep learning on highway driving. *arXiv preprint* arXiv:1504.01716, 2015.

[19] Zou, Q.; Jiang, H. W.; Dai, Q. Y.; Yue, Y. H.; Chen, L.; Wang, Q. Robust lane detection from continuous driving scenes using deep neural networks. *IEEE Transactions on Vehicular Technology* doi: 10.1109/TVT.2019.2949603, 2019.

[20] Zhang, W.; Mahale, T. End to end video segmentation for driving: Lane detection for autonomous car. *arXiv preprint* arXiv:1812.05914 2018.

[21] Quach, C. H.; Tran, V. L.; Nguyen, D. H.; Nguyen, V. T.; Pham, M. T.; Phung, M. D. Real-time lane marker detection using template matching with RGB-D camera. In: Proceedings of the 2nd International

Conference on Recent Advances in Signal Processing, Telecommunications & Computing, 152–157, 2018.

[22] Garnett, N.; Cohen, R.; Pe'er, T.; Lahav, R.; Levi, D. 3D-LaneNet: End-to-end 3D multiple lane detection. In: Proceedings of the IEEE International Conference on Computer Vision, 2921–2930, 2019.

[23] Lee, S.; Kim, J.; Yoon, J. S.; Shin, S.; Bailo, O.; Kim, N.; Lee, T.-H.; Hong, H. S.; Han, S.-H.; Kweon, I. S. VPGNet: Vanishing point guided network for lane and road marking detection and recognition. In: Proceedings of the IEEE International Conference on Computer Vision, 1947–1955, 2017.

[24] Neven, D.; De Brabandere, B.; Georgoulis, S.; Proesmans, M.; Gool, L. V. Towards end-to-end lane detection: An instance segmentation approach. In: Proceedings of the IEEE Intelligent Vehicles Symposium (IV), 286–291, 2018.

[25] Van Gansbeke, W.; De Brabandere, B.; Neven, D.; Proesmans, M.; Van Gool, L. End-to-end lane detection through differentiable least-squares fitting. In: Proceedings of the IEEE International Conference on Computer Vision, 2019.

[26] Chen, P. R.; Lo, S. Y.; Hang, H. M.; Chan, S. W.; Lin, J. J. Efficient road lane marking detection with deep learning. In: Proceedings of the IEEE 23rd International Conference on Digital Signal Processing, 1–5, 2018.

[27] Worrall, D. E.; Garbin, S. J.; Turmukhambetov, D.; Brostow, G. J. Harmonic networks: Deep translation and rotation equivariance. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 5028–5037, 2017.

[28] Ouyang, Z. Y.; Feng, J. J.; Su, F.; Cai, A. N. Fingerprint matching with rotation-descriptor texture features. In: Proceedings of the 18th International Conference on Pattern Recognition, 417–420, 2006.

[29] Dieleman, S.; Willett, K. W.; Dambre, J. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society* Vol. 450, No. 2, 1441–1459, 2015.

[30] Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint* arXiv:1409.1556, 2014.

[31] Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M. et al. TensorFlow: A system for large-scale machine learning. In: Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, 265–283, 2016.

[32] Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 39, No. 12, 2481–2495, 2017.

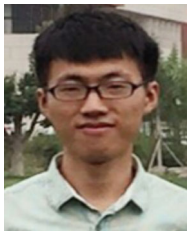
[33] Kim, J.; Park, C. End-to-end ego lane estimation based on sequential transfer learning for self-driving cars. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 30–38, 2017.

[34] Zhang, J.; Xu, Y.; Ni, B. B.; Duan, Z. Y. Geometric constrained joint lane segmentation and lane boundary detection. In: *Computer Vision – ECCV 2018. Lecture Notes in Computer Science, Vol. 11205*. Ferrari, V.; Hebert, M.; Sminchisescu, C.; Weiss, Y. Eds. Springer International Publishing, 502–518, 2018.

[35] Liang, D.; Guo, Y.; Zhang, S.; Zhang, S.-H.; Hall, P.; Zhang, M.; Hu, S. LineNet: A zoomable CNN for crowdsourced high definition maps modeling in urban environments. *arXiv preprint* arXiv:1807.05696, 2018.

**Ruochen Fan** is a master student at Computer Science Department, Tsinghua University under the supervision of Prof. Shi-Min Hu. He currently focuses on perception system for autonomous driving, especially for point cloud segmentation and RGB detection. Before that, He did some research on saliency detection and weakly-supervised segmentation.

**Xuanrun Wang** is currently a senior undergraduate student in the Department of Computer Science and Technology at Tsinghua University. His research interest is computer vision.



**Qibin Hou** is at present a third-year Ph.D. student under Prof. Ming-Ming Cheng's supervision. Before joining in the media group at Nankai University, he was a machine learning engineer in Baidu. His research interests include low-level vision, deep learning, and multimedia applications.

**Hanchao Liu** is currently a master student in the Department of Computer Science and Technology, Tsinghua University. His research interests include image/video processing and computer vision.

**Tai-Jiang Mu** is currently a assistant researcher in the Department of Computer Science and Technology, Tsinghua University, where he received his Ph.D. and B.S. degrees in 2016 and 2011, respectively. His research area is computer graphics, mainly focusing on stereoscopic image and video processing, and stereoscopic perception.