Polynomial Regression Network for Variable-Number Lane Detection

Bingke Wang, Zilei Wang(✉), and Yixin Zhang

University of Science and Technology of China, Hefei, China
{wbkup,zhyx12}@mail.ustc.edu.cn, zlwang@ustc.edu.cn

Abstract. Lane detection is a fundamental yet challenging task in autonomous driving and intelligent traffic systems due to perspective projection and occlusion. Most of previous methods utilize semantic segmentation to identify the regions of traffic lanes in an image, and then adopt some curve-fitting method to reconstruct the lanes. In this work, we propose to use polynomial curves to represent traffic lanes and then propose a novel polynomial regression network (PRNet) to directly predict them, where semantic segmentation is not involved. Specifically, PRNet consists of one major branch and two auxiliary branches: (1) polynomial regression to estimate the polynomial coefficients of lanes, (2) initialization classification to detect the initial retrieval point of each lane, and (3) height regression to determine the ending point of each lane. Through the cooperation of three branches, PRNet can detect variable-number of lanes and is highly effective and efficient. We experimentally evaluate the proposed PRNet on two popular benchmark datasets: TuSimple and CULane. The results show that our method significantly outperforms the previous state-of-the-art methods in terms of both accuracy and speed.

Keywords: Lane detection · Polynomial curve · Deep neural network · Polynomial regression

1 Introduction

The past decade has witnessed the great progress of autonomous driving and intelligent transport systems in academia and industry. In these systems, lane detection is one of the fundamental tasks to fully understand the traffic environment, in which the road lanes represent some kind of traffic rules made by human being. Currently, lane detection is still challenging due to the diversity of lane appearance (*e.g.*, colors, line types) and complexity of traffic environmental conditions (*e.g.*, various weathers, lights, and shadows). For example, it is quite difficult to detect lanes in crowded traffic conditions even for human being due to heavy occlusion by vehicles.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-58523-5_42) contains supplementary material, which is available to authorized users.

A. Vedaldi et al. (Eds.): ECCV 2020, LNCS 12363, pp. 719–734, 2020.
https://doi.org/10.1007/978-3-030-58523-5_42

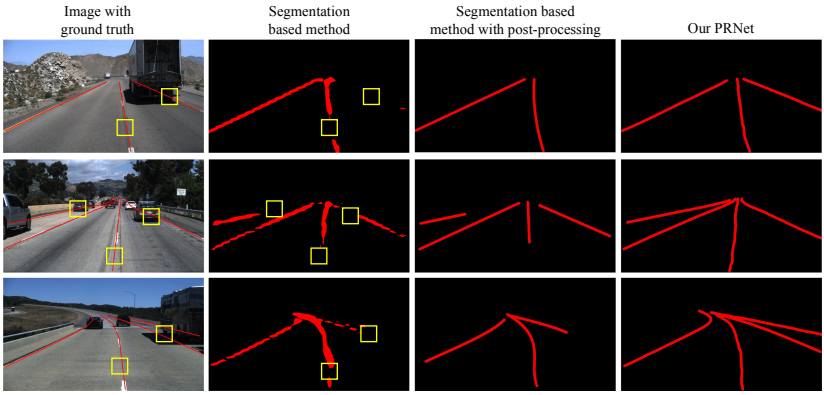

Fig. 1. Comparison of lane detection results between the segmentation based method and our PRNet. Typically, the segmentation based method would suffer the noisy points and intermittent lane segments which need post-processing methods to handle, while PRNet can avoid them due to polynomial representation of traffic lanes.

Many lane detection methods have been proposed to tackle these challenges. Traditional methods [1,2,9,14] usually utilize hand-crafted low-level features to detect the edges or colors, which cannot handle complex conditions. In recent years, some works try to employ the popular deep neural networks to solve this problem [6,10,12,16–18]. Typically, most of these methods treat lane detection as a semantic segmentation task, where each image pixel is classified if it belongs to one of lanes. However, the segmentation based methods often suffer discontinuous and noisy detection results due to thinness of traffic lanes, as shown in Fig. 1. To alleviate this issue, these methods usually use some curve-fitting strategy to filter the noise points [12,16] or cluster the intermittent lane segments [12]. Here we argue that it is unnecessary to explicitly produce semantic segmentation maps for lane detection because such a task essentially targets to get the curves of traffic lanes in an image.

In this paper, we propose to use polynomial curves to represent the traffic lanes and a novel polynomial regression network (PRNet) to directly predict them, in which no semantic segmentation is performed. The key idea of PRNet is to use a piecewise curve to represent a traffic lane rather than a set of image pixels in the previous works. Following this idea, we decompose lane detection into one major subtask and two auxiliary subtasks, *i.e.*, polynomial regression, initialization classification, and height regression, as shown in Fig. 2. Here polynomial regression is used to estimate the polynomial coefficients of lane segments in an image. Initialization classification is used to detect the point to retrieve the initial polynomial coefficients of each lane. Height regression is used to predict the height of ending point for each lane, which together with the estimated polynomial curves determines the ending point of a traffic lane. In this work, we particularly define the initial retrieval point of one lane as the lane point closest to the bottom boundary of image. Evidently, the initial retrieval points

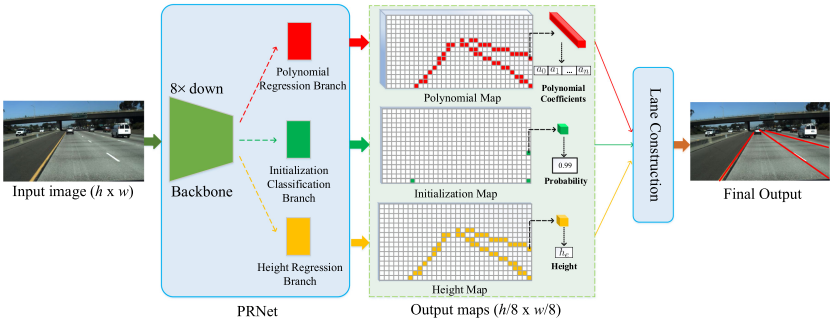

Fig. 2. Illustration of our proposed PRNet. The input image is first transformed into low-resolution feature maps by a backbone network. Then three branches, *i.e.*, polynomial regression, initialization classification, and height regression, take the feature maps as input to predict the polynomial curves of traffic lanes. Finally, the lanes are constructed by fusing the information from three branches. Best viewed in color. (Color figure online)

of different lanes in an image are usually far apart from each other according to the traffic rules.

Different from the segmentation based methods that assign the pixels of different lanes different semantic labels, PRNet identifies a lane by detecting its initial retrieval point. Thus PRNet can detect variable-number lanes, like object detection. Moreover, the curve representations of traffic lanes are inherently smooth, and thus no extra post-processing is needed in constructing lane curves.

The contributions of this work are summarized as:

- We propose to use polynomial curves to represent a traffic lane in images, and then formulate lane detection into three subtasks, *i.e.*, polynomial regression, initialization classification, and height regression.
- We propose a novel polynomial regression network (PRNet) to efficiently perform the three subtasks by three branches, in which low-resolution feature maps having global receptive field at the input images are shared.
- We experimentally verify the effectiveness of our proposed PRNet, and the results on both TuSimple and CULane well demonstrate the superiority of our method to other state-of-the-art methods.

2 Related Work

2.1 Traditional Methods

Traditional methods generally use hand-crafted features to detect traffic lanes. For example, the Gaussian filter [1], Steerable filter [14,15], and Gabor filter [26] are adopted to extract the edge features for lane detection. The color features [9] and histogram based features [7] are also exploited to achieve more accurate lane detection results. For these methods, Hough Transformation (HT) [2] is often employed to perform the lane fitting as a post-processing technique. In practice,

however, the traditional methods would suffer serious performance degradation when complex traffic conditions are presented [16].

2.2 CNN-Based Methods

Deep convolution neural networks [8,11,21,22] have shown powerful capabilities in various visual tasks. In particular, many CNN-based lane detection methods have been proposed in the past few years. Here we divide them into two broad categories: segmentation based methods and non-segmentation based methods.

Segmentation Based Methods. VPGNet [12] proposes a multi-task network to jointly handle lane and road marking detection under the guidance of vanishing point. Spatial CNN (SCNN) [17] generalizes the traditional deep layer-by-layer convolutions to slice-by-slice convolutions within feature maps, which contribute to detecting long continuous slender structure or large objects. LaneNet [16] proposes to formulate lane detection into an instance segmentation problem and then predict a perspective transformation matrix for better fitting lanes. Embedding-loss GAN (EL-GAN) [6] introduces a GAN framework to make the produced semantic segmentation maps more realistic or better structure-preserving. Self Attention Distillation (SAD) [10] allows a model to learn from itself and gains substantial improvement without any additional supervision or labels. Different from these methods, our proposed method in this work does not involve semantic segmentation.

Non-segmentation Based Methods. Inspired by Faster RCNN [19], Li *et al.*proposed Line-CNN that utilizes line proposals as references to locate traffic curves [13]. Line-CNN need generate a large number of line proposals to achieve good performance. FastDraw [18] proposes to estimate the joint distribution of neighboring pixels belonging to the same lane and draw the lane in an iterative way, in which a binary segmentation map is needed as guidance. 3D-LaneNet [5] directly predicts the 3D layout of lanes in a road scene from a single image through an end-to-end network, which uses the anchor-based lane representation similar to Line-CNN. [24] proposes to estimate lane curvature parameters by solving a weighted least-squares problem in-network, whose weights are generated by a deep network conditioned on the input image. However, the method needs to generate the segmentation-like weight map for each lane separately, and thus can only detect a fixed number of lanes. In addition, the involved huge matrix operation for solving the weighted least-squares problem is time-consuming.

Most of previous methods need to perform some post-processing method to obtain the final traffic curves in practice. For the segmentation based methods, the clustering method (*e.g.*, DBSCAN [3]) or line fitting method (*e.g.*, RANSAC [4]) is often required. In addition, Line-CNN needs to employ NMS [19] to eliminate the redundant line proposals. Evidently, the post-processing in these methods would involve extra computational cost. On the contrary, our proposed network can directly produce the traffic curves and the number of lanes in an image is not required to be fixed.

3 Our Approach

Traffic lanes belong to the man-made objects that are used to specify the traffic rules. In general, lanes are drawn on the roads with a shape of line or curve. So we propose to use the intrinsic curves to represent the traffic lanes in images, and it is expected that such curves can be directly predicted by some network. Following this idea, we particularly propose polynomial curves to represent traffic lanes.

Due to the perspective projection, a lane in images may present a complicated shape that is hard to be accurately represented by one single polynomial curve. To tackle the issue, we propose to use the piecewise polynomials with different coefficients to represent one lane curve. As a result, each lane in an image can be represented by

$$\begin{cases} [a_0^1, a_1^1, \cdots, a_{n-1}^1, a_n^1] & if\ h_r - H \leqslant x \leqslant h_r \\ [a_0^2, a_1^2, \cdots, a_{n-1}^2, a_n^2] & if\ h_r - 2H \leqslant x < h_r - H \\ \cdots & \\ [a_0^k, a_1^k, \cdots, a_{n-1}^k, a_n^k] & if\ h_e \leqslant x < h_r - (k-1)H \end{cases} \tag{1}$$

where n is the polynomial order, k is the number of polynomials, and $\{a_i^j\}_{i=0}^n$ are the polynomial coefficients of the j^{th} polynomial piece. In addition, h_r is the height of the initial retrieval point, H is a hyper-parameter that denotes the height of each polynomial piece, and h_e is the height of the ending point. Obviously, we have $k = \lceil \frac{h_r - h_e}{H} \rceil$. Different from the splines to represent lanes by identifying the control points, our proposed piecewise polynomials target to get the polynomial coefficients directly.

According to the above formulation, our task turns to predict the polynomial coefficients $\{a_i^j\}_{i=0}^n$ for each lane segment. Here our main challenges lie in how to model all polynomial pieces in an image so that each lane curve can be effectively constructed, and how to design efficient implementation. To this end, we propose a novel Polynomial Regression Network (PRNet) in this paper, as shown in Fig. 2. Specifically, we formulate lane detection into three subtasks, *i.e.*, polynomial regression, initialization classification, and height regression, and complete them by three branches with sharing the input features. Here polynomial regression is the major task that is used to estimate the polynomial coefficients of lane segments. Initialization classification is used to detect the initial point of each lane for retrieving the coefficients of the first segment from polynomial map. Height regression is to estimate the height of the ending point for each lane, which determines the ending point together with the estimated polynomial curve. Once the results of three branches are obtained, we can directly construct the curve representation of lanes, where each lane consists of k polynomials.

More specifically, a backbone network is employed to extract the shared features of three branches of PRNet. Here the down-sampled features with global receptive field at the input images are used, *i.e.*, the decoder in the segmentation-based methods is eliminated, since the information of encoded features is enough

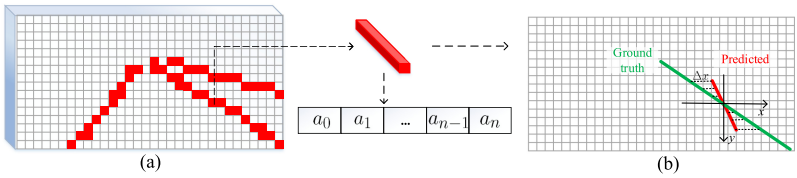

Fig. 3. Illustration of Polynomial Regression. (a) Polynomial map, where the red points denote the used points during training, namely, *polynomial points*. (b) One polynomial piece, where the red line denotes the predicted one from a polynomial point, and the green one corresponds to the ground truth. The differences of sampled points on two lines are used to calculate the loss. Best viewed in color. (Color figure online)

for PRNet. Such a design makes PRNet very efficient. In our implementation, 8× down-sampling is particularly adopted that can achieve a good trade-off between efficiency and effectiveness. Note that we design the output maps of three branches to have the same spatial size, in which the points of three maps at the same position together represent the polynomial curve of a lane segment. In the following, we elaborate on the important components of PRNet.

3.1 Polynomial Regression

The polynomial regression branch is used to predict the polynomial coefficients of all lane segments in an image. For such a task, we design the output to be a $(n+1)$-channel map with the same size as the input features, which is called *polynomial map*. One point in the polynomial map denotes a n-order polynomial. In our implementation, only a part of points are chosen to represent the lane segments, which are called *polynomial points* in this paper. Particularly, the points lying on the lanes are used to calculate the loss during training, *e.g.*, the red points in Fig. 2. Each polynomial point is to perform regression of the closest lane segment. More specifically, we segment traffic lanes in images along the vertical orientation, *i.e.*, the height H is used to denote the length of polynomial pieces. In our implementation, this branch only contains one convolutional layer and thus is highly efficient.

Formally, let $[a_0, a_1, \cdots, a_n]$ and $[\bar{a}_0, \bar{a}_1, \cdots, \bar{a}_n]$ denote the predicted polynomial coefficients and corresponding ground truth for one lane segment. To supervise training of the network, we propose to transform each polynomial segment into some sampling points in images. Particularly, we first sample m points uniformly along the vertical orientation for each lane segment, and then compute the corresponding horizontal coordinates by applying them to the involved polynomial. As a result, we can get $\{(x_p^1, y_p^1), (x_p^2, y_p^2), \cdots, (x_p^m, y_p^m)\}$ and $\{(x_{gt}^1, y_{gt}^1), (x_{gt}^2, y_{gt}^2), \cdots, (x_{gt}^m, y_{gt}^m)\}$ corresponding to the predicted polynomial piece and ground truth. Obviously, $y_p^i = y_{gt}^i$. In this work, the polynomials are enforced to fit the ground truth that are inherently continuous and we use the differences of sampled points on the x-coordinate to define the loss, *i.e.*,

$$L_{poly}(x_p, x_{gt}) = \frac{1}{m}\sum_{i=1}^{m} smooth_{L_1}(x_p^i - x_{gt}^i), \tag{2}$$

where

$$smooth_{L_1}(x) = \begin{cases} \dfrac{0.5(x)^2}{\beta} & if\ |x| < \beta \\ |x| - 0.5\beta & otherwise \end{cases} \tag{3}$$

It can be seen that when $|x_p^i - x_{gt}^i| < \beta$, the predicted point is considered near the traffic lane and the L_2 loss is adopted. The computation of polynomial regression loss is illustrated in Fig. 3.

3.2 Initialization Classification

The initialization classification branch is used to detect the initial retrieval points of all lanes in an image. Through this subtask, we can identify arbitrary number of lanes in principle as each point represents one traffic lane. Here we particularly define the initial retrieval point of a lane by its closest point to the bottom boundary of image. Considering the perspective projection of car cameras, such points are usually far apart from each other, which makes accurate detection easier than the dense points. Note that the initial retrieval points are mainly used to retrieve the polynomial coefficients from the polynomial map rather than to determine their starting points in image. Here the standard cross entropy loss is adopted, and a probability map with the same size as input features would be produced, which is called *initialization map*. Similar to polynomial regression, this branch only contains one convolutional layer. During inference, we get the initial retrieval points by scanning the initialization map. The points whose probability is local maximum and greater than the threshold are considered as the initial retrieval points. Here no post-processing technique are applied in our implementation.

3.3 Height Regression

An intuitive approach to get the ending point of each lane is to directly detect them, as in initialization classification. However, the ending points of traffic lanes in an image are often close to each other due to perspective projection. Consequently, it is difficult to accurately localize them and match them with traffic lanes. Instead, we propose to estimate the height of ending point for each traffic lane, as in [24], which together with the estimated polynomial curve can exactly produce the ending point.

Similar to the polynomial regression branch, this branch regresses the heights of ending points of all traffic lanes, and produces an one-channel *height map* with the same size as the input features. One point in the height map gives the estimated height of ending point of the traffic lane it belongs to. Specifically, only the points lying on traffic lanes are used in training the network, *e.g.*, the yellow points in Fig. 2. Here the smooth $L1$ loss [19] in Eq. (3) is adopted. Similarly, the branch only contains one convolutional layer.

Algorithm 1: Lane Construction

Input: L_1: Initial retrieval point whose height is h_r; P_{map}: Polynomial map; H_{map}: Height map; H: Height of lane segments.

Output: Polynomial coefficients and height of the lane.

Retrieve the polynomial coefficients of first polynomial piece $A_1 = [a_0^1, a_1^1, \cdots, a_n^1]$ from P_{map} at L_1;

Retrieve the initial ending height h_{end}^1 from the height map H_{map} at L_1 ;

$i = 1$; h_e=h_{end}^1;

while $h_e < h_r - i * H$ **do**

    i=i+1;

    Get L_i by applying the y-coordinate $(h_r - (i-1)H)$ to A_{i-1};

    Retrieve the polynomial coefficients of i^{th} polynomial piece $A_i = [a_0^i, a_1^i, \cdots, a_n^i]$ from P_{map} at L_i;

    Retrieve the i^{th} height h_{end}^i from H_{map} at L_i;

    Update h_e by voting over $\{h_{end}^1, \cdots, h_{end}^i\}$;

Return Polynomial coefficients $\{A_i\}_{i=1}^k$ and the height h_e.

3.4 Lane Construction

The three branches of PRNet produce the polynomial coefficients, initial retrieval points, and heights of ending points. Here we explain how to construct each traffic lane in an image using the produced information. Algorithm 1 gives the procedure to construct one of traffic lanes, and Fig. 4 illustrates it. Note that the maps produced by three branches have the same size, implying that they can naturally match with each other.

Specifically, we first get all initial retrieval points by scanning the initialization map, each of which represents one traffic lane. Then we construct the traffic lanes one-by-one by connecting multiple lane pieces belonging to the same lane and at the same time calculating the height of the ending point. For a single traffic lane, the initial retrieval point is used to retrieve the polynomial coefficients of first polynomial piece and initial height of ending point, and additionally its height is considered as the height of the starting point. For next lane segment, we first use the vertical interval H to get the y-coordinate of retrieval point and then get the x-coordinate by applying it to the current polynomial piece. That is, the ending point of current polynomial piece is regarded as the retrieval point of next polynomial piece. For each iteration, we would update the estimated height of ending point. Here a voting strategy is particularly adopted over the currently obtained height values, *i.e.*, the most often value is selected as the estimated height. Note that the height values are discretized with an interval of ten pixels in our implementation. In our experimental evaluation, the lanes are represented by the sampled points from polynomials which inherently form the continuous lane curves.

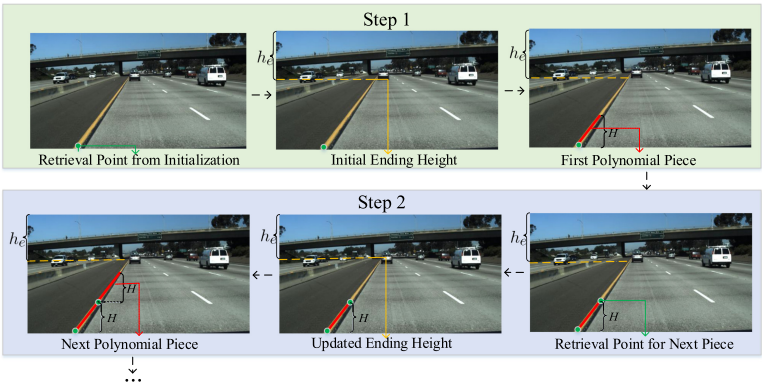

Fig. 4. Illustration of lane construction. We first get the initial retrieval point by scanning initialization map, and use it to retrieve the initial height from height map and polynomial coefficients of first polynomial piece from polynomial map. Then we can get the ending point of current polynomial piece, which is used as the retrieval point of next polynomial piece. The procedure is repeated until the ending point of the traffic lane is reached. Here the polynomial pieces are connected to form a traffic lane and the height is updated iteratively. Best viewed in color. (Color figure online)

4 Experiment

In this section, we experimentally evaluate our proposed PRNet on two popular benchmark datasets: TuSimple [23] and CULane [17]. The representative lane detection methods are used for comparison, including Line-CNN [13], LaneNet [16], EL-GAN [6], SCNN [17], FastDraw [18], 3D-LaneNet [5], SAD [10], and LeastSquares [24]. For each dataset, the reported results of methods in the original literatures are adopted for performance comparison, and one method would not be involved if it does not offer the corresponding results.

4.1 Experimental Setup

To show the generalization of our PRNet, we choose the BiSeNet [25] with ResNet18 [8] and ERFNet [20] as the backbone. Both of them are efficient and their features have a global receptive field at the input image. Specifically, we replace the FFM module of BiSeNet and the decoder module of ERFNet with one convolutional layer followed by a SCNN_D block [17], which can effectively extract discriminative features for lane detection. All the networks are implemented in PyTorch, and we run experiments on NVIDIA GTX1080Ti GPUs. The model pretrained on the ImageNet is used for initialization. For PRNet, we train the three branches jointly and the loss weights of the three branches are set to 1, 1 and 0.1 respectively. Adam optimizer is adopted for optimization. The learning rate is set to 0.0001. The hyper-parameters m, β mentioned in Sect. 3.1 are set to 20 and 0.005, which are determined empirically by cross validation.

Table 1. Performance comparison of different lane detection methods on TuSimple (test set).

Method	Backbone	Extra data	Accuracy	FP	FN
Line-CNN [13]	ResNet50	-	96.87%	0.0442	0.0197
LaneNet [16]	ENet	No	96.38%	0.0780	0.0244
EL-GAN [6]	ENet	No	96.39%	0.0412	0.0336
SCNN [17]	VGG16	Yes	96.53%	0.0617	0.0180
FastDraw [18]	ResNet50	No	95.2%	0.076	0.045
SAD [10]	ENet	No	96.64%	0.0602	0.0205
3D-LaneNet [5]	VGG16	No	95.20%	-	-
LeastSquares [24]	ERFNet	No	95.80%	-	-
PRNet	BiSeNet	No	**97.18%**	**0.0397**	**0.0172**
PRNet	ERFNet	No	97.00%	0.0491	0.0209

Throughout the experiments, images in TuSimple and CULane datasets are first resized to 256×512 and 256×768 respectively. Three types of data augmentation strategies are adopted, including randomly flipped, randomly rotated, and randomly varying brightness.

4.2 Results on TuSimple

Dataset. TuSimple [23] is a popular dataset for lane detection in recent years. It includes $3,268$ images for training, 358 images for validation, and $2,782$ images for test. The sizes of these images are all 720×1280. The annotations of traffic lanes are given in the form of polylines of lane markings, which have a fixed height-interval of 10 pixels. For each image, only the current (ego) lanes and left/right lanes are annotated in both the training and test set. When a lane is crossed, a 5^{th} lane would be added to avoid confusion, which means that each image contains at most 5 lanes.

Evaluation Metrics. We follow the official evaluation metrics ($Acc/FP/FN$). The accuracy is defined as $Acc = \frac{C_{pred}}{T_{gt}}$, where C_{pred} is the number of lane points correctly predicted by the network and T_{gt} is the total number of lane points in ground truth. FP and FN are defined as $FP = \frac{F_{pred}}{N_{pred}}$ and $FN = \frac{M_{pred}}{N_{gt}}$, where F_{pred} is the number of wrongly predicted lanes, N_{pred} is the total number of predicted lanes, and M_{pred} is the number of missed lanes, and N_{gt} is the number of all groundtruth lanes.

Performance Comparison. Table 1 reports the performance comparison of our PRNet against the previous representative methods, where the test set of TuSimple is adopted for evaluation. It can be seen that our method outperforms the previous state-of-the-art methods on all three metrics, which implies that PRNet can detect the lanes more accurately with less wrong prediction and lane missing. Note that no extra data are used for training our PRNet.

Table 2. Performance (F_1-measure) of different lane detection methods on CULane (test set). Here $*$ denotes that the backbone is BiSeNet with ResNet18 and † denotes that the backbone is ERFNet. For crossroad, only FP is reported for fair comparison. The second column denotes the proportion of each scenario in the test set.

Category	Proportion	SCNN [17]	FastDraw [18]	SAD [10]	**PRNet***	**PRNet†**
Normal	27.7%	90.6	85.9	90.7	90.8	**92.0**
Crowded	23.4%	69.7	63.6	70.0	72.3	**74.7**
Night	20.3%	66.1	57.8	66.3	69.2	**70.5**
No line	11.7%	43.4	40.6	43.5	47.6	**51.7**
Shadow	2.7%	66.9	59.9	67.0	70.6	**76.0**
Arrow	2.6%	84.1	79.4	84.4	85.2	**87.8**
Dazzle light	1.4%	58.5	57.0	59.9	64.2	**68.4**
Curve	1.2%	65.7	65.2	65.7	67.2	**70.0**
Crossroad	9.0%	1990	7013	2052	**1113**	2114
Total	–	71.6	–	71.8	74.8	**76.4**

4.3 Results on CULane

Dataset. CULane [17] is a large lane detection dataset which contains about $130k$ images. The dataset is divided into the training set with $88,880$ images, validation set with $9,675$ images, and test set with $34,680$ images. The images are collected at the urban, rural, and highways in Beijing. All images in CULane datest have the same resolution of 590×1640. For each image, only at most 4 lanes are annotated: the current (ego) lanes and left/right lanes. The format of annotations are same with the TuSimple dataset. In general, the CULane dataset is considered more challenging than the TuSimple dataset.

Evaluation Metrics. Following SCNN [17], we extend the predicted lanes to a width of 30 pixels and then calculate the intersection-over-union (IoU) between the ground truth and prediction. True positives (TP) are the number of predicted lanes whose IoUs are greater than a certain threshold, and false positives (FP) are opposite. Here we choose 0.5 as the preset threshold by following [17]. False negatives (FN) are the number of missed lanes. Then we adopt the F_1-measure to evaluate the methods, which is defined as $F_1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$, where $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$.

Performance Comparison. As the CULane dataset is more challenging than the TuSimple dataset, the results on CULane can better demonstrate the capacity of different methods. Table 2 gives the detection performance of different methods, and we have the following observations. First, our method can always get better results than the previous state-of-the-art methods for each category. Second, the performance improvement of our method is more significant for complex scenarios, *e.g.*, *crowded*, *dazzle light*, and *no line*, which well demonstrates the robustness of PRNet to the traffic conditions. We also provide the visualization results of some examples in Fig. 5, which intuitively show the performance of different lane detection methods.

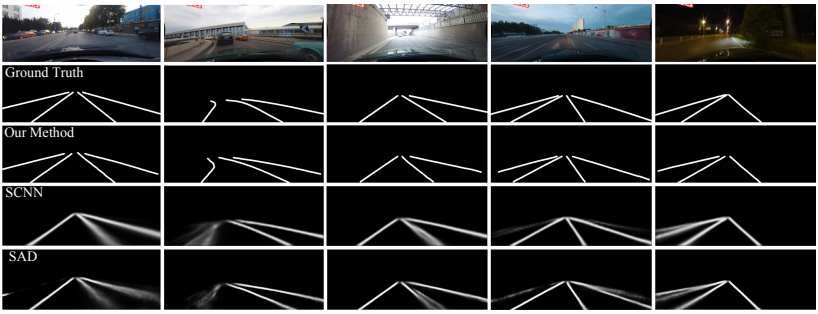

Fig. 5. Visualization of different methods. The segmentation based methods often fail to predict some lanes when the scenarios are complex. Our PRNet could handle the complex scenarios well.

Table 3. Performance comparison between segmentation and regression. Here $*$ denotes that the backbone is BiSeNet with ResNet18 and † denotes that the backbone is ERFNet. Here the accuracy and F_1-measure are reported for TuSimple and CULane respectively, and the test set is used.

Method	TuSimple	CULane
Segmentation*	96.11%	70.7
Regression (ours)*	**97.18%**	**74.8**
Segmentation†	95.92%	73.6
Regression (ours)†	**97.00%**	**76.4**

4.4 Ablation Study

Regression *vs* Segmentation. The key idea of our PRNet is to use polynomial regression to complete lane detection rather than semantic segmentation in previous works. Here we particularly explore the advantages of regression by fairly comparing them with the same backbone and settings. Specifically, we construct a semantic segmentation head (producing lane markings) and a lane classification head (judging existence of lane markings) by following SCNN [17], and then append them to the backbone of PRNet, like the polynomial regression head. Moreover, the segmentation results will be fitted as splines for evaluation. We conduct the experiments on both TuSimple and CULane datasets, and Table 3 provides the results. Evidently, the experimental results well demonstrate the superiority of our proposed regression to semantic segmentation, especially for more challenging CULane.

Polynomial Order and Piece Height of Polynomials. In PRNet, the polynomial order n and piece height of polynomials H are two main hyper-parameters, which represent the ability and complexity to describe lane curves. In principle, a smaller piece height requires a lower polynomial order since shorter lane segments are easier to be fitted by curves. Here we study the effects of dif-

Table 4. Detection performance of different polynomial orders and piece heights. Here the F_1-measure on CULane (validation set) is particularly reported.

Order (n) /Height (H)	8	16	32	64
1	77.31	76.85	76.82	76.66
2	77.48	**77.72**	77.18	76.80
3	77.66	77.70	77.59	77.23

Table 5. Run-time performance of different methods. Here * denotes that we run the model provided by authors on the used platform, and otherwise the result reported in the original paper is directly adopted.

Method	Platform	Input size	FPS
Line-CNN (Res50) [13]	Titan X	288×512	30
SCNN (VGG16)* [17]	GTX1080Ti	208×976	20
LaneNet (ENet) [16]	GTX1080Ti	256×512	52
FastDraw (Res50) [18]	GTX1080	128×256	90
SAD (ENet)* [10]	GTX1080Ti	208×976	79
LeastSquares (ERFNet)* [24]	GTX1080Ti	256×512	50
Ours (BiSeNet)*	GTX1080Ti	256×512	**110**
Ours (ERFNet)*	GTX1080Ti	256×512	81

Table 6. The statistics of failure cases on two datasets. Here the four failure categories are adopted.

Category	IRPM	IRPW	PRI	HRI
TuSimple	134	502	9	26
CULane	28457	16104	669	334

ferent combinations of polynomial order and piece height. Particularly, CULane is adopted due to its challenging and the BiSeNet with ResNet18 is chosen as the backbone of PRNet. Table 4 shows the results, and we have the following observations. First, for a large piece height (*e.g.*, 64), a higher polynomial order is better since more powerful ability of fitting is required. Second, a low order is enough to achieve good detection performance for some reasonable piece height (*e.g.*, 16). Considering the complexity, we finally set the polynomial order $n = 2$ and piece height $H = 16$ throughout the experiments.

Run-Time Performance. Here we evaluate the run-time performance of different lane detection methods. Particularly, one GTX1080Ti GPU is used for fair comparison. Table 5 gives the run-time performance. It can be seen that our proposed PRNet achieves a speed of 110 FPS for the backbone of BiSeNet with ResNet18, which is very competitive to other state-of-the-art methods.

Fig. 6. Visualization of Failure cases. Top Row: images with ground truth. Bottom Row: the results produced by our PRNet. Here the four categories of failures are shown from left to right, including initial retrieval point missing (IRPM), wrong prediction of initial retrieval points (IRPW), inaccuracy of polynomial regression (PRI), and inaccuracy of height regression (HRI).

Failure Cases Analysis. Here we analyse the failure cases of our PRNet on both TuSimple and CULane datasets. We classify failure cases into four categories: initial retrieval point missing (IRPM), wrong prediction of initial retrieval points (IRPW), inaccuracy of polynomial regression (PRI), and inaccuracy of height regression (HRI). Table 6 shows the statistics over the four categories of failure cases, and Fig. 6 gives the visualization of typical failure cases. From the results, it can be seen that most of failures are about initial retrieval points, including wrong prediction and detection missing. Furthermore, we visualize many failure cases to deeply analyse the failure causes. We find that detection missing is mainly due to irregular scenes, *e.g.*, dark light, crowded vehicles, and no lane markings, and wrong prediction is mainly due to deceptive or confused scenes, *e.g.*, lane-like lines and unlabeled lanes (both datasets limit the number of lanes to annotate according to their protocols). For the complex scenes, however, our PRNet performs much better than other methods, as shown in Table 2. To further address these issues, we plan to introduce the structure information in the future works, *e.g.*, embedding the layout of lanes into network.

5 Conclusion

In this paper, we propose to use the in-network polynomial curves to represent the traffic lanes in images, and then propose a novel polynomial regression network (PRNet) for variable-number lane detection. Specifically, PRNet consists of three cooperative branches: polynomial regression, initialization classification, and height regression. The experimental results on two benchmark datasets show our proposed method significantly outperforms the previous state-of-the-art methods, and achieves competitive run-time performance. In particular, our PRNet presents better robustness to complex traffic conditions than other methods.

Acknowledgment. This work is supported by the National Natural Science Foundation of China under Grant 61673362 and 61836008, Youth Innovation Promotion Association CAS (2017496), and the Fundamental Research Funds for the Central Universities.

References

1. Aly, M.: Real time detection of lane markers in urban streets. In: IV (2008)
2. Borkar, A., Hayes, M., Smith, M.T.: Polar randomized hough transform for lane detection using loose constraints of parallel lines. In: ICASSP (2011)
3. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD (1996)
4. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24**, 381–395 (1981)
5. Garnett, N., Cohen, R., Pe'er, T., Lahav, R., Levi, D.: 3D-lanenet: end-to-end 3D multiple lane detection. In: CVPR (2019)
6. Ghafoorian, M., Nugteren, C., Baka, N., Booij, O., Hofmann, M.: EL-GAN: embedding loss driven generative adversarial networks for lane detection. In: Leal-Taixé, L., Roth, S. (eds.) ECCV 2018. LNCS, vol. 11129, pp. 256–272. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11009-3_15
7. Gonzalez, J.P., Ozguner, U.: Lane detection using histogram-based segmentation and decision trees. In: IEEE Intelligent Transportation Systems (2000)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
9. He, Y., Wang, H., Zhang, B.: Color-based road detection in urban traffic scenes. In: IEEE Transactions on Intelligent Transportation Systems, pp. 309–318 (2004)
10. Hou, Y., Ma, Z., Liu, C., Loy, C.C.: Learning lightweight lane detection CNNs by self attention distillation. In: ICCV (2019)
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
12. Lee, S., et al.: Vpgnet: Vanishing point guided network for lane and road marking detection and recognition. In: ICCV (2017)
13. Li, X., Li, J., Hu, X., Yang, J.: Line-CNN: end-to-end traffic line detection with line proposal unit. IEEE Trans. Intell. Transp. Syst. **21**, 248–258 (2019)
14. McCall, J.C., Trivedi, M.M.: An integrated, robust approach to lane marking detection and lane tracking. In: IV (2004)
15. McCall, J.C., Trivedi, M.M.: Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation (2006)
16. Neven, D., De Brabandere, B., Georgoulis, S., Proesmans, M., Van Gool, L.: Towards end-to-end lane detection: an instance segmentation approach. In: IV (2018)
17. Pan, X., Shi, J., Luo, P., Wang, X., Tang, X.: Spatial as deep: Spatial cnn for traffic scene understanding. In: AAAI (2018)
18. Philion, J.: FastDraw: addressing the long tail of lane detection by adapting a sequential prediction network. In: CVPR (2019)
19. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems (2015)

20. Romera, E., Alvarez, J.M., Bergasa, L.M., Arroyo, R.: ERFNet: efficient residual factorized convNet for real-time semantic segmentation. IEEE Trans. Intell. Transp. Syst. **19**, 263–272 (2017)
21. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
22. Szegedy, C., et al.: Going deeper with convolutions. In: CVPR (2015)
23. TuSimple: http://benchmark.tusimple.ai/#/t/1. Accessed 08 Sept 2018
24. Van Gansbeke, W., De Brabandere, B., Neven, D., Proesmans, M., Van Gool, L.: End-to-end lane detection through differentiable least-squares fitting. In: ICCV Workshop (2019)
25. Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., Sang, N.: BiSeNet: bilateral segmentation network for real-time semantic segmentation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11217, pp. 334–349. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01261-8_20
26. Zhou, S., Jiang, Y., Xi, J., Gong, J., Xiong, G., Chen, H.: A novel lane detection based on geometrical model and Gabor filter. In: IV (2010)