

# Focus on Local: Detecting Lane Marker from Bottom Up via Key Point

Zhan Qu\* Huan Jin Yang Zhou Zhen Yang Wei Zhang

Noah's Ark Lab, Huawei Technologies

{quzhan, jinhuan3, zhouyang116, yang.zhen, wz.zhang}@huawei.com

## Abstract

*Mainstream lane marker detection methods are implemented by predicting the overall structure and deriving parametric curves through post-processing. Complex lane line shapes require high-dimensional output of CNNs to model global structures, which further increases the demand for model capacity and training data. In contrast, the locality of a lane marker has finite geometric variations and spatial coverage. We propose a novel lane marker detection solution, FOLOLane, that focuses on modeling local patterns and achieving prediction of global structures in a bottom-up manner. Specifically, the CNN models low-complexity local patterns with two separate heads, the first one predicts the existence of key points, and the second refines the location of key points in the local range and correlates key points of the same lane line. The locality of the task is consistent with the limited FOV of the feature in CNN, which in turn leads to more stable training and better generalization. In addition, an efficiency-oriented decoding algorithm was proposed as well as a greedy one, which achieves 36% runtime gains at the cost of negligible performance degradation. Both of the two decoders integrated local information into the global geometry of lane markers. In the absence of a complex network architecture design, the proposed method greatly outperforms all existing methods on public datasets while achieving the best state-of-the-art results and real-time processing simultaneously.*

## 1. Introduction

In autonomous driving system (**ADS**), lane detection plays an important role. On the one hand, the location of host and other traffic participants in the lane forms the basis of autonomous driving decisions. On the other hand, the geometry of a lane marker can be viewed as an important *landmark* of the environment and aligned with a high-resolution or vector map for high-precision positioning. At the same time, lane detection has been widely used in Ad-

vanced Driver Assistance Systems (**ADAS**) and is the basis for some common features such as Lane Keep Assist (**LKA**) and Adaptive Cruise Control (**ACC**).

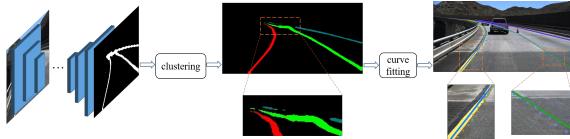
Recent advances in lane detection can be attributed to the development of convolutional neural networks (**CNN**). Most existing methods adopt well-studied frameworks such as semantic segmentation and object detection to parse lane markers and transform the network output into parametric curves through post-processing. However, the mostly used frameworks can not be seamlessly generalized to curved-shaped lane lines because lane detection task requires precise representation of local positions and global shapes simultaneously, showing their own limitations.

The semantic segmentation-based approach predicts binary masks of lane marker regions, inserts clustering models into training and inference, groups masked pixels into individual instances, and finally uses curve fitting to parametric results. However, the clustering procedure complicates the training and inference pipeline. In addition, pixel-level inputs to curve fitting are often redundant and noisy, all of which bring negative impact to the accuracy of the final results. Fig.1(a) shows several cases where the prediction errors may increase. Object detection approaches are originally designed for compact target and produce bounding box as output, which is insensitive to pixel-level error when faced with large-scale object. As for lane markers, they typically span half or more of the image, and pixel-level localization errors significantly impair detection performance, which can be attributed to the limited field of view (**FOV**) of features learned through CNN being insufficient to model content that is too far apart. Fig.1(b) illustrates the effect of FOV in complex scenario. Moreover, most of these solutions model global geometry directly, and the network must produce high-dimensional outputs to describe the curves. Theoretically, however, uncompact outputs increase the demand for data and model capacity, ultimately masking the generalization ability of the resulting model.

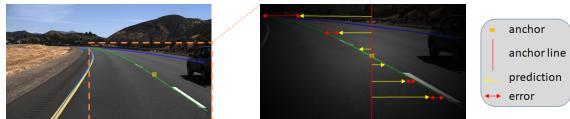
Although the global structure of lane markers has some complexity, we note that local lane markers are extremely simple and that global lane markers can be approximated by a combination of local line segments. Moreover, spa-

---

\*Z. Qu is the corresponding author.



(a) Segmentation based method and intermediate results.



(b) Object detection based method.

Figure 1. Pipeline of existing methods and illustration of common error prediction. (a). the enlarged window in the middle of the pipeline shows the incorrect clustering of the segmentation mask, and the other two orange windows in the curve fitting output show the position deviation of the prediction curve due to redundant and noisy pixels, note the solid line is ground truth, dotted line is prediction. (b). shows the influence of FOV in detection based methods, the brightness of the right image reflects the practical FOV of anchor.

tial locality is more suitable for modeling with CNN. Following this intuition, a novel lane marker detection method, FOOLane, is proposed that focuses on modeling local geometry and integrating them into the global results in a bottom-up manner. Specifically, the geometry of the lane marker is predicted by estimating adjacent *keypoints* on the it. In the bottom stage, a fully convolutional network is used to capture keypoints in the local scope through two separate heads. The first one gives the probability that keypoints appear in pixel space, and the second one gives the offset between keypoints and the most spatially correlated local lane marker, which is used to refine the positions of keypoints generated by the first head and construct associations between keypoints on the same lane markers. Based on the local information, two decoding algorithms with different preferences are proposed to predict global geometry of lane markers. The bottom-up pipeline of the proposed method is shown in Fig.2.

Compared with existing works [2, 19, 16, 10, 12], the proposed approach concentrates the capabilities of CNN on a local scale, which is suitable for CNN's limited FOV, and significantly reduces the complexity of the task and the dimension of the output. As a result, the compact output leads to stable and efficient training without additional effort in network architecture design and data collection. Considering the continuity of lane markers, the proposed decoder is able to associate keypoints of the same instance and optimize the geometry of network predictions without affecting performance and efficiency. Furthermore, during network training and instance decoding, we model and predict keypoints using features with the highest spatial correlation guided by coarse-to-fine strategies. The proposed bottom-

up solution achieves the best state-of-the-art level, ***Acc: 96.92%*** on TuSimple and ***F1 score: 78.8%*** on CULane, and excellent generalization in the two public datasets. Together with the compatibility with network architectures, our approach shows a promising application future.

We emphasize that our method is the first to formulate lane detection into multi-key-points estimation and association problem, which is inspired by the bottom-up human pose estimation framework [15, 1, 3]. The proposed local scope based method avoids the inaccurate prediction where far from the anchor, which occurs in detection-based methods. And the sparsity of key points prevents the noisy and redundant output occurred in segmentation-based methods, which decrease the precision and increase the delay of curve fitting. With extensive experiments, our solution proves the potential of applying pose estimation approaches on lane detection, which opens up a new direction to solve this important application problem. Our solution does not depend on *CNN* architecture, is readily compatible to newly developed architecture and shows scalable potential on accuracy and efficiency.

Our contributions can be summarized as follows:

- Lane detection is firstly decompose into subtasks of modelling local geometry, which is achieved by estimating keypoints on local curve. Simplified targets and focus on spatially limited scope helps the network to provide precise estimation of local curve.
- Two decoding algorithms with different preferences are designed to integrate local information into global prediction, which enable the system to achieve high accuracy in ultra real time.
- Experimental results showed that our approach outperforms all existing methods by a substantial margin. Besides, our model shows the best generalization ability in comparison, which further proves the potential for productization.

## 2. Related Work

**Lane Marker Detection.** Lane marker detection based on deep learning can be categorized into two groups: detection based and segmentation based. The former one: [2] proposed an anchor-based lane marker detection model for forward-looking cameras. Lane markers were uniformly sampled along the vertical axis in the image, and dense regression was performed by predicting the offset between each sample point and an anchor line, then Non-Maximum Suppression(*NMS*) was applied to suppress the overlapping detection and select the best lane marker with the highest score. [19] proposed the use of neural architecture search(*NAS*) to find a better backbone and a point blending based post processing to further improve the performance

of lane marker detection task. [6] proposed to train a CNN to predict the existence, position and feature embedding of lane markers in an image. A lane marker instance was clustered based on the trained feature embedding. [16] formulated lane marker detection as a pixel-wise classification problem for each row of an image. A specific feature map was predicted to indicate the position of a lane marker on each row.

Segmentation based: [8] proposed a multitask framework, which predicted pixel-wise multi-label and clustered the pixels belonging to same lane instance in bird eye view image using DBSCAN. It also added an auxiliary task: vanish point estimation, to increase the stability of lane marker detection. [10] proposed an end-to-end joint semantic segmentation and feature embedding network architecture. Pixels on the same lane marker were assigned an identical instance id. [12] also designed an instance segmentation network for lane marker detection problem. Different from [10], [12] predicted a probability map for each lane marker separately and used cubic splines to fit it. In stead of using pixel-wise classification, [20] introduced a row-wise classification architecture. For each row, it predicted the most possible grid of a lane marker in an image and recovered a lane marker instance through post processing. [9] proposed a CycleGAN based method to enhance lane detection performance in low light conditions. [4] claimed a more accurate method by using EL-GAN for lane marker detection, which used a generator to segment the lane markers and a discriminator to refine the segmentation result. [5] proposed a self-attention distillation method for lane marker segmentation task by forcing shallow layers to learn rich context feature from deep layers.

**Bottom-Up Human Key Point Detection.** [15] proposed a bottom-up method for crowded scenes, which detected keypoints and built a densely connected graph, the weight of each edge represented the correlation of two keypoints. By optimizing the graph, keypoints belonging to one person were clustered. [1] predicted a heat map for each keypoint and part affinity fields (PAFs) which were used to associate body parts with individuals in the image. Similar to [3, 10], [11] introduced feature embedding to facilitate keypoints clustering of one person while predicting the heat map of keypoints. [13] further split the problem into two stages: (1) predicting heat map and short-range offset for keypoints detection, (2) clustering key points using mid-range offset for one person.

We find that lane marker detection can be abstracted as discrete keypoints detection and association problem, which is very similar to bottom-up human key point detection task. [14] proposed a method based on this idea. A network was trained to extract all possible lane marker pixels and output the pixels in the neighboring row, which belongs to the same lane as the current lane marker pixel. As the problems

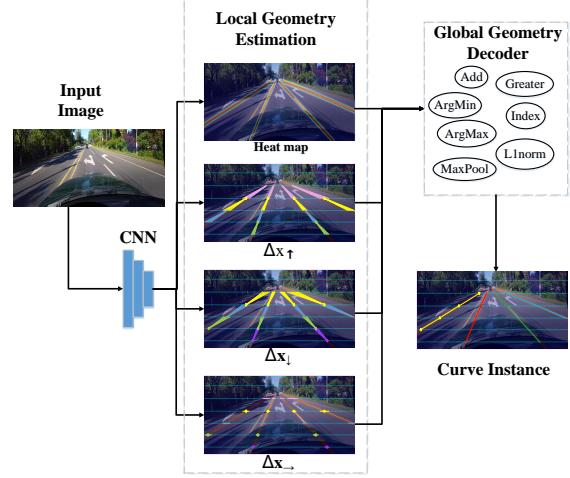


Figure 2. The inference process of FOOLane. The network produces 4 logits expressing the geometry of the local curve. The decoder module constituted with low-level operators integrates the local information into curve instances.

discussed above, the inherent segmentation-based method inhibited the precise representation of a lane marker. In addition, the pixel-wise joint distribution prediction was redundant.

### 3. Methodology

As shown in Fig.2, we proposed a bottom-up lane detection method by estimating the existence and the offsets of the local lane point through the network, followed by a novel global geometry decoder to generate the final curve instances.

#### 3.1. Network for local geometry

In the proposed approach, each predicted marker curve is represented as an ordered keypoints set, where the key points are of fixed/predefined vertical interval  $\Delta y$  across neighboring rows. First of all, the task of curve prediction is decomposed into local subtasks via a fully convolutional network with two heads. The heatmap outputted by the first head expresses the possibility that keypoint appears, which resolves the existence of local curves. The second head predicts offsets to key points of the most closed local curve, which describes the precise geometry of the local curve.

**Key point estimation.** Motivated by a curve constituted of points, we adopt a keypoint-estimation-based framework. The network firstly outputs a heatmap with the same resolution as input, which models the probability that pixel is a keypoint of the curve. In the training phase, the points set as annotation of the  $j - th$  curve are interpolated to be continuous in pixel space as  $l_j$ . Each pixel of the curve  $l_j$  is considered as a key point and yields ground-truth value

for neighbors via unnormalized Gaussian kernel. The standard deviation  $\sigma_h$  depends on the scale of input, and if the ground-truth value of some pixel is assigned by multiple keypoints, the maximum will be kept.

To deal with the class imbalance problem coming with the sparsity of key points, we employ penalty-reduced focal loss for this head as in [7, 22], where only pixels with ground truth equal to 1 are considered positive and all others are negative. The penalty from negative pixels arises with the distance to positive, which helps to reduce the influence of ambiguity. We denote the output of  $i$ -th pixel at heatmap as  $s_i$  and the ground-truth value assigned by Gaussian Kernel as  $g_i$ . Define penalty coefficients  $\hat{g}_i$  and  $\hat{s}_i$  as:

$$\hat{g}_i = \begin{cases} 0 & \text{if } g_i = 1 \\ g_i & \text{otherwise} \end{cases}, \quad \hat{s}_i = \begin{cases} s_i & \text{if } g_i = 1 \\ 1 - s_i & \text{otherwise} \end{cases}, \quad (1)$$

and the loss function for heatmap head is constructed as:

$$Loss_h = -\frac{1}{N} \sum_i^N (1 - \hat{g}_i)^\beta (1 - \hat{s}_i)^\gamma \log(\hat{s}_i), \quad (2)$$

where  $\beta$  and  $\gamma$  are tunable hyperparameters, controlling the penalty reduction for ambiguous and simple samples respectively.  $N$  is the number of key points in the current image.

Compared with segmentation-based methods, the loss function Eq.2 guides the network to learn positive and negative samples of keypoint with reduced supervision from the total pixels, prompting pixels best suited for expressing geometry to the response. An example of the heatmap can be found in Fig.2 as the first output of the network, the center of lane marker responses highest, and the neighborhood became colder gradually, which helps prevent the noise and redundancy from propagating to subsequent procedures as well.

**Local geometry construction.** For precise geometry, the second head of the network regresses a vector  $[\Delta x_{\uparrow}(p), \Delta x_{\rightarrow}(p), \Delta x_{\downarrow}(p)]^T$ , describing the local geometry of the closest curve to pixel  $p$ . The elements indicate the horizontal offsets to 3 neighboring key points with fixed vertical interval  $\Delta y$ , which have been colorized for visualization in Fig.2. Given the vector, we can simply recover the local curve related to pixel  $p$ :

$$\hat{l}(p) = \begin{bmatrix} \hat{p}_{\uparrow}(p) \\ \hat{p}_{\rightarrow}(p) \\ \hat{p}_{\downarrow}(p) \end{bmatrix} = p + \begin{bmatrix} \Delta x_{\uparrow}(p) & -\Delta y \\ \Delta x_{\rightarrow}(p) & 0 \\ \Delta x_{\downarrow}(p) & \Delta y \end{bmatrix}, \quad (3)$$

where  $\hat{p}_{\uparrow}(p)$ ,  $\hat{p}_{\rightarrow}(p)$  and  $\hat{p}_{\downarrow}(p)$  denote the actual location with fixed vertical interval  $\Delta y$  to pixel  $p$ , respectively.

In the training phase, all pixels within a fixed distance from key points of the curve  $l$ ,  $N_{\sigma_g(l)}$ , are taken to compute loss for  $\Delta x_{\uparrow}$ ,  $\Delta x_{\downarrow}$ .

$$\begin{aligned} Loss_{\uparrow}(l) &= \frac{1}{|N_{\sigma_g}(l)|} \sum_{p \in N_{\sigma_g}(l)} \|\hat{p}_{\uparrow}(p) - \varphi(l, f_y(p) - \Delta y)\|_1, \\ Loss_{\downarrow}(l) &= \frac{1}{|N_{\sigma_g}(l)|} \sum_{p \in N_{\sigma_g}(l)} \|\hat{p}_{\downarrow}(p) - \varphi(l, f_y(p) + \Delta y)\|_1, \end{aligned} \quad (4)$$

where  $f_y(\cdot)$  denotes the function retrieving vertical coordinate of the pixel,  $\varphi(l, y)$  is function retrieving horizontal coordinate of curve  $l$  on specific row  $y$ .

For  $\Delta x_{\rightarrow}$ , a coarse-to-fine strategy is employed:

$$\begin{aligned} Loss_{\rightarrow}(l) &= \frac{1}{2|N_{\sigma_g}(l)|} \sum_{p \in N_{\sigma_g}(l)} ( \\ &\quad \|\hat{p}_{\rightarrow}((\hat{p}_{\uparrow}(p))) - \varphi(l, f_y(p) - \Delta y)\|_1 + \\ &\quad \|\hat{p}_{\rightarrow}((\hat{p}_{\downarrow}(p))) - \varphi(l, f_y(p) + \Delta y)\|_1 ), \end{aligned} \quad (5)$$

where the training pixels come from the decoded prediction of  $\Delta x_{\uparrow}$  and  $\Delta x_{\downarrow}$  in Eq.4, which is used to compensate for the error in predicting  $\Delta x_{\uparrow}$  and  $\Delta x_{\downarrow}$  and keeps in line with the coarse-to-fine behavior in the decoding stage. L1 loss is employed for all the regression terms.

**Network architecture.** To justify the effectiveness of focusing on local geometry, we adopt light-weight architecture ERFNet [17] and BiSeNet [21], which were originally designed for semantic segmentation on mobile devices. During the feature extraction, the encoder abstracts image into downsampled feature map, then the decoder broadcasts the high-level semantics to the same resolution as input. All 4 logits are yielded by the last block of the decoder for saving memory. Most experiments in this paper are performed basing on ERFNet. Since the method is designed for working in real traffic scenarios, which is required to handle the case of a merged or split marker and any number of instances, there is no extra branch specialized for predefined lane markers as in [12, 5]. The final cost function is formulated as

$$Loss = Loss_h + \lambda(Loss_{\uparrow} + Loss_{\downarrow} + Loss_{\rightarrow}), \quad (6)$$

### 3.2. Decoder for global geometry

In the above section, CNN produces pixel-wise heatmap and offset for keypoints in local scope. These local information are subsequently integrated into prediction of global curve. Specifically, the heatmap is used to determine emergence and termination of curve. The offsets is used to associate keypoints on same curve instance and refine geometry further. To this end, we propose two novel and simple algorithms for decoding the output of CNN under different demand scenarios, which responds to preferences for accuracy and efficiency respectively.

**Greedy decoder** works through iteratively extending the neighbors of keypoint in a greedy search-like manner. For each input image,



Figure 3. Illustration of greedy decoding process. All the keypoints found in process have been shown in color. The colored arrows indicate the refinement of position of keypoints, or the prediction of neighboring points. The invalid points is displayed in gray. Finally, the decoded curve instance is represented as set of keypoints in same color. It's best to zoom in the figure and view it in color.

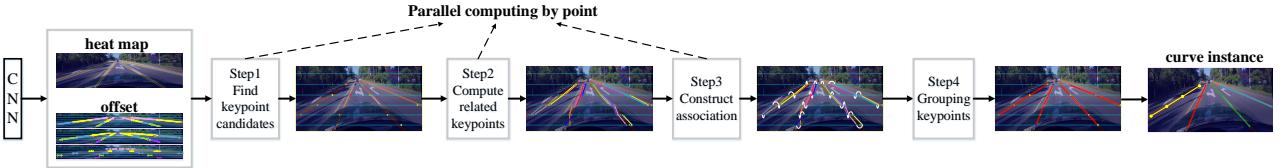


Figure 4. Illustration of efficient decoding process. Different from greedy decoder, which searches keypoints in an iterative manner, the efficient decoder found all the keypoint candidates at the beginning. For these candidates, the position refinement, neighbor prediction and association construction are perform in one step through parallel computing. The white curve indicates association relationship among keypoints.

**Step1** Find the row containing greatest number of local maximum response on heatmap. This row and the points are taken as starting line and **current** keypoints.

**Step2** Refine the position of **current** keypoints. For point  $p$ , refinement can be formulated as  $\hat{p} = p + [\Delta x_{\rightarrow}(p), 0]^T$ .

**Step3** Explore the vertical neighbors of **current** points, the coordinates of which can be computed as  $p_{\uparrow} = \hat{p} + [\Delta x_{\uparrow}(p), -\Delta y]^T$  and  $p_{\downarrow} = \hat{p} + [\Delta x_{\downarrow}(p), \Delta y]^T$ .

**Step4** Examine the heatmap value of  $p_{\uparrow}$  and  $p_{\downarrow}$ . If the value reaches threshold  $\theta_h$ , the corresponding neighboring points is used to update **current** keypoint, and **Step2~4** are repeated. Otherwise the search is terminated, all the points searched from one single point are taken as one global curve.

To sum up, the decoding algorithm gradually extends the global curve by exploring neighbors of keypoint, and refine the geometry of curve in a coarse-to-fine manner. This algorithm can produce precise geometry of curve, but its low efficiency limits the useability in practical application. The process have been shown in color in Fig.4.

**Efficient decoder** is proposed in order to solve the inefficiency problem of greedy decoders, which utilizes the parallelism of computing devices. For each image,

**Step1** Extract rows at equal interval  $\Delta y$  on heatmap. On each row, take the points with local maximum response as **current** keypoints.

**Step2** For each keypoint  $p$ , compute three related points as  $p_{\rightarrow} = p + [\Delta x_{\rightarrow}(p), 0]^T$ ,  $p_{\uparrow} = p + [\Delta x_{\uparrow}(p), -\Delta y]^T$  and  $p_{\downarrow} = p + [\Delta x_{\downarrow}(p), \Delta y]^T$ .

**Step3** Construct association among **current** keypoints located in neighboring rows. For a point  $p$  in  $i$ -th row, two points in  $(i - \Delta y)$ -th row and  $(i + \Delta y)$ -th row will be associated with it, which are closest to the position of  $p_{\uparrow}$  and  $p_{\downarrow}$  respectively.

**Step4** Starting with the row with maximum number of current keypoints. According to the association relationship created in Step3, for each current keypoint, all the keypoints associated with it in above/below rows are iteratively taken out as a single group. Each keypoint group is considered as a global curve, and  $p_{\rightarrow}$  of points are used to refine geometry of curve further.

The efficient decoding algorithm leverages the parallel computing power of device, to create association among keypoints and refine their position, from step1 to step3. Step4 involves only index operations, thus the time overhead is very low. The process have been shown in color in Fig.3.

## 4. Experiments

In this section, firstly we describe the implementation details and evaluation datasets. Followed by the results of comparison with the state-of-the-art, including quantitative and qualitative results. Finally, the discussion of ablation study and generalization are detailed.

Dataset	# Frame	Train	Validation	Test	Resolution	Road type	# Lane
TuSimple	6408	3268	358	2782	1280×720	highway	<=5
CULane	133235	88880	9675	34680	1640×590	urban, rural and highway	<=4

Table 1. Basic information of two lane marker detection datasets.

Category	Proportion	SCNN[12]	ENet-SAD[5]	ERFNet-E2E[20]	SIM-CycleGAN +ERFNet[9]	UFNet[16]	PINet(4H)[6]	<b>FOLOLane (ours)</b>
Normal	27.7%	90.6	90.1	91.0	91.8	90.7	90.3	<b>92.7</b>
Crowded	23.4%	69.7	68.8	73.1	71.8	70.2	72.3	<b>77.8</b>
Night	20.3%	66.1	66.0	67.9	69.4	66.7	67.7	<b>74.5</b>
No line	11.7%	43.4	41.6	46.6	46.1	44.4	49.8	<b>52.1</b>
Shadow	2.7%	66.9	65.9	74.1	76.2	69.3	68.4	<b>79.3</b>
Arrow	2.6%	84.1	84.0	85.8	87.8	85.7	83.7	<b>89.0</b>
Dazzle light	1.4%	58.5	60.2	64.5	66.4	59.5	66.3	<b>75.2</b>
Curve	1.2%	64.4	65.7	<b>71.9</b>	67.1	69.5	65.6	69.4
Crossroad	9.0%	1990	1998	2022	2346	2037	<b>1427</b>	1569
Total	-	71.6	70.8	74.0	73.9	74.4	72.3	<b>78.8</b>

Table 2. Performance of different methods on CULane testing set, with IoU threshold=0.5. For crossroad, only FP are shown.

Method	Accuracy(%)	FP	FN
SCNN[12]	96.53	0.0617	<b>0.0180</b>
LaneNet(+H-Net)[10]	96.40	0.0780	0.0244
EL-GAN[4]	96.39	0.0412	0.0336
PointLaneNet[2]	96.34	0.0467	0.0518
FastDraw[14]	95.2	0.0760	0.0450
ENet-SAD[5]	96.64	0.0602	0.0205
ERFNet-E2E[20]	96.02	0.0321	0.0428
PINet(4H)[6]	96.75	<b>0.0310</b>	0.0250
<b>FOLOLane(ours)</b>	<b>96.92</b>	0.0447	0.0228

Table 3. Performance of different methods on TuSimple testing set.

#### 4.1. Implementation Details

We first resized the width of an image to 976 and kept the aspect ratio on both datasets. The  $\Delta y$  was set as 10 pixels for a trade-off between precision and efficiency. The weight  $\lambda$  for loss function in Eq.[6] was set as 0.02. For optimization, we used Adam optimizer and poly learning rate schedule with an initial learning rate of 0.001. Each mini-batch contained 16 images per GPU and we trained the model using 8 V-100 GPUs for 40 epochs on CULane and 200 epochs on TuSimple, respectively. To reduce overfitting, we used a 0.3 probability of dropout and weight decay with 0.0001. Furthermore, we also applied data augmentation, including random scaling, cropping, horizontal flipping, random rotation, and color jittering, which have been proved to be effective. In the testing phase, we set the threshold of lane existence confidence as 0.5.

As illustrated in Table 1, the basic information of TuSim-

ple and CULane datasets are detailed. And for evaluation criteria, we follow the official metric used in [18] and [12].

#### 4.2. Results

In this section, we show the results on two lane detection datasets. In all experiments, ERFNet[17] is used as our baseline network if not specially mentioned.

**Quantitative results.** To verify the effectiveness of our proposed method, we compared it with state-of-the-art algorithms based on either segmentation or object detection, including SCNN[12], LaneNet(+H-Net)[10], EL-GAN[4], PointLaneNet[2], FastDraw[14], ENet-SAD[5], ERFNet-E2E[20], SIM-CycleGAN+ERFNet[9], UFNet[16] and PINet[6].

As illustrated in Table 2, the proposed method achieves a new SOTA result on the CULane testing set with a 78.8 F1 measure. Compared with the best model as far as we know, PINet(4H), our method outperforms almost all of the scenarios, whose F1 measure improves 4.4%. Because of local occlusions and fogged traffic lines, PINet shows degraded performance in some categories, such as Crowded, Arrow and Curve. Although our method and PINet are both based on key points estimation, in the aforementioned categories, our method outperforms PINet with 5.5%, 5.3%, and 3.8% F1 measure improvements respectively, which indicates our local geometry modeling model and bottom-up pipeline have better lane marker representation capabilities. Besides, an interesting point is that SIM-CycleGAN+ERFNet, which aims at dealing with low light conditions using CycleGAN, is not comparable to our lane marker detection model in the night and dazzle light scenarios, which implies that our approach is of better generalization ability even than GAN

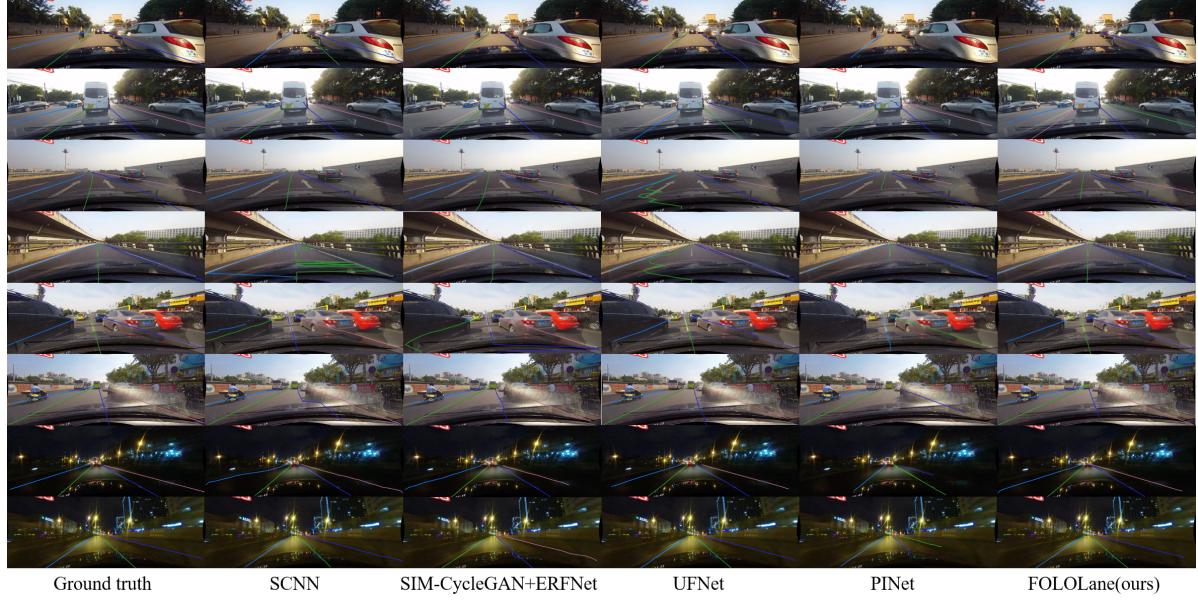


Figure 5. Visualized results of SCNN, SIM-CycleGAN+ERFNet, UFNet, PINet and FOOLane on CULane testing set.

augmented data.

The results of different methods on the TuSimple testing set are shown in Table 3. Due to the limited scale (train/test:3.3k/2.8k) and homogeneous scenario (highway), most methods achieved near-saturated accuracy (more than 96%). Despite this, our method still outperforms the 2nd by 0.17%, close to the difference between 2nd and 4th.

**Qualitative results.** We also show qualitative results of the proposed method and SCNN, SIM-CycleGAN+ERFNet, UFNet, PINet on the CULane testing set. As shown in Fig.5, our method focusing on local geometry and bottom-up strategy helps to distinguish the occlusion of crowded roads and the missing lane marker clues. Through keypoint estimation, the proposed method could yield a smoother and more accurate curve than the others do. Even though in night and dazzling light scenarios, the predicted results are still satisfactory. In conclusion, the proposed method leads to visible improvements in lane marker detection among recently developed segmentation-based and regression-based approaches.

### 4.3. Ablation Study

To investigate the effects of the locally based designs, an ablation study is carried out on the CULane dataset. The experiments are all conducted with the same settings as described in Sec. 4.1 if not specially mentioned.

**Key point estimation.** Different from segmentation-based solutions, our key point estimation method focuses on the center of the lane marker, which achieves an impressive result. Table 4 shows that the proposed method improves

Heatmap	Coarse-to-fine		Decoder		Architecture	F1	Rt.				
	Se.	Ke.	@ test	@ train	Gre.	Eff.	ERF	BiSe			
✓					✓		✓		74.2	-	
	✓				✓		✓		76.6	-	
	✓	✓			✓		✓		77.5	-	
	✓	✓	✓		✓		✓		<b>78.8</b>	25ms	
	✓	✓	✓			✓	✓		78.3	16ms	
	✓	✓	✓			✓		✓	77.5	<b>9ms</b>	

Table 4. Ablation studies on CULane testing set. **Se.:** Semantic segmentation. **Ke.:** Keypoint estimation based heatmap. **@test:** use  $\Delta x_{\rightarrow}$  to refine the geometry of the curve in testing. **@train:** use coarse-to-fine strategy to sample training data for  $\Delta x_{\rightarrow}$  in training. **Gre.:** Greedy decoding. **Eff.:** Efficient decoding. **Rt.:** runtime.

the F1 measure from 74.2 to 76.6, which indicates that the suppression of ambiguous and noisy pixels helps achieve accurate geometry and fewer false positives, improving the performance of a system in turn.

**Coarse-to-fine geometry refinement.** During both network training and instance decoding, we adopt a coarse-to-fine geometry refinement for a more accurate position of key points. In the training phase, the training pixels come from the decoded prediction of  $\Delta x_{\uparrow}$  and  $\Delta x_{\downarrow}$ . In the inference phase, the predicted  $\Delta x_{\rightarrow}$  is employed to refine the position of initial key points and newly explored neighboring key points. The results of different configurations are shown in Table 4. Only using coarse-to-fine in inference improves the F1-measure 0.9%. When coarse-to-fine is extended to training, the performance outperforms that of uni-

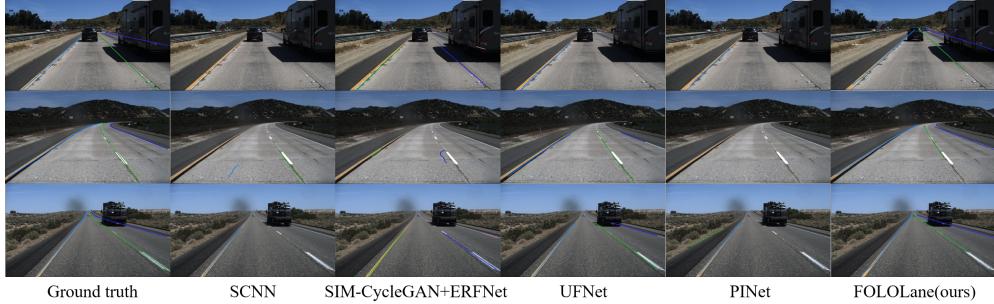


Figure 6. Visualization results of generalizing SCNN, SIM-CycleGAN+ERFNet, UFNet, PINet and FOOLane on TuSimple testing set.

form sampling in  $N_{\sigma_g}(l)$  significantly by 1.3%. The result shows that the direct prediction leads to suboptimal position estimation and our coarse-to-fine strategy could guide the spatially most related representation to capture the geometry of the curve and achieve a more accurate prediction.

**Efficiency-oriented implementation.** As mentioned in Sec. 3.2, efficient decoding is aimed at real-time processing. The main difference from a greedy decoder is that the iteration of decoding neighboring key points is replaced by parallel processing. The parallel decoding significantly improves the efficiency, which achieves 16 ms (64%) runtime gains than greedy decoder at the cost of 0.8% performance degradation. The reason can be attributed to the lack of local optimal estimation in each iteration of greedy decoding.

To maximize efficiency for application, we further replace the basic network from ERFNet to BiSeNet, which is a real-time semantic segmentation network originally designed for mobile devices. Since the output of BiSeNet is 8 times downsampled from the input size, real-time performance is achieved by reaching more than 100 fps and 77.5 F1 measure simultaneously, which is still the best state-of-the-art results excluding the accuracy-oriented version of our approach. On the other side, the experiment also proves the compatibility of the proposed system, which can be readily adapted for more powerful and efficient network architectures up to date.

#### 4.4. Generalization

To further verify the generalization of our proposed method, we employ the checkpoint trained from the CU-Lane training set to inference on the TuSimple testing set. To our knowledge, this is the first attempt to investigate the generalization between these two widely used datasets. Table 5 shows that the proposed method achieves obvious superiority with an accuracy of 84.36%, which surpasses other methods by a significant margin of nearly 20%. The SCNN and PINet(4H) approaches suffer most from the generalization ability, which decreases 90% and 60% respectively. The generalized visualization results on the TuSimple testing set are shown in Fig.6. This result indicates that the

simplified task and the compact output of the network reduce the demand for model capacity and training data, the resulting stableness and efficiency in training finally lead to advantageous generalization to other domains, which shows promising potential for application.

Method	Accuracy(%)	FP	FN
SCNN[12]	0.29	<b>0.0068</b>	1.0
SIM-CycleGAN+ERFNet[9]	62.58	0.9886	0.9909
UFNet[16]	65.53	0.5680	0.6546
PINet(4H)[6]	36.31	0.4886	0.8988
<b>FOOLane(ours)</b>	<b>84.36</b>	0.3964	<b>0.3841</b>

Table 5. Evaluation of generalization ability of different methods from CULane training set to TuSimple testing set.

## 5. Conclusion and Future Work

In this paper, we propose a local-based bottom-up solution for lane detection. Experimental results show the key-point estimation and the coarse-to-fine refinement strategy circumvent the influence from ambiguous and noisy pixels, effectively improves the accuracy of curve geometry. More importantly, the principle of focusing on local geometry and the bottom-up pipeline have been proved to be particularly resultful, which significantly simplifies the task by reducing the dimension of the output of CNN and is believed to be the principal cause of the excellent performance and generalization capacity.

The proposed method also shows superiority in adaptation to the rapid evolution of neural networks for performance and efficiency. We have plan to incorporate more powerful architectures into **FOOLane** framework, e.g. the ones with self-attention mechanism, to improve the performance further. We also want to use **FOOLane** on MindSpore<sup>1</sup>, which is a new deep learning computing framework. These problems are left for future work.

<sup>1</sup><https://www.mindspore.cn/>

## References

- [1] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017. 2, 3
- [2] Zhenpeng Chen, Qianfei Liu, and Chenfan Lian. Pointlanenet: Efficient end-to-end cnns for accurate real-time lane detection. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2563–2568. IEEE, 2019. 2, 6
- [3] Netalee Efrat, Max Bluvstein, Noa Garnett, Dan Levi, Shaul Oron, and Bat El Shlomo. Semi-local 3d lane detection and uncertainty estimation. *arXiv preprint arXiv:2003.05257*, 2020. 2, 3
- [4] Mohsen Ghafoorian, Cedric Nugteren, Nóra Baka, Olaf Booij, and Michael Hofmann. El-gan: Embedding loss driven generative adversarial networks for lane detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 3, 6
- [5] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. Learning lightweight lane detection cnns by self attention distillation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1013–1021, 2019. 3, 4, 6
- [6] Yeongmin Ko, Jiwon Jun, Donghwuy Ko, and Moongu Jeon. Key points estimation and point instance segmentation approach for lane detection. *arXiv preprint arXiv:2002.06604*, 2020. 3, 6, 8
- [7] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision*, pages 734–750, 2018. 4
- [8] Seokju Lee, Junsik Kim, Jae Shin Yoon, Seunghak Shin, Oleksandr Bailo, Namil Kim, Tae-Hee Lee, Hyun Seok Hong, Seung-Hoon Han, and In 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*, pages 1947–1955, 2017. 3
- [9] Tong Liu, Zhaowei Chen, Yi Yang, Zehao Wu, and Haowei Li. Lane detection in low-light conditions using an efficient data enhancement: Light conditions style transfer. *arXiv preprint arXiv:2002.01177*, 2020. 3, 6, 8
- [10] Davy Neven, Bert De Brabandere, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Towards end-to-end lane detection: an instance segmentation approach. In *2018 IEEE intelligent vehicles symposium (IV)*, pages 286–291. IEEE, 2018. 2, 3, 6
- [11] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in neural information processing systems*, pages 2277–2287, 2017. 3
- [12] Xingang Pan, Jianping Shi, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Spatial as deep: Spatial cnn for traffic scene understanding. *arXiv preprint arXiv:1712.06080*, 2017. 2, 3, 4, 6, 8
- [13] George Papandreou, Tyler Zhu, Liang-Chieh Chen, Spyros Gidaris, Jonathan Tompson, and Kevin Murphy. Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–286, 2018. 3
- [14] Jonah Philion. Fastdraw: Addressing the long tail of lane detection by adapting a sequential prediction network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11582–11591, 2019. 3, 6
- [15] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4929–4937, 2016. 2, 3
- [16] Zequn Qin, Huanyu Wang, and Xi Li. Ultra fast structure-aware deep lane detection. *arXiv preprint arXiv:2004.11757*, 2020. 2, 3, 6, 8
- [17] Eduardo Romera, José M Alvarez, Luis M Bergasa, and Roberto Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):263–272, 2017. 4, 6
- [18] TuSimple. Tusimple benchmark. <https://github.com/TuSimple/tusimple-benchmark> Accessed October, 2019. 6
- [19] Hang Xu, Shaoju Wang, Xinyue Cai, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Curvelane-nas: Unifying lanesensitive architecture search and adaptive point blending. In *The European Conference on Computer Vision (ECCV)*, 2020. 2
- [20] Seungwoo Yoo, Hee Seok Lee, Heesoo Myeong, Sungrack Yun, Hyoungwoo Park, Janghoon Cho, and Duck Hoon Kim. End-to-end lane marker detection via row-wise classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 1006–1007, 2020. 3, 6
- [21] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018. 4
- [22] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 4