SRLane+: Improved Sketch and Refinement Method for Lane Detection

Hang Liu[1][0009-0002-0873-0051] and Jinhua Xu[2][0000-0002-8450-3144]

School of Computer Science and Technology, East China Normal University, Shanghai 200062, China
jhxu@cs.ecnu.edu.cn

Abstract. Lane detection is a significant task with applications in autonomous driving tasks such as adaptive cruise control, lane departure warning, and lane-keep assistance. Convolutional neural networks (CNN) and transformers have been used for lane detection and achieved great performance. However, there are still challenges under complex scenarios, such as crowded or dazzling conditions. Utilizing the global and slender shape of lanes, line anchor-based methods have attracted much attention. Sketch and refinement (SRLane) is a two-stage anchor-based method for lane detection, composed of proposal generation stage (Sketch) and Refinement stage. This paper improves SRLane on both stages. At the first stage, position embedding is introduced and fused with the multi-scale feature maps to estimate local direction map more accurately for anchor proposal generation. At the second stage, a new fine regressor is proposed, which includes a multi-scale feature fusion module and a transformer decoder to further refine the lanes utilizing the context information. The whole method adopts an end-to-end training mechanism. Experiments are conducted on the CULane and Tusimple datasets. The results show that the proposed method achieves 80.35% F1 score on the CULane dataset, outperforming existing methods.

Keywords: Lane Detection, Feature Fusion, Transformer Decoder, Line Anchor.

1 Introduction

In autonomous driving systems, lane detection is a fundamental task to ensure safe driving and achieve advanced driver assistance. Convolutional neural networks (CNN) and transformers have been used for lane detection and achieved great performance. However, existing lane detection methods still face challenges in complex scenes.

Deep learning-based lane detection methods can be broadly categorized into three types, segmentation-based method, anchor-based method, parametric curve-based methods. Anchor-based methods [1]–[4] leverage strong prior knowledge of lane line shapes to model entire lanes, alleviating issues related to the local receptive field and

the lack of visual cues. Therefore, anchor-based methods have attracted great attention. There are two stages in the anchor-based method, anchor initialization and regression. At the first stage, the demand for high quality proposals results in a dilemma between efficiency and adaptability. Typically, sparse proposals are computationally efficient but have limited fitting ability, while dense proposals offer better fitting ability but can be more computationally expensive. At the second stage, the line geometry of anchors causes hard regression when dealing with lanes with complex topologies, such as curve lanes and branch lanes. To address the above issues, Sketch and Refine (SRLane) [5] was proposed, which first predicts the initial proposals (sketch) of the lanes from a local direction map, and then uses a segmented association optimization strategy to fine-tune the sketch. Since the line anchors are broken into segments, the curved lane detection is improved.

However, SRLane still has two key bottlenecks. First, at the Sketch stage, the anchor initialization relies solely on multi-scale visual features, lacking explicit spatial position information. This often results in suboptimal initial proposals, particularly for distant or occluded lanes. Second, at the Refinement stage, the model adjusts proposals using only local features, without fully exploiting global contextual information. This limits its ability to handle challenging scenarios like sharp curves, dazzling light, and heavy occlusions.

To address these issues, we propose SRLane+, an improved sketch and refinement framework. At the first stage, we introduce a position embedding to inject explicit spatial priors into multi-scale feature maps, enabling more accurate anchor initialization. At the second stage, we add a novel fine regression step after the coarse step of SRLane. This step consists of a Multi-Scale Feature Fusion (MSFF) module and a transformer decoder. The MSFF module adaptively fuses low-level and high-level features. The decoder then refines the lane proposals by capturing both global context and inter-lane relationships. Experiments on CULane and TuSimple datasets demonstrate that SRLane+ significantly outperforms the baseline, achieving an F1 score of 80.35% on CULane.

Our main contributions can be summarized as follows:

- We introduce a position embedding for anchor initialization.
- We introduce a fine regression step, in which a multi-scale feature fusion module (MSFF) and a decoder are designed to utilize the context information.
- We conduct experiments on two publicly available datasets. The proposed SRLane+ performs better than the baseline method and achieves better or comparable results with state-of-the-art models.

2 Related work

Lane detection aims to detect lane markings in images captured by onboard cameras and provide 2D position information for lane lines. Current deep learning-based lane detection methods can be broadly categorized into three types.

2.1 Segmentation-based method

This kind of method regards lane detection as a pixel-level classification task, and relies on the encoder-decoder architecture to generate high-resolution segmentation maps. In [6], an instance segmentation method is introduced, which uses a clustering algorithm to dynamically ascertain the number of lane lines to be detected. In SCNN [7], the prior shape information of lane lines is leveraged to enhance the model's capacity for lane detection. The advantage of segmentation methods is that it has theoretical compatibility with lanes of any topological structure. But it faces limitations such as excessive computational load and complex post-processing.

2.2 Anchor-based method

The anchor-based methods can be divided into two categories, row anchor-based and line anchor-based. Ultra Fast Lane Detection (UFLD) is a row anchor-based method [8], [9]. A lane is represented by the coordinates on a series of predefined row anchors, then the lane detection task is formulated as an ordinal classification problem to get the coordinates of lanes. Considering the shape of lanes, the line anchor is first proposed in [1], and line proposals are utilized as references to locate accurate lane curves, which forces the model to learn the global feature representation of the entire lines. LaneAtt [2] introduces an attention mechanism to aggregate global information and effectively handle complex scenarios such as occlusion, missing lane markings, or changes in illumination. CLRNet [3]and CLRERNet [10] detect lanes with high-level semantic features for contextual information, then performs refinement based on low-level features for local detailed lane information. FLAMNet [4] is a hybrid model which integrates attention modules in a CNN model to establish global long-distance dependencies and to expand the receptive field of line anchors in the horizontal direction.

Benefiting from the strong prior characteristics of anchors, anchor-based representation can effectively handle the problem of absent visual clues. Our method is based on line anchors. We focus on the anchor initialization and anchor regression.

2.3 Parametric curve-based methods

Parametric curve-based methods model lane lines using a suitable mathematical formulation, then fit lane lines based on the model. Parametric curve-based methods can naturally learn holistic lane line representations [11]. In PolyLaneNet [12] , a lane is represented as a polynomial curve and the parameters are directly regressed. In LSTR [13], a cubic curve is used to approximate a lane line on a flat road surface and a Transformer is proposed to estimate the parameters of the lane shape. In [14] a parametric Bézier curves are employed to model the lane lines, enabling more flexible manipulation of lane line shapes with just a few simple points. Curve fitting is sensitive to noise and difficult to adapt to curvature mutation or bifurcation structure.

In summary, while anchor-based methods achieve strong performance by leveraging lane shape priors, a key challenge remains: how to generate high-quality initial anchors and effectively refine them using contextual information. SRLane made a significant step by proposing a two-stage sketch-and-refine pipeline. Our work, SRLane+, directly builds upon and improves this framework. We focus on enhancing the spatial sensitivity of the Sketch stage via position embedding, and significantly boosting the refinement power via multi-scale feature fusion and a transformer decoder. This distinguishes our approach from prior works that often focus on only one of the two stages.

3 Method

This section systematically presents the proposed SRLane+ framework. We first define the lane and anchor representations used throughout the paper (Sec. 3.1). Then, we introduce the overall two-stage architecture (Sec. 3.2). Specifically, we detail the position embedding module designed to improve the Sketch stage, followed by the MSFF module and the transformer decoder which form the core of our newly proposed Fine Regression step in the Refinement stage. Finally, we describe the multi-task loss function for end-to-end training (Sec. 3.3).

3.1 Lane representation

As in [1], [2], a lane is represented as a sequence of 2D points,

$$P = \left\{(x_0, y_0), (x_1, y_1), \dots, \left(x_{N_p-1}, y_{N_p-1}\right)\right\},$$

where the y_i are sampled at equal intervals in the vertical direction of the image, that is:

$$y_i = \frac{i}{(N_p-1)H}. \tag{1}$$

Here H represents the image height, and the x_i is the x coordinate associated with the corresponding y-value.

We adopt line anchors in this paper and an anchor can be regarded as a ray originating from the starting point (x_s, y_s) and having a direction angle θ (the angle with the x-axis), with its parameterized representation being (x_s, y_s, θ). This line anchor aids the network in locating lane lines precisely. Based on the N_p sampled points, an anchor is represented as

$$P^A = \left\{(x_0^A, y_0), \dots, \left(x_{N_p-1}^A, y_{N_p-1}\right)\right\}.$$

Then the lane can be represented based on the anchor as:

$$P = \left\{(x_0^A + \Delta x_0, y_0), \ldots, \left(x_{N_p-1}^A + \Delta x_{N_p-1}, y_{N_p-1}\right)\right\}. \quad (2)$$

In the first stage, we predict the starting points and the direction of the anchor and obtain the anchor point positions (x_i^A, y_i). In the second stage, we predict the offset Δx_i of each point relative to the line anchor, and obtain the lane points using eq.2.

3.2 Architecture

The overall architecture is shown in Fig.1. It is a two-stage framework modified from the SRLane [5]. Given an input image $I \in R^{H\times W\times 3}$, the backbone is used to extract multi-scale features from *I*. As in [5], ResNet is used as the backbone. At the first stage, anchor proposals are generated from the local direction map. Different from the sketch stage of SRLane, position embedding is fused with the feature maps to improve the initial anchor proposals. At the second stage, a fine step is added in the refine stage of SRLane (coarse step in Fig.1), in which multi-scale feature fusion module (MSFF) and a decoder are designed to further refine the lane prediction.

1) Position Embedding: In the original SRLane, the Sketch stage predicts a local direction map purely from the backbone's feature maps. We argue that these feature maps, while semantically rich, lack an explicit sense of spatial location, which is crucial for determining the precise starting point and the direction of a lane anchor. Therefore, we introduce an extremely simple yet effective positional embedding strategy. Specifically, the feature maps from the backbone are combined with multi-scale positional embedding (PE). Then the direction map is estimated through convolutional layers. The multi-scale positional embedding is generated as follows: For feature maps of various resolutions (e.g., 40×100, 20×50, 10×25) from the backbone, corresponding positional embeddings are independently constructed for each scale. The position embeddings are normalized to the range [0, 1].

$$\mathbf{PE}_k(i,j) = [i/h_k, j/w_k]. \quad (3)$$

Here(i,j) is the spatial coordinate, and $h_k$and w_k are the dimensions of the *k*-th feature map $\mathbf{F}_k$. Then the position embeddings are integrated into the feature maps by point-wise addition.

$$\mathbf{F}_k^p = \mathbf{F}_k \oplus \mathbf{PE}_k. \quad (4)$$

In this method, PE is applied independently to each level of the original feature maps, with the goal of preserving the original semantic features. In the low-level feature maps, this method enables end-to-end propagation of detailed information. By encoding the locations of details, it effectively improves the localization accuracy of lane edges in the close range. In the high-level feature maps, encoding global positional information enhances the continuity of lanes in the distant range.

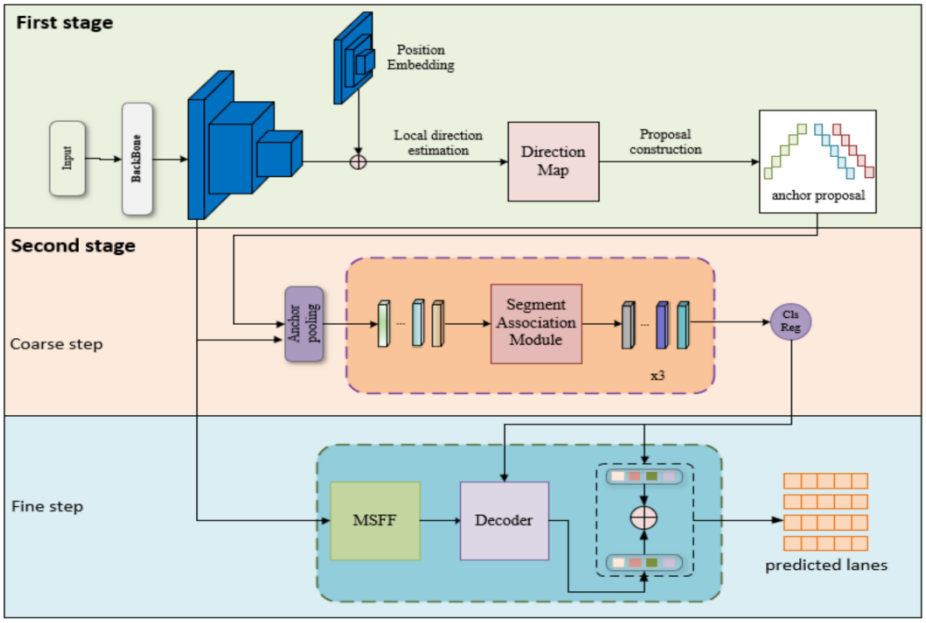

Fig. 1. Overview of the proposed method.

2) Multi-Scale Feature Fusion (MSFF) Module: The final two stage feature maps of the backbone, $F_3 \in R^{\frac{1}{4}H\times\frac{1}{4}W\times C}$ and $F_4 \in R^{\frac{1}{8}H\times\frac{1}{8}W\times C}$, are selected to represent low-level and high-level features, respectively, here C denotes the feature dimension, H and W are the height and the width of the input image respectively. The MSFF module fuses the high-level features F_4 and the low-level features F_3. Initially, the MSFF module adjusts the high-level features to the same spatial size as the low-level features through bilinear interpolation.

$$F_4' = \text{Interpolate}\big(F_4, \text{size} = (H, W)\big). \tag{5}$$

Then the module flattens the spatial dimensions of F_3 and F_4 , converting each feature map into a sequence:

$$F_4^{\text{flat}} = \text{reshape}(F_4') \in \mathbb{R}^{B\times(H\times W)\times C}, \tag{6}$$

$$F_3^{\text{flat}} = \text{reshape}(F_3) \in \mathbb{R}^{B\times(H\times W)\times C}. \tag{7}$$

Use the low-level feature F_3 as the Query and the high-level feature F_4 as the Key and the Value for cross attention calculation:

$$CrossAtt(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{C}}\right)V. \tag{8}$$

Then, residual connections and LayerNorm are applied.

$$F_{out} = LayerNorm\left(F_3^{flat} + CrossAtt(Q,K,V)\right). \tag{9}$$

The cross attention enables the model to effectively integrate features from different levels. As shown in the experiments, the lane detection accuracy from the fused feature is improved.

3) Decoder: We design a decoder as the final refinement engine, consisting of self-attention and cross-attention layers, as shown in Fig. 2. The cross-attention, which uses the lane proposals as queries and the fused feature F_{out} from the MSFF module as keys and values, serves as a powerful feature selector. It dynamically pools the most relevant multi-scale contextual information for each anchor point, thereby integrating useful scene context into the proposals. The calculation process of self-attention follows the standard Query-Key-Value pattern,

$$SelfAtt(X) = softmax\left(\frac{(XW_q)(XW_k)^T}{\sqrt{C}}\right)(XW_v). \tag{10}$$

where X represents the proposal feature. W_q, W_k and $W_q \in \mathbb{R}^{C\times C}$ are the parameters to be learned.

Then residual connections and layer normalization are applied.

$$X' = \text{LayerNorm}\big(X + SelfAtt(X)\big). \tag{11}$$

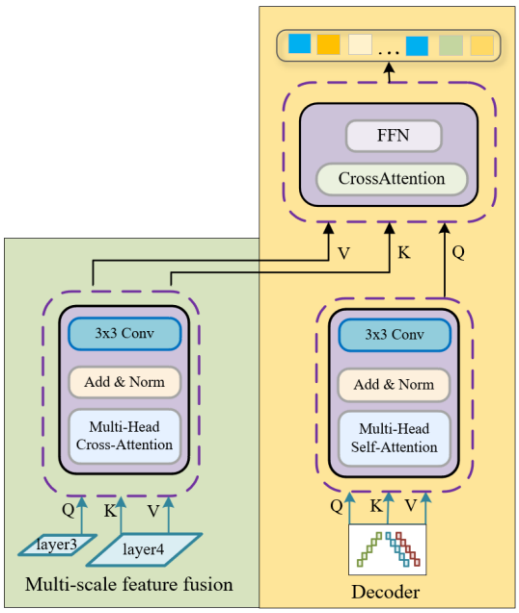

Fig.2. Architectures of the MSFF module and the decoder of the fine step.

As shown in Fig.1, the predicted offset of the decoder is added to the prediction of the coarse step to generate the refined lane prediction.

3.3 Loss functions

The loss function of the lane detection model adopts a multi-task learning framework, including classification loss, regression loss and IoU loss to form an end-to-end optimization objective. The overall loss function can be expressed as:

$$L_{total} = L_{cls} + \lambda_1 L_{l1} + \lambda_2 L_{iou} . \tag{12}$$

Each component of the loss function works collaboratively to ensure optimal model performance across multiple dimensions, including classification accuracy, localization precision, and structural rationality. L_{cls} employs Focal Loss to distinguish between the foreground (lane lines) and the background. L_{l1} is utilized to optimize the prediction of the lane line's starting point coordinates and the direction. Line IoU loss L_{iou} [3] is employed to optimize the spatial overlap between the predicted lane lines and the ground truths. We set $\lambda_1 = 1.0$ and $\lambda_2 = 2.0$ in experiments.

4 Experiments

In this section, we first introduce implementation details, two public datasets used in the experiments and evaluation metrics, then present the experimental results and compare them with existing methods. Finally we discuss the ablation study and demonstrate the effectiveness of each proposed component.

4.1 Implementation Details

The models were implemented on PyTorch. ResNet18 and ResNet34 are selected as the backbone network. Multi-scale feature maps from the last three(two) stages are used in the first(second) stage. Input images are adjusted to a fixed size (800 × 320). Batch size is set to 24. In the optimizing process, we use the AdamW optimizer with an initial learning rate of 10^{-3} and cosine decay learning rate strategy. Random flipping, affine transformation, color jitter and JPEG compression techniques are used for Data augmentation to improve the robustness and generalization ability of the model.

4.2 Datasets

The experiments are conducted on two widely used lane detection datasets CULane and TuSimple. CULane is a large-scale dataset in the field of lane detection, including 88,880 training frames, 9,675 validation frames and 34,680 test frames. The dataset covers a variety of complex road scenes, including different weather conditions, illumination changes, the degree of wear of road marking lines, and a variety of lane types such as straight, turning, bifurcation, etc. Therefore, the CULane dataset is very suitable for evaluating the performance of lane detectors in diverse real-world driving scenarios. The TuSimple dataset contains 3,626 training images and 2,782 test images, mainly collected in highway scenes. Compared with CULane, TuSimple focuses more on evaluating the performance of the algorithm in the highway environment, where the lane lines are relatively regular, but there are still some challenges, such as the distinction between the dashed and solid lines of the lane line, the curvature change of the lane line, etc.

4.3 Evaluation metrics

In the experiment, accuracy (Acc), $F1$ score, false positive (FP) and false negative (FN) are used as the main evaluation indicators. $F1$ score is the harmonic average of Precision and Recall, which can comprehensively reflect the accuracy and integrity of the model when detecting lane lines. Acc and $F1$ are defined as:

$$Acc = \frac{TP+TN}{TP+FN+TN+FP}, \tag{13}$$

$$F_1 = 2 \times \frac{Precision \times \text{Recall}}{Precision + \text{Recall}}, \tag{14}$$

$$Precision = \frac{TP}{TP+FP}, \tag{15}$$

$$Recall = \frac{TP}{TP+FN}. \tag{16}$$

Table 1. Experimental results on the CULane dataset. Top panel and bottom panel include keypoint-based and proposal-based methods respectively. Results of SRLane* are from our re-implementation.

Method	Backbone	F1@50	Normal	Crowded	Dazzle	Shadow	No-line	Arrow	Curve	Night	Cross	GFlops
LaneAF [15]	DLA34	77.4	91.8	75.6	71.8	79.1	51.4	86.9	72.0	73.0	1360	
FOLOLane [16]	ERFNet	78.8	92.7	77.8	75.2	79.3	52.1	89.0	69.4	74.5	1569	-
GANet [17]	ResNet18	78.8	93.2	77.2	71.2	77.9	53.6	89.6	75.9	72.8	1240	
RCLane [18]	SegB0	79.5	93.4	77.9	73.3	80.3	53.8	89.0	75.7	74.3	1298	
Laneformer [19]	ResNet18	71.71	88.60	69.02	64.07	65.02	45.00	81.55	60.46	64.76	**25**	13.8
LaneATT [2]	ResNet18	75.13	91.17	72.71	65.82	68.03	49.13	87.82	63.75	68.58	1020	9.3
O2SFormer [20]	ResNet18	76.07	91.89	73.86	70.40	74.84	49.83	86.08	68.68	70.74	2361	15.21
SGNet [21]	ResNet18	76.12	91.42	74.05	66.89	72.17	50.16	87.13	67.02	70.67	1164	-
CondLane [22]	ResNet18	78.14	92.87	75.79	70.72	80.01	52.39	89.37	72.40	73.23	1569	10.2
AQ-LTR [23]	ResNet18	79.31	93.09	78.23	73.73	80.66	53.72	89.98	71.71	74.43	1927	19.14
CLRNet [3]	ResNet18	79.58	93.30	78.33	73.71	79.66	53.14	**90.25**	71.56	75.11	1321	11.9
SRlane [5]	ResNet18	78.9	93.5	77.8	71.6	78.8	52.1	90.2	74.7	74.7	1365	
SRLane*	ResNet18	79.56	93.31	78.36	73.41	82.20	55.50	89.93	74.99	74.39	1348	11.37
SRLane+(ours)	ResNet18	**80.14**	**93.80**	**78.52**	**75.31**	**82.24**	**56.05**	89.52	**77.71**	**75.55**	1437	16.01
UFLDv1 [8]	ResNet34	72.30	90.30	72.30	66.30	68.40	49,80	83.70	65.20	67.70	1427	16.9
Laneformer [19]	ResNet34	74.70	90.74	72.31	69.12	71.57	47.37	85.07	65.90	67.77	**26**	23.0
UFLDv2 [9]	ResNet34	75.90	92.70	77.80	75.20	79.30	52.10	89.00	69.40	74.50	1569	-
LaneATT [2]	ResNet34	76.68	92.14	75.03	66.47	78.15	49.39	88.38	67.72	70.72	1330	18.0
O2SFormer [20]	ResNet34	77.03	92.50	75.25	70.93	77.72	50.97	87.63	68.10	72.88	2749	25.05
SGNet [21]	ResNet34	77.27	92.07	75.41	67.75	74.31	50.90	87.97	69.65	72.69	1373	-
AQ-LTR [23]	ResNet34	79.52	93.43	78.46	75.60	81.84	54.13	90.22	71.74	74.72	2029	27.93
CLRNet [3]	ResNet34	79.73	93.49	78.06	74.57	79.92	54.01	90.59	72.77	75.02	1216	21.5
FLAMNet [4]	ResNet34	80.15	93.61	78.35	73.64	81.31	53.68	90.67	73.38	75.50	982	30.1
SRLane+(ours)	ResNet34	**80.35**	**93.84**	**79.08**	**75.51**	**82.35**	**56.54**	**90.72**	**77.75**	**75.64**	1475	25.47

Among them, the precision measures how many of the samples predicted by the model as lane lines are real lane lines; the recall rate measures how many of all actual lane lines are correctly detected by the model. False Positive (FP) indicates that the model incorrectly predicts non-lane areas as the lanes. The low FP value means that the model

has high specificity when detecting lane lines. False negative (FN) represents the number of actual lane lines that the model fails to detect correctly. The lower FN value indicates that the model has higher sensitivity and can identify more real lane lines.

4.4 Results

We evaluate our proposed method on the CULane and TuSimple datasets and compare it with other mainstream lane detection methods.

Results on Culane: As showed in Table 1, our method achieves F1 score of 80.14 (80.35) using ResNet18 (ResNet34) as the backbone, which are higher than other networks with the same backbone. Compared with the baseline model SRLane, ResNet18 has improved $F1$ scores in total and in most scenarios. Especially, the $F1$ score is increased from 74.99 to 77.71 for the curve lanes, and from 73.41 to 75.31 for the dazzle scenes, indicating that our method performs better in these challenging scenarios.

ResNet34 backbone further improves the performance of the model, achieving the highest $F1$ score 80.35 in all scenarios. This shows that increasing the network depth can improve the detection accuracy to a certain extent with more computational cost.

Table 2. Experimental results on the TuSimple dataset.

Method	Backbone	Acc(↑)	F1(↑)	FP(↓)	FN(↓)
LaneAF [15]	DLA34	95.62	96.49	2.80	4.18
GANet [17]	Res18	95.95	97.71	1.97	2.62
RCLane [18]	SegB0	96.58	97.64	2.21	2.57
FOLOLane [16]	ERFNet	96.92	96.59	4.47	2.28
CondLane [22]	ResNet18	95.48	97.01	**2.18**	3.80
LaneATT [2]	ResNet18	95.57	96.71	3.56	3.01
UFLDv2 [9]	ResNet18	95.65	96.16	3.06	4.61
CLRNet [3]	ResNet18	96.84	**97.89**	2.28	1.92
SRLane [5]	ResNet18	**96.85**	97.66	2.80	**1.85**
SRLane* [5]	ResNet18	95.47	97.07	2.38	3.50
SRLane+(ours)	ResNet18	96.36	97.45	2.32	2.77
CondLane [22]	ResNet34	95.37	96.98	**2.20**	3.82
LaneATT [2]	ResNet34	95.63	96.77	3.53	2.92
UFLDv1 [8]	ResNet34	95.86	88.02	18.91	3.75
UFLDv2 [9]	ResNet34	95.72	88.14	18.54	3.60
CLRNet [3]	ResNet34	96.87	97.82	2.27	2.08
FLAMNet [4]	ResNet34	**96.94**	**97.83**	2.43	**1.89**
LaneFormer [19]	ResNet34	96.56	95.60	5.39	3.37
SRLane* [5]	ResNet34	95.45	97.10	2.37	3.47
SRLane+(ours)	ResNet34	96.47	97.32	2.26	3.11

Fig. 3. Visualization of the detection results of SRLane and our method on CULane test set.

Results on TuSimple: We evaluate our proposed method on the TuSimple dataset and compare it with other mainstream lane detection methods in Table 2. Due to the limited scale and homogeneous highway scenarios in the TuSimple dataset, the performance gap between different methods remains marginal. Our method achieves F1 scores of 96.36 and 97.47, FP of 2.32 and 2.26 using ResNet18 and ResNet34 backbone respectively, which are comparable with the baseline SRLane and the SOTA methods.

Table 3. Ablation experimental results on the CULane dataset.

Baseline	PE	MSFF	Decoder	F1@50
√				79.56
√	√			79.80
√	√		√	80.02
√	√	√	√	80.14

4.5 Ablation study

In Section III, we have described our proposed module including PE, MSFF and the decoder. In this section, we conduct detailed ablation experiments on the CULane dataset to verify the effectiveness of each module. We used ResNet18 as the backbone and the re-implemented SRLane as our baseline, and progressively incorporated the PE,

MSFF and the decoder to evaluate the contribution of each component. The ablation study results are presented in Table 3.

Position Embedding (PE): By integrating position embed- ding to the feature maps, the F1 score is improved from 79.56 to 79.80, indicating PE can improve the initial proposals of the first stage.

PE+Decoder: By integrating PE and the decoder, the F1 score is further improved to 80.02, indicating the decoder of the fine regression can improve the detection results.

PE+MSFF+Decoder: When further integrating MSFF, the model achieves the best F1 score 80.14.

The ablation experiment results indicate that the proposed modules are effective. The position embedding provides crucial spatial position information for the anchor generation. MSFF enhances the model's ability of feature extraction. The decoder integrates context information from the fused feature with the anchors, improving the detection performance. The combination of all three modules achieves the best results on the CULane dataset.

4.6 Visualization of the lane detection results

The lane detection results of our method and SRLane are visualized and compared in Fig.3. In the crowd and no-line scenes, SRLane fails to detect the lanes occluded by the vehicles. In the light, shadow and night scenes, SRLane fails to detect the lane affected by the light and the shadow. In the curve scene, SRLane fails to predict the curve shape of the lane and misses one lane. Our method succeeds in all above scenes. In the arrow scene, our method fails to detect the second lane from the left while SRLane fails on two lanes. Visualization results indicate that our proposed method performs better in these challenging scenarios, especially on the curve lanes, consistent with the quantitative comparison in Table I.

4.7 Model Complexity

As show in Table 1, compared with the baseline SRLane, GFLOPS is increased from 11.37 to 16.01. The proposed method improves detection accuracy with more computational cost from the fine regression step.

5 Conclusion

In this paper, we proposed an improved sketch and refinement approach based on a two-stage lane detection framework. The framework fuses position embedding and multi-scale features in the first stage to generate preliminary lane proposals. In the second stage, these proposals are refined using self-attention and cross-attention mechanisms to enhance detection accuracy. Our method achieves $F1$-score of 80.35 on the CULane dataset, outperforming state-of-the-art methods. Future work could integrate 3D geometric information and dynamic lane generation strategies to improve the capability in handling complex scenarios.

References

1. Li, X., Li, J., Hu, X., Yang, J.: Line-cnn: End-to-end traffic line detection with line proposal unit. IEEE Transactions on Intelligent Transportation Systems 21(1), 248–258 (2019)
2. Tabelini, L., Berriel, R., Paixao, T.M., Badue, C., De Souza, A.F., Oliveira-Santos, T.: Keep your eyes on the lane: Real-time attention-guided lane detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 294–302 (2021)
3. Zheng, T., Huang, Y., Liu, Y., Tang, W., Yang, Z., Cai, D., He, X.: Clrnet: Cross layer refinement network for lane detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 898–907 (2022)
4. Ran, H., Yin, Y., Huang, F., Bao, X.: Flamnet: A flexible line anchor mechanism network for lane detection. IEEE Transactions on Intelligent Transportation Systems (2023)
5. Chen, C., Liu, J., Zhou, C., Tang, J., Wu, G.: Sketch and refine: Towards fast and accurate lane detection. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 1001–1009. Springer (2024).
6. Neven, D., De Brabandere, B., Georgoulis, S., Proesmans, M., Van Gool, L.: Towards end-to-end lane detection: an instance segmentation approach. In: 2018 IEEE intelligent vehicles symposium (IV). pp. 286–291. IEEE (2018)
7. Pan, X., Shi, J., Luo, P., Wang, X., Tang, X.: Spatial as deep: Spatial cnn for traffic scene understanding. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
8. Qin, Z., Wang, H., Li, X.: Ultra fast structure-aware deep lane detection. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16. pp. 276–291. Springer (2020)
9. Qin, Z., Zhang, P., Li, X.: Ultra fast deep lane detection with hybrid anchor driven ordinal classification. IEEE transactions on pattern analysis and machine intelligence (2022)
10. Honda, H., Uchida, Y.: Clrernet: improving confidence of lane detection with laneiou. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1176–1185 (2024)

11. Luo, X., Huang, Y., Cui, J., Zheng, K.: Deep learning-based lane detection for intelligent driving: A comprehensive survey of methods, datasets, challenges and outlooks. Neurocomputing 650, 130795 (2025)
12. Tabelini, L., Berriel, R., Paixao, T.M., Badue, C., De Souza, A.F.Oliveira-Santos, T.: Polylanenet: Lane estimation via deep polynomial regression. In: 2020 25th International Conference on Pattern Recognition (ICPR). pp. 6150–6156. IEEE (2021)
13. Liu, R., Yuan, Z., Liu, T., Xiong, Z.: End-to-end lane shape prediction with transformers. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. pp. 3694–3702 (2021)
14. Feng, Z., Guo, S., Tan, X., Xu, K., Wang, M., Ma, L.: Rethinking efficient lane detection via curve modeling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 17062–17070 (2022)
15. Abualsaud, H., Liu, S., Lu, D.B., Situ, K., Rangesh, A., Trivedi, M.M.: Laneaf: Robust multi-lane detection with affinity fields. IEEE Robotics and Automation Letters 6(4), 7477–7484 (2021)
16. Qu, Z., Jin, H., Zhou, Y., Yang, Z., Zhang, W.: Focus on local: Detecting lane marker from bottom up via key point. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14122–14130 (2021)
17. Wang, J., Ma, Y., Huang, S., Hui, T., Wang, F., Qian, C., Zhang, T.: A keypoint-based global association network for lane detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1392–1401 (2022)
18. Xu, S., Cai, X., Zhao, B., Zhang, L., Xu, H., Fu, Y., Xue, X.: Rclane: Relay chain prediction for lane detection. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) Computer Vision – ECCV 2022. pp. 461–477. Springer Nature Switzerland, Cham (2022)
19. Han, J., Deng, X., Cai, X., Yang, Z., Xu, H., Xu, C., Liang, X.: Laneformer: Objectaware row-column transformers for lane detection. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36, pp. 799–807 (2022)
20. Zhou, K., Zhou, R.: End-to-end lane detection with one-to-several transformer (2023)
21. Su, J., Chen, C., Zhang, K., Luo, J., Wei, X., Wei, X.: Structure guided lane detection. In: Zhou, Z.H. (ed.) Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21. pp. 997–1003. International Joint Conferences on Artificial Intelligence Organization (8 2021)
22. Liu, L., Chen, X., Zhu, S., Tan, P.: Condlanenet: a top-to-down lane detection framework based on conditional convolution. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3773–3782 (2021)
23. Xing, Y., Xu, J.: Anchor query based transformer for lane detection. In: 2024 International Joint Conference on Neural Networks (IJCNN). pp. 1–8 (2024)