RCLane: Relay Chain Prediction for Lane Detection

Shenghua Xu[1*], Xinyue Cai[2*], Bin Zhao[1], Li Zhang[1**], Hang Xu[2],
Yanwei Fu[1], and Xiangyang Xue[1]

[1] Fudan University
[2] Huawei Noah's Ark Lab

Abstract. Lane detection is an important component of many real-world autonomous systems. Despite a wide variety of lane detection approaches have been proposed, reporting steady benchmark improvements over time, lane detection remains a largely unsolved problem. This is because most of the existing lane detection methods either treat the lane detection as a dense prediction or a detection task, few of them consider the unique topologies (*Y-shape, Fork-shape, nearly horizontal lane*) of the lane markers, which leads to sub-optimal solution. In this paper, we present a new method for lane detection based on *relay chain* prediction. Specifically, our model predicts a segmentation map to classify the foreground and background region. For each pixel point in the foreground region, we go through the forward branch and backward branch to recover the whole lane. Each branch decodes a transfer map and a distance map to produce the direction moving to the next point, and how many steps to progressively predict a relay station (next point). As such, our model is able to capture the keypoints along the lanes. Despite its simplicity, our strategy allows us to establish new state-of-the-art on four major benchmarks including *TuSimple, CULane, CurveLanes* and *LLAMAS*.

Keywords: Lane detection, relay chain.

1 Introduction

Lane detection, the process of identifying lanes as approximated curves, is a fundamental step in developing advanced autonomous driving system and plays a vital role in applications such as driving route planning, lane keeping, real-time positioning and adaptive cruise control.

Early lane detection methods [14,29,36,3,8,9,10,11] usually extract hand-crafted features and cluster foreground points on lanes through post-processing. However, traditional methods can not detect diverse lanes correctly for so many

---

[*] Equal contribution.
[**] Li Zhang (lizhangfd@fudan.edu.cn) is the corresponding author with School of Data Science, Fudan University.

Fig. 1. Challenging scenes (curve lanes, Y-shape lanes). The first row of (a) shows the ground truth while the second row is our predictions. The first row of (b) shows the result of segmentation-based methods that global shape of lane is not well fitted. While the second row of (b) shows proposal-based methods, can not depict local locations of Y-shape and curve lanes.

complicated scenes in driving scenarios. Thanks to the development of deep-learning, a wide variety of lane detection approaches based on convolution neural network(CNN) have been proposed, such as segmentation-based methods and proposal-based methods, reporting steady benchmark improvements over time.

Proposal-based methods initialize a fixed number of anchors directly and model global information focusing on the optimization of proposal coordinates regression. LaneATT [27] designs slender anchors according to long and thin characteristic of lanes. However, line proposals fail to generalize local locations of all lane points for curve lanes or lanes with more complex topologies. While segmentation-based methods treat lane detection as dense prediction tasks to capture local location information of lanes. LaneAF [1] focuses on local geometry to integrate into global results. However, this bottom-up manner can not capture the global geometry of lanes directly. In some cases such as occlusion or resolution reduction for points on the far side of lane, model performance will be affected due to the loss of lane shape information. Visualization results in Fig. 1(b) of these methods show their shortcomings. Lanes always span half or almost all of the image, these methods neglect this long and thin characteristic of lanes which requires networks to focus on the global shape message and local location information simultaneously. In addition, complex lanes such as Y-shape lanes and Fork-shape lanes are common in the current autonomous driving scenario, while existing methods often fail at these challenging scenes which are shown in Fig. 1(a).

To address this important limitation of current algorithms, we propose a more accurate lane detection solution in the unconstrained driving scenarios, which is called *RCLane* inspired by the idea of **Relay Chain** for focusing on local location and global shape information of lanes at the meanwhile. Each foreground point on the lane can be treated as a relay station for recovering the whole lane sequentially in a chain mode. Relay station construction is proposed for strengthening the model's ability of learning local message that is fundamental to describe flexible shapes of lanes. To be specific, we construct a transfer map

representing the relative location from current pixel to its two neighbors on the same lane. Furthermore, we apply bilateral prediction strategy aiming to improve generalization ability for lanes with complex topologies. Finally, we design global shape message learning module. Concretely, this module predicts the distance map describing the distance from each foreground point to the two end points on the same lane. The contributions of this work are as follows:

- We propose novel relay chain representation for lanes to model global geometry shape and local location information of lanes simultaneously.
- We introduce a novel pair of lane encoding and decoding algorithms to facilitate the process of lane detection with relay chain representation.
- Extensive experiments on four major lane detection benchmarks show that our approach beats the state-of-the-art alternatives, often by a clear margin and achieves real-time performance.

2   Related work

Existing methods for lane detection can be categorized into: segmentation-based methods, proposal-based methods, row-wise methods and polynomial regression methods.

Segmentation-based methods. Segmentation-based methods [7,12,13,20,21], typically make predictions based on pixel-wise classification. Each pixel will be classified as either on lane or background to generate a binary segmentation mask. Then a post-processing step is used to decode it into a set of lanes. But it is still challenging to assign different points to their corresponding lane instances. A common solution is to predict the instance segmentation mask. However, the number of lanes has to be predefined and fixed when using this strategy, which is not robust for real driving scenarios.

Proposal-based methods. Proposal-based methods [4,34,27], take a top-to-down pipeline that directly regresses the relative coordinates of lane shapes. Nevertheless, they always struggle in lanes with complex topologies such as curve lanes and Y-shaped lanes. The fixed anchor shape has a major flaw when regressing the variable lane shapes in some hard scenes.

Row-wise methods. Based on the grid division of the input image, row-wise detection approaches [6,22,23,35,15] have achieved great progress in terms of accuracy and efficiency. Generally, row-wise detection methods directly predict the lane position for each row and construct the set of lanes through post-processing. However, detecting nearly horizontal lanes which fall at small vertical intervals is still a major problem.

Polynomial regression methods. Polynomial regression methods [16,28] directly outputs polynomials representing each lane. The deep network is firstly used in [28] to predict the lane curve equation, along with the domains for these polynomials and confidence scores for each lane. [16] uses a transformer [31] to learn richer structures and context, and reframes the lane detection output as parameters of a lane shape model. However, despite of the fast speed polynomial
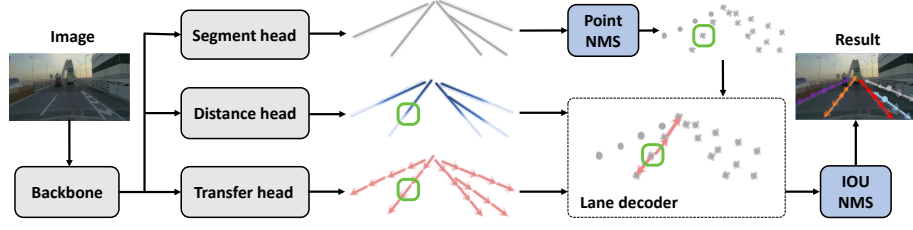
Fig. 2. Schematic illustration of proposed RCLane. Standard Segformer [33] is
used as backbone. The output head consists of three branches. The segment head predicts segmentation map (S). The distance head and the transfer head predict distance
map (D) and transfer map (T) respectively. Both kinds of maps contain forward and
backward parts. Then, Point-NMS is used for sparse segmentation results. All predictions are fed into the lane decoder (Fig. 5), to get final results.

regression methods achieve, there is still some distance from the state of the art
results.

3   Method

Given an input image $I \in \mathbb{R}^{H \times W \times C}$, the goal of RCLane is to predict a collection
of lanes $L = \{l_1, l_2, \cdots, l_N\}$, where N is the total number of lanes. Generally,
each lane l_k is represented as follows:

$$l_k = \{(x_1, y_1), (x_2, y_2), \cdots, (x_{N_k}, y_{N_k})\}, \tag{1}$$

The overall structure of our RCLane is shown in Fig. 2. This section will first
present the concept of lane detection with relay chain, then introduce the lane
encoder for relay station construction, followed by a lane decoder to attain curve
lanes. Finally, the network architecture and losses we adopt is detailed.

3.1   Lane detection with relay chain

Focusing on the combination of local location and global shape information to
detect lanes with complex topologies, we propose a novel lane detection method
RCLane with the idea of relay chain. Relay chain is a structure composed of
relay stations which are connected in a chain mode. Relay station is responsible
for data processing and transmitting it to adjacent stations, while chain is a
kind of structure that organizes these stations from an overall perspective. All
stations are associated to corresponding lane points respectively.

We design the structure of relay chain which is appropriate for combining local location and global geometry message in lane detection and propose RCLane
in this work. To be specific, each foreground point on the lane is treated as a
relay station and can extend to the neighbor points iteratively to decode the lane

in a chain mode. All foreground points are supervised by two kinds of message mentioned above. Moreover, the structure of chain has high flexibility to fit lanes with complex topologies.

Next, we will introduce the relay station construction and propose bilateral predictions for complex topologies and global shape message learning to explain how to detect lanes with the idea of *Relay Chain* progressively.

Relay station construction. Segmentation-based approaches normally predict all foreground points on lanes and cluster them via post-processing. [1] predicts horizontal and vertical affinity fields for clustering and associating pixels belonging to the same lane. [24] regresses a vector describing the local geometry of the curve that current pixel belongs to and refines shape further in the decoding algorithm. Nevertheless, they both fix the vertical intervals between adjacent points and decode lanes row-by-row from bottom to top. In fact, horizontal offsets are used for refining the position of current points while vertical offsets are for exploring the vertical neighbors of them. And the fixed vertical offsets can not adapt to the high degree of freedom for lanes. For example, they can only detect a fraction of the nearly horizontal lanes. Thus, we propose relay station construction module to establish relationships between neighboring points on the lane. Each relay station $p = (p_x, p_y)$ predicts offsets to its neighboring point $p^{next} = (p_x^{next}, p_y^{next})$ on the same lane with a fixed step length d as is shown in Eq. 2, 3 in two directions. And the deformation trend of lanes can be fitted considerably by eliminating vertical constraints. All relay stations are then connected to form a chain which is the lane exactly.

$$(p_x^{next}, p_y^{next}) = (p_x, p_y) + (\Delta x, \Delta y), \tag{2}$$

$$\Delta x^2 + \Delta y^2 = d^2. \tag{3}$$

Bilateral predictions for complex topologies. The current autonomous driving scenario contains lanes with complex topologies such as Y-shape and Fork-shape lanes, which can be regarded as that two lanes merges as the stem. One-way prediction can only detect one of lanes because it can only extend to one limb when starting from the stem of these lanes. We adopt a two-way detection strategy that splits the next neighboring point p^{next} into the forward point p^f and the backward point p^b. Points on different limbs can recover lanes they belong to respectively and compose the final Y-shape or fork-shape lanes as is illustrated in Fig. 3(b). Let F denotes the output feature map from the backbone whose resolution drops by a factor of 4 compared to the original image. We design a transfer output head and pick F as input. F goes through convolution-based transfer head to get the transfer map T which consists of forward and backward components $T_f, T_b \in \mathbb{R}^{H \times W \times 2}$. Each location in T_f is a 2D vector, which represents the offsets between the forward neighboring point p^f and the current pixel p. The definition of T_b is similar as T_f. Consequently, we can detect the forward and backward neighboring points p^f, p^b of p guided by T.

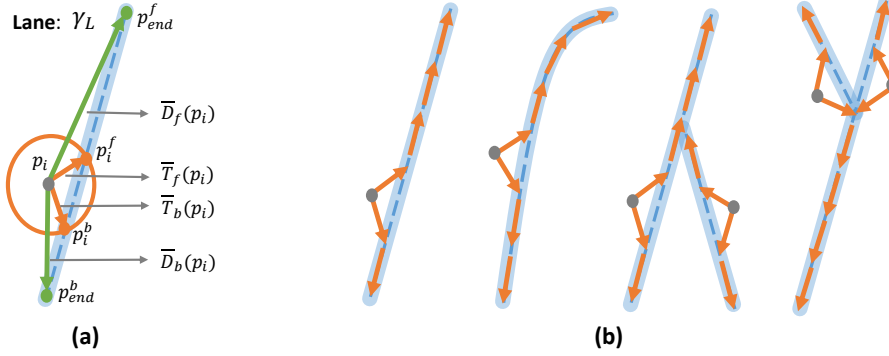$$p^f = p + T_f(p), \quad p^b = p + T_b(p). \tag{4}$$

Fig. 3. (a) is an illustration of the transfer vectors and distance scalars for p_i. $\overline{T}_{f,b}(p_i)$ are the forward and backward transfer vectors. $\overline{D}_{f,b}(p_i)$ are the forward and backward distance scalars. (b) shows our bilateral predictions can not only decode Y-shape or fork-shape lanes, but also fit simple structures, like straight lanes and curved lanes.

With the guidance of local location information in transfer map T, the whole lane can be detected iteratively via bilateral strategy.

Global shape message learning. Previous works predict positions of end points for lanes to guide decoding process. FastDraw [22] predicts end tokens to encode the global geometry while CondLaneNet [15] recovers the row-wise shape through the vertical range prediction. These methods actually ignores the relation between the end points and other points on the same lane. We make every relay station learns the global shape message transmitted in the chain by utilizing the relation mentioned above. In detail, we design a distance head to predict the distance map D that consists of the forward and backward components $D_f, D_b \in \mathbb{R}^{H \times W \times 1}$. Each location in D_f is a scalar, which represents the distance from the current pixel p to the forward end point p_{end}^f on the lane. With this global shape information, we can know when to stop the lane decoding process. Specifically speaking, the iterations for decoding the forward branch of p is $\frac{D_f}{d}$. The definition of D_b is similar as D_f as well. With the combination of local location and global geometry information, our relay chain prediction strategy performs considerably well even in complex scenarios. Next, we will introduce the novel pair of lane encoding and decoding algorithms designed for lane detection.

3.2   Lane encoder for relay station construction

The lane encoder is to create the supervision of transfer and distance maps for training. Given an image $I \in \mathbb{R}^{H \times W \times 3}$ and its segmentation mask $\overline{S} \in \mathbb{R}^{H \times W \times 1}$, for any foreground point $p_i = (x_i, y_i) \in \overline{S}$ we denote its corresponding lane as γ_L. The two forward and backward end points of γ_L are denoted as $p_{end}^f = (x_{end}^f, y_{end}^f)$ and $p_{end}^b = (x_{end}^b, y_{end}^b)$, which have the minimum and maximum
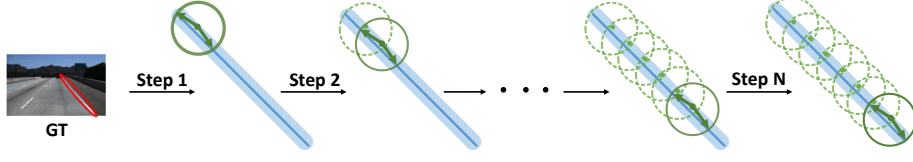
Fig. 4. Lane encoder. All foreground points are matched with the nearest lanes. The arrows in a circle indicate transfer vectors of a foreground point to its two neighbors on lane. The distance scalars represent distances between the current point and two end points of the lane. All results are generated with point-wise traversal.

y-coordinates respectively. The forward distance scalar $\overline{D}_f(p_i)$ and backward distance scalar $\overline{D}_b(p_i)$ of p_i are formulated as the following:

$$\overline{D}_f(p_i) = \sqrt{(x_i - x_{end}^f)^2 + (y_i - y_{end}^f)^2}, \tag{5}$$

$$\overline{D}_b(p_i) = \sqrt{(x_i - x_{end}^b)^2 + (y_i - y_{end}^b)^2}. \tag{6}$$

To generate the forward transfer vector and backward transfer vector for pixel p_i, we first find the two neighbors on γ_L of it with the fixed distance d. They are denoted as $p_i^f = (x_i^f, y_i^f)$ and $p_i^b = (x_i^b, y_i^b)$ and represent the forward neighbor and backward neighbor respectively. Then the forward transfer vector $\overline{T}_f(p_i)$ and the backward transfer vector $\overline{T}_b(p_i)$ for pixel p_i are defined :

$$\overline{T}_f(p_i) = (x_i^f - x_i, y_i^f - y_i), \tag{7}$$

$$\overline{T}_b(p_i) = (x_i^b - x_i, y_i^b - y_i), \tag{8}$$

$$||\overline{T}_f(p_i)||_2 = ||\overline{T}_b(p_i)||_2 = d. \tag{9}$$

The details are shown in Fig. 3(a). In addition, for two separate parts of one Y-shape lane: $l_1 = \{(x_1, y_1), \cdots, (x_m, y_m), (x_{m+1}^1, y_{m+1}^1), \cdots, (x_{n_1}^1, y_{n_1}^1)\}$, $l_2 = \{(x_1, y_1), \cdots, (x_m, y_m), (x_{m+1}^2, y_{m+1}^2), \cdots, (x_{n_2}^2, y_{n_2}^2)\}$. $\{(x_1, y_1), \cdots, (x_m, y_m)\}$ is the shared stem. We randomly choose one point from (x_{m+1}^1, y_{m+1}^1) and (x_{m+1}^2, y_{m+1}^2) as the forward neighboring point of (x_m, y_m) while (x_m, y_m) is the common backward neighboring point of (x_{m+1}^1, y_{m+1}^1) and (x_{m+1}^2, y_{m+1}^2). All foreground pixels on the $\overline{S}$ are processed following the same formula and then $\overline{T}_{f,b}$ and $\overline{D}_{f,b}$ can be generated. The process is shown in Fig. 4.

3.3   Lane decoder with transfer and distance map

With the predictions of local location and global geometry, we propose a novel lane decoding algorithm to detect all curves in a given image.
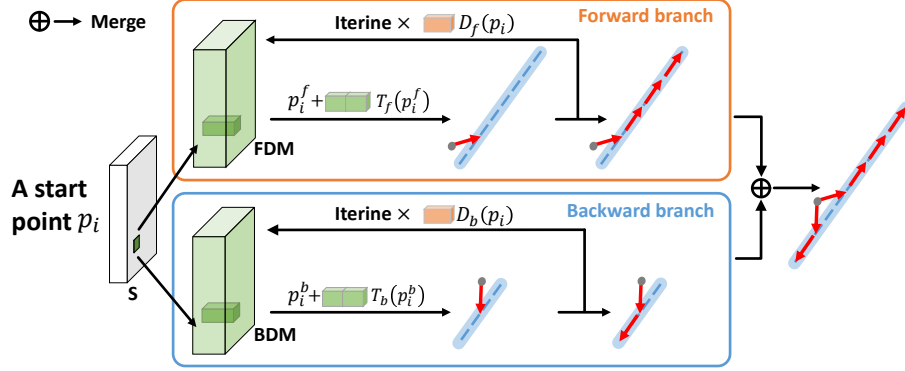
Fig. 5. The illustration of the lane decoder. The forward branch predicts the forward part of the lane via forward transfer map T_f and forward distance map D_f. The backward part can be decoded from the backward branch similarly.

Given the predicted binary segmentation mask S, tranfer map T and distance map D, we collect all the foreground points of S and use a Point-NMS to get a sparse set of key points K. Every key point $p \in K$ serves as a start point to recover one global curve.

Step1: Find the forward transfer vector $T_f(p)$ and forward distance scalar $D_f(p)$ for p. The moving steps we should extend the neighbors for the forward branch can be defined as $M^f = \frac{D_f(p)}{d}$. In other words, we can infer the location of the forward end point of p with $D_f(p)$ on the same lane.

Here d is the step length. Then the forward neighbor pixel p^f_{i+1} of p^f_i can be calculated iteratively by:

$$p^f_{i+1} = p^f_i + T_f(p^f_i), \ i \in \{0, 1, 2, \cdots, M^f - 1\}, \ p_0 = p. \tag{10}$$

The forward branch of the curve can be recovered by connecting $\{p, p^f_1, \cdots, p^f_{M^f}\}$ sequentially. The detail is shown on the top of Fig. 5.

Step2: We calculate the point set $\{p, p^b_1, p^b_2, \cdots, p^b_{M^b}\}$ following Eq. 10 via T_b and D_b and connect them sequentially to recover the backward branch.

Step3: We then merge the backward and forward curve branches together to get the global curve:

$$\gamma_L = \{p^b_{M^b}, \cdots, p^b_2, p^b_1, p, p^f_1, p^f_2, \cdots, p^f_{M^f}\}. \tag{11}$$

Finally, the non-maximum suppression [19] is performed on all the predicted curves to get the final results.

3.4   Network architecture

The overall framework is shown in Fig. 2. SegFormer [33] is utilized as our network backbone, aiming to extract global contextual information and learn the long and thin structures of lanes. SegFormer-B0, B1 and B2 are used as small, medium and large backbones in our experiments respectively. Given an image $I \in R^{H \times W \times 3}$, the segmentation head predicts the binary segmentation mask $S \in R^{H \times W \times 1}$, the transfer head predicts the transfer map T which consists of the forward and backward parts $T_f, T_b \in \mathbb{R}^{H \times W \times 2}$, and the distance head predicts the distance map D that consists of $D_f, D_b \in \mathbb{R}^{H \times W \times 1}$.

3.5   Loss function

To train our proposed model, we adopt different losses for predictions. For the binary segmentation mask, we adopt the OHEM loss [26] to train it in order to solve class imbalance problem due to the sparsity of lane segmentation points. The OHEM loss is formulated as follows:

$$L_{seg} = \frac{1}{N_{pos} + N_{neg}} (\sum_{i \in S_{pos}} y_i log(p_i) + \sum_{i \in S_{neg}} (1 - y_i) log(1 - p_i)). \qquad (12)$$

where S_{pos} is the set of positive points and S_{neg} is the set of hard negative points which is most likely to be misclassified as positive. N_{pos} and N_{neg} denote the number of points in S_{pos} and S_{neg} respectively. The ratio of N_{neg} to N_{pos} is a hyperparmeter μ. As for the per-pixel transfer and distance maps, we simply adopt the smooth L_1 loss, which are denoted as L_T and L_D, to train them.

$$L_D = \frac{1}{N_{pos}} \sum_{i \in S_{pos}} L_{smooth_{L_1}}(D(p_i), \overline{D}(p_i)), \qquad (13)$$

$$L_T = \frac{1}{N_{pos}} \sum_{i \in S_{pos}} L_{smooth_{L_1}}(T(p_i), \overline{T}(p_i)). \qquad (14)$$

In the training phase, the total loss is defined as follows:

$$L_{total} = L_{seg} + L_T + L_D. \qquad (15)$$

4   Experiment

4.1   Experimental setting

Dataset. We conduct experiments on four widely used lane detection benchmark datasets: CULane [21], TuSimple [30], LLAMAS [2] and CurveLanes [34]. CULane consists of 55 hours of videos which comprises nine different scenarios, including normal, crowd, dazzle night, shadow, no line, arrow, curve, cross and

Table 1. Lane detection datasets.

Dataset	Train	Val.	Test	Road type
CULane [21]	88.9k	9.7k	34.7k	urban, highway
CurveLanes [34]	100K	20K	30K	urban, highway
TuSimple [30]	3.3k	0.4k	2.8k	highway
LLAMAS [2]	58.3k	20.8k	20.9k	highway

Table 2. State-of-the-art comparison on CULane. Even the small version of our RCLane achieves the state-of-art performance with only 6.3M parameters.

Method	Total	Normal	Crowded	Dazzle	Shadow	No line	Arrow	Curve	Cross	Night	Params(M)	FPS
SCNN [21]	71.60	90.60	69.70	58.50	66.90	43.40	84.10	64.40	1990	66.10	-	7.5
CurveLanes-NAS-S [34]	71.40	88.30	68.60	63.20	68.00	47.90	82.50	66.00	2817	66.20	-	-
CurveLanes-NAS-M [34]	73.50	90.20	70.50	65.90	69.30	48.80	85.70	67.50	2359	68.20	-	-
CurveLanes-NAS-L [34]	74.80	90.70	72.30	67.70	70.10	49.40	85.80	68.40	1746	68.90	-	-
LaneATT-S [27]	75.13	91.17	72.71	65.82	68.03	49.13	87.82	63.75	1020	68.58	13.3	250
LaneATT-M [27]	76.68	92.14	75.03	66.47	78.15	49.39	88.38	67.72	1330	70.72	23.4	171
LaneATT-L [27]	77.02	91.74	76.16	69.47	76.31	50.46	86.29	64.05	1264	70.81	18.8	26
LaneAF (DLA-34) [1]	77.41	91.80	75.61	71.78	79.12	51.38	86.88	72.70	1360	73.03	20.2	-
FOLO [24]	78.80	92.70	77.80	75.20	79.30	52.10	89.00	69.40	1569	74.50	-	-
CondLaneNet-S [15]	78.14	92.87	75.79	70.72	80.01	52.39	89.37	72.40	1364	73.23	12.1	220
CondLaneNet-M [15]	78.74	93.38	77.14	71.17	79.93	51.85	89.89	73.88	1387	73.92	22.2	152
CondLaneNet-L [15]	79.48	93.47	77.44	70.93	80.91	**54.13**	90.16	75.21	1201	74.80	49.9	58
RCLane-S (Ours)	79.52	93.41	77.93	**73.32**	80.31	53.84	89.04	75.66	1298	74.33	6.3	45.6
RCLane-M (Ours)	80.03	93.59	78.77	72.44	**84.37**	52.77	90.31	78.39	**907**	73.96	17.2	43.8
RCLane-L (Ours)	**80.50**	**94.01**	**79.13**	72.92	81.16	53.94	**90.51**	**79.66**	931	**75.10**	30.9	24.5

night. The TuSimple dataset is collected with stable lighting conditions on highways. LLAMAS is a large lane detection dataset obtained on highway scenes with annotations auto-generated by using high-definition maps. CurveLanes is a recently proposed benchmark with cases of complex topologies such as Y-shape lanes and dense lanes. The details of four datasets are shown in Tab. 1.

Evaluation metrics. For CULane, CurveLanes and LLAMAS, we utilize F1-measure as the evaluation metric. While for TuSimple, accuracy is presented as the official indicator. And we also report the F1-measure for TuSimple. The calculation method follows the same formula as in CondLaneNet [15].

Implementation details. The small, medium and large versions of our RCLane-Det are used on all four datasets. Except when explicitly indicated, the input resolution is set to 320×800 during training and testing. For all training sessions, we use AdamW optimizer [17] to train 20 epochs on CULane, CurveLanes and LLAMAS, 70 epochs on TuSimple respectively with a batch size of 32. The learning rate is initialized as $6e$-4 with a "poly" LR schedule. We set η for calculating IOU between lines as 15, the ratio of N_{neg} to N_{pos} μ as 15, the minimum distance between any two foreground pixels of in Point-NMS τ as 2. We implement our method using the Mindspore [18] on Ascend 910.
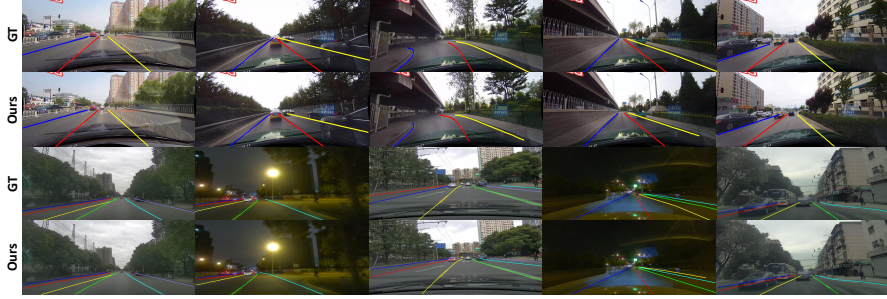
Fig. 6. Visualization results of our RCLane. The first two rows are of CULane and the last two rows are of CurveLanes. Our model can detect curve lanes, dense lanes and Y-shape lanes in different scenarios.

4.2 Results

CULane. As illustrated in Tab. 2, RCLane achieves a new state-of-the-art result on the CULane testing set with an 80.50% F1-measure. Compared with the best model as far as we know, CondLaneNet [15], although our method performs better only 1.02% of F1-measure compared with the best model before CondLaneNet since CULane is a simpler dataset with may straight lines, it has an considerable improvements in crowded and curve scenes, which demonstrates that *Relay Chain* can strengthen local location connectivity through global shape learning for local occlusions and complex lane topologies. The visualization result is shown in the first two rows of Fig. 6.

Table 3. Performance of different methods on CurveLanes.

Method	F1(%)	Precision(%)	Recall(%)
SCNN [21]	65.02	76.13	56.74
Enet-SAD [7]	50.31	63.60	41.60
PointLaneNet [4]	78.47	86.33	72.91
CurveLane-S [34]	81.12	93.58	71.59
CurveLane-M [34]	81.80	93.49	72.71
CurveLane-L [34]	82.29	91.11	75.03
CondLaneNet-S [15]	85.09	87.75	82.58
CondLaneNet-M [15]	85.92	88.29	83.68
CondLaneNet-L [15]	86.10	88.98	83.41
RCLane-S (Ours)	90.47	93.33	87.78
RCLane-M (Ours)	90.96	93.47	88.58
RCLane-L (Ours)	**91.43**	**93.96**	**89.03**

CurveLanes. CurveLanes [34] is a challenging benchmark with many hard scenarios. The evaluation results are shown in Tab. 3. We can see that our largest

Fig. 7. Comparison of visualization results between our RCLane and other lane detection methods. Our method performs better in different scenarios such as occluded, curved and Y-shape lanes.

model (with SegFormer-B2) surpasses CondLaneNet-L by 5.33% in F1-measure, which is more pronounced than it on CULane. Due to that CurveLanes is more complex with Fork-shape, Y-shape and other curve lanes, improvements both in recall rate and accuracy prove that RCLane has generalization ability on lanes. The visualization results is shown in the last two rows of Fig. 6. And the qualitative comparison with other methods is shown in Fig. 7.

TuSimple. The results on TuSimple are shown in Tab. 4. As Tusimple is a small dataset and scenes are more simple with accurate annotations, the gap between all methods is small. Moreover, our method also achieves a new state-of-the-art F1 score of 97.64%.

Table 4. Performance of different methods on TuSimple.

Method	F1(%)	Acc(%)	FP(%)	FN(%)
SCNN [21]	95.97	96.53	6.17	**1.80**
PointLaneNet [4]	95.07	96.34	4.67	5.18
LaneATT-ResNet18 [27]	96.71	95.57	3.56	3.01
LaneATT-ResNet34 [27]	96.77	95.63	3.53	2.92
LaneATT-ResNet122 [27]	96.06	96.10	5.64	2.17
CondLaneNet-S [15]	97.01	95.48	2.18	3.80
CondLaneNet-M [15]	96.98	95.37	2.20	3.82
CondLaneNet-L [15]	97.24	96.54	**2.01**	3.50
LaneAF(DLA-34) [1]	96.49	95.62	2.80	4.18
FOLO [24]	-	**96.92**	4.47	2.28
RCLane-S (Ours)	97.52	96.49	2.21	2.57
RCLane-M (Ours)	97.61	96.51	2.24	2.36
RCLane-L (Ours)	**97.64**	96.58	2.28	2.27

LLAMAS LLAMAS [2] is a new dataset with more than $100K$ images from highway scenarios. The results of our RCLane on LLAMAS is shown in Tab. 5. The best result of our method is 96.13% F1 score with RCLane-L.

Table 5. Performance of different methods on LLAMAS.

Method	F1(%)	Precision(%)	Recall(%)
PolyLaneNet [28]	88.40	88.87	87.93
LaneATT-ResNet-18 [27]	93.46	**96.92**	90.24
LaneATT-ResNet-34 [27]	93.74	96.79	90.88
LaneATT-ResNet-122 [27]	93.54	96.82	90.47
LaneAF(DLA-34) [1]	96.07	96.91	95.26
RCLane-S (Ours)	96.05	96.70	95.42
RCLane-M (Ours)	96.03	96.62	95.45
RCLane-L (Ours)	**96.13**	96.79	**95.48**

4.3   Ablation study

Different modules. In this section, we perform the ablation study to evaluate the impact of the proposed relay station construction, bilateral predictions and global shape message learning on CurveLanes. The results is shown in Tab. 6. The first row shows the baseline result, which only uses binary segmentation plus post processing named DBSCAN [5] to detect lanes. In the second row, the lane is recovered from bottom to top gradually with the guidance of the forward transfer map and forward distance map. While the third row detect lanes from top to bottom. In the fourth row, we only use the forward and backward transfer maps to predict the lane. And we present our full version of RCLane in the last row, which attains a new state-of-art result 91.43% on CurveLanes.

Comparing the first two rows, we can see that the proposed relay station construction has greatly improved the performance. Then, we add global shape information learning with distance map which can improve the performance from 88.19% to 91.43%. While we do additional two experiments in the second and third lines, the lane is detected by transfer and distance maps from one-way direction and there is a certain gap with the highest F1-score. It proves that our bilateral prediction has generalization in depicting topologies of lanes. In addition, there exists a gap between the forward the backward models. As the near lanes (the bottom region of the image) are usually occluded by the ego car, the corresponding lane points get low confidence scores from the segmentation results. Therefore the starting points are usually outside of the occluded area and the forward counterpart eventually has no chance back to cover the lanes at the bottom of the image. In contrast, the backward model detects lanes more completely with the help of the distance map when decoding from the top, including the occluded area.

Table 6. Comparison of different components on CurveLanes. The T_f, T_b, D_f, D_b represent the forward transfer map, backward transfer map, forward distance map and backward distance map respectively.

Baseline	T_f	T_b	D_f	D_b	F1(%)
$\checkmark$					51.22
	$\checkmark$		$\checkmark$		$75.06^{+23.84}$
		$\checkmark$		$\checkmark$	$83.78^{+32.56}$
	$\checkmark$	$\checkmark$			$88.19^{+36.97}$
	$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	$91.43^{+40.21}$

Comparisons with other methods using the same backbone. We additionally use Segformer-B2 [33] as backbone to train CondLaneNet [15] and LaneAF [1] respectively and show their results on Tab. 7 below. Without changing the parameters of their models, our model still outperforms LaneAF and CondLaneNet by a margin on CULane [21] dataset due to its superior precision, which demonstrates the high quality of lanes detected by RCLane. It further fairly verifies the superiority of our proposed relay chain prediction method, which can process local location and global geometry information simultaneously to improve the capacity of the model.

Table 7. Comparisons with other methods using the same backbone Segformer-B2.

method	Precision(%)	Recall(%)	F1(%)
LaneAF [1]	80.89	71.71	76.02
CondLaneNet [15]	82.58	**76.01**	79.16
RCLane (Ours)	**88.52**	73.82	**80.50**

Experimental setting for step length d. Step length d is the distance between the two neighbors when encoding the lane, which is fixed as 10 for all the experiments in our method. LaneATT [27] sets the line width as 30 for calculating IoU. We set it as 15 according to the resolution scale initially. And d should be at least half of the line width to ensure all foreground points find neighbors sited at the center line. Thus we set d as 7, 8, 9, 10, 11 and 12 and conduct a series of experiments on CurveLanes [34]. The quantitative results are shown in Tab. 8. We find that F1-score decreases as d increases since small step length can describe the local variable shape of lanes more precisely while increasing decoding time. 10 is chosen as the final setting considering the performance-speed trade-off.

Local location and global shape message modeling. In Fig. 8 $A.(1,3)$, the transfer map can capture local location information depicting topology of the lane precisely, while the distance map in Fig. 8 $A.(2,4)$ models global shape information with large receptive field. Furthermore, in some driving scenarios, there occurs loss of lane information due to the disappearance of trace for lanes

Table 8. Ablation study on step length on CurveLanes.

Step length	Precision(%)	Recall(%)	F1(%)
7	94.29	88.87	91.50
8	94.26	88.84	91.47
9	94.28	88.79	91.45
10	93.96	89.03	91.43
11	93.93	88.97	91.38
12	94.16	88.93	91.31

as is shown in Fig. 8(B). However, lanes are still captured faintly in the transfer map with the global shape information learning. The results show the robustness of our RCLane with local location and global shape message modeling.

Fig. 8. Visualization of network outputs. A.$(1, 3)$ are features of D_f and D_b, while A.$(2, 4)$ are features of T_f and T_b. A.(5) is the segmentation result and becomes sparse map A.(6) via Point-NMS. B is a harder frame compared to A.

5   Conclusion

In this paper, we have proposed to solve lane detection problem by learning a novel relay chain prediction model. Compared with existing lane detection methods, our model is able to capture global geometry and local information

progressively with the novel relay station construction and global shape message learning. Furthermore, bilateral predictions can adapt to hard topologies, such as Fork-shape and Y-shape. Extensive experiments on four benchmarks including CULane, CurveLanes, Tusimple and LLAMAS demonstrate state-of-the-art performance and generalization ability of our *RCLane*.

Acknowledgments

References

1. Abualsaud, H., Liu, S., Lu, D., Situ, K., Rangesh, A., Trivedi, M.M.: Laneaf: Robust multi-lane detection with affinity fields. arXiv preprint (2021)
2. Behrendt, K., Soussan, R.: Unsupervised labeled lane markers using maps. In: ICCV workshops (2019)
3. Borkar, A., Hayes, M., Smith, M.T.: Robust lane detection and tracking with ransac and kalman filter. In: ICIP (2009)
4. Chen, Z., Liu, Q., Lian, C.: Pointlanenet: Efficient end-to-end cnns for accurate real-time lane detection. In: IEEE intelligent vehicles symposium (2019)
5. 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)
6. Hou, Y., Ma, Z., Liu, C., Hui, T.W., Loy, C.C.: Inter-region affinity distillation for road marking segmentation. In: CVPR (2020)
7. Hou, Y., Ma, Z., Liu, C., Loy, C.C.: Learning lightweight lane detection cnns by self attention distillation. In: ICCV (2019)
8. Hur, J., Kang, S.N., Seo, S.W.: Multi-lane detection in urban driving environments using conditional random fields. In: IV (2013)
9. Jiang, R., Klette, R., Vaudrey, T., Wang, S.: New lane model and distance transform for lane detection and tracking. In: International Conference on Computer Analysis of Images and Patterns (2009)
10. Jiang, Y., Gao, F., Xu, G.: Computer vision-based multiple-lane detection on straight road and in a curve. In: 2010 International Conference on Image Analysis and Signal Processing (2010)
11. Kim, Z.: Robust lane detection and tracking in challenging scenarios. IEEE Transactions on intelligent transportation systems (2008)
12. Ko, Y., Lee, Y., Azam, S., Munir, F., Jeon, M., Pedrycz, W.: Key points estimation and point instance segmentation approach for lane detection. IEEE Transactions on Intelligent Transportation Systems (2021)
13. Lee, S., Kim, J., Shin Yoon, J., Shin, S., Bailo, O., Kim, N., Lee, T.H., Seok Hong, H., Han, S.H., So Kweon, I.: Vpgnet: Vanishing point guided network for lane and road marking detection and recognition. In: ICCV (2017)

14. Liu, G., Wörgötter, F., Markelić, I.: Combining statistical hough transform and particle filter for robust lane detection and tracking. In: IEEE Intelligent Vehicles Symposium (2010)
15. Liu, L., Chen, X., Zhu, S., Tan, P.: Condlanenet: a top-to-down lane detection framework based on conditional convolution. In: ICCV (2021)
16. Liu, R., Yuan, Z., Liu, T., Xiong, Z.: End-to-end lane shape prediction with transformers. In: WACV (2021)
17. Loshchilov, I., Hutter, F.: Fixing weight decay regularization in adam (2018)
18. Mindspore: `https://www.mindspore.cn/` (2020)
19. Neubeck, A., Van Gool, L.: Efficient non-maximum suppression. In: ICPR (2006)
20. Neven, D., De Brabandere, B., Georgoulis, S., Proesmans, M., Van Gool, L.: Towards end-to-end lane detection: an instance segmentation approach. In: IEEE intelligent vehicles symposium (2018)
21. Pan, X., Shi, J., Luo, P., Wang, X., Tang, X.: Spatial as deep: Spatial cnn for traffic scene understanding. In: AAAI (2018)
22. Philion, J.: Fastdraw: Addressing the long tail of lane detection by adapting a sequential prediction network. In: CVPR (2019)
23. Qin, Z., Wang, H., Li, X.: Ultra fast structure-aware deep lane detection. In: ECCV (2020)
24. Qu, Z., Jin, H., Zhou, Y., Yang, Z., Zhang, W.: Focus on local: Detecting lane marker from bottom up via key point. In: CVPR (2021)
25. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: MICCAI (2015)
26. Shrivastava, A., Gupta, A., Girshick, R.: Training region-based object detectors with online hard example mining. In: CVPR (2016)
27. 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: CVPR (2021)
28. Tabelini, L., Berriel, R., Paixao, T.M., Badue, C., De Souza, A.F., Oliveira-Santos, T.: Polylanenet: Lane estimation via deep polynomial regression. In: ICPR (2021)
29. Tan, H., Zhou, Y., Zhu, Y., Yao, D., Li, K.: A novel curve lane detection based on improved river flow and ransa. In: IEEE conference on intelligent transportation systems (itsc) (2014)
30. TuSimple: Tusimple benchmark. `https://github.com/TuSimple/tusimple-benchmark` (2019)
31. Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NeurIPS (2017)
32. Wang, H., Zhu, Y., Green, B., Adam, H., Yuille, A., Chen, L.C.: Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In: ECCV (2020)
33. Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M., Luo, P.: Segformer: Simple and efficient design for semantic segmentation with transformers. arXiv preprint (2021)
34. Xu, H., Wang, S., Cai, X., Zhang, W., Liang, X., Li, Z.: Curvelane-nas: Unifying lane-sensitive architecture search and adaptive point blending. In: ECCV (2020)
35. Yoo, S., Lee, H.S., Myeong, H., Yun, S., Park, H., Cho, J., Kim, D.H.: End-to-end lane marker detection via row-wise classification. In: CVPR workshops (2020)
36. Zhou, S., Jiang, Y., Xi, J., Gong, J., Xiong, G., Chen, H.: A novel lane detection based on geometrical model and gabor filter. In: IEEE Intelligent Vehicles Symposium (2010)

A   Appendix

A.1   Attention mechanisms

We consider the recent attention mechanism is suitable for information processing and do ablation experiments on *CurveLanes*. Results are shown in Tab. 9. Adding self-attention [31] to U-Net [25], which operated on the deepest feature map, makes F1-score increase from 89.41% and reaches 89.49%. In the third row, we replace it with axial attention [32] and further improves the performance considering the row-column style attention adapts to the long and thin characteristics of lanes. Finally, we utilize the efficient transformer-based network SegFormer as backbone and achieve the best result. From the above results, we can find attention mechanism can help our model focus on local location and global shape information simultaneously.

Table 9. Comparison among different settings of attention on the CurveLanes. The attention module is just added to the feature map of the smallest resolution.

Backbone	Precision(%)	Recall(%)	F1(%)
Unet [25]	93.11	86.00	89.41
Unet [25] + self-attention [31]	92.95	86.27	$89.49^{+0.08}$
Unet [25] + axial-attention [32]	92.79	88.41	$90.55^{+1.14}$
SegFormer-B2 [33]	93.96	89.03	$91.43^{+2.02}$

A.2   Generalization

This section aims to verify the generalization capacity of our RCLane following FOLOLane [24]. We utilize the model trained on the CULane [21] training set to evaluate on the TuSimple [30] test set. The results are shown in Tab. 10. Our RCLane surpasses FOLOLane [24] by 2.88% , with smaller FP and FN indicating that our method is more robust than previous lane detection methods.

Table 10. Evaluation of generalization ability of different methods from CULane training set to TuSimple testing set.

Method	Accuracy(%)	FP(%)	FN(%)
PINet [12]	36.31	48.86	89.88
UFNet [23]	65.53	56.80	65.46
FOLOLane [24]	84.36	39.64	38.41
RCLane (Ours)	**87.24**	**22.06**	**21.56**

A.3   Visualization results on Tusimple and LLAMAS

The qualitative results on TuSimple [30] and LLAMAS [2] are shown in Fig. 9. TuSimple and LLAMAS are two benchmarks taken from the highway driving scenarios and are easier compared with CULane [21] and CurveLanes [34]. In some scenarios such as curve lanes or the far end of lanes, our RCLane even shows better performance than ground truths, as is shown in the last row and forth column in Fig. 9.
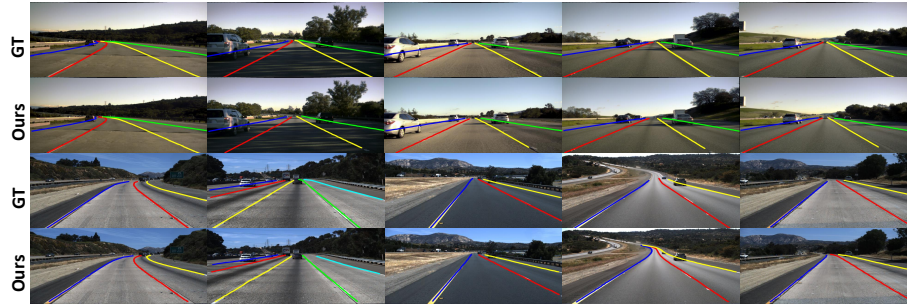

Fig. 9. Visualization on Tusimple and LLAMAS. The first two rows are the ground truth and our predictions on LLAMAS and the last two rows are the ground truth and our predictions on Tusimple.

A.4   Visualizations on complex scenes

Moreover, we visualize more results of our RCLane on other hard cases on Curve-Lanes in Fig 10 and achieve good performance both of Fork-shape lanes and nearly horizontal lanes.

Fig. 10. Visualization of detection results for complex topologies, such as fork-shape, Y-shape and nearly horizontal lanes, shows the capacity of RCLane.

A.5   Pseudo code

Pseudo code for lane encoder. Algorithm·1 details the process of lane encoder, which aims to generate the ground truth of the transfer map and distance map to supervise the training process.

Algorithm 1 Lane encoder

Input: d: the step length

$\overline{S}$: the foreground point set in segmentation map

L: the set of lanes in the image I

$g_1(\cdot)$: the function to compute the distance between a point and a lane

$g_2(\cdot)$: the function to compute the distance between two points

Output: $\overline{T}_f$: the forward transfer map

$\overline{T}_b$: the backward transfer map

$\overline{D}_f$: the forward distance map

$\overline{D}_b$: the backward distance map

1: **for all** $p_i = (x_i, y_i) \in \overline{S}$ **do**

2:     $l = \arg\min_{\gamma \in L} g_1(p_i, \gamma)$

3:     **if** $g_1(p_i, l) < d$ **then**

4:         find the forward and backward end points of l:

$p_{end}^f = (x_{end}^f, y_{end}^f) = \arg\max_{(x,y)\in l}(y)$,

$p_{end}^b = (x_{end}^b, y_{end}^b) = \arg\min_{(x,y)\in l}(y)$

5:         compute the forward and backward distance scalars:

$\overline{D}_f(p_i) = \sqrt{(x_i - x_{end}^f)^2 + (y_i - y_{end}^f)^2}$,

$\overline{D}_b(p_i) = \sqrt{(x_i - x_{end}^b)^2 + (y_i - y_{end}^b)^2}$

6:         find the forward and backward point for p_i on lane l:

$p_i^f = (x^f, y^f) = \arg\max_{g_2((x,y),p_i)=d}(y)$,

$p_i^b = (x^f, y^f) = \arg\min_{g_2((x,y),p_i)=d}(y)$

7:         compute the two transfer vectors:

$\overline{T}_f(p_i) = (x_i^f - x_i, y_i^f - y_i)$,

$\overline{T}_b(p_i) = (x_i^b - x_i, y_i^b - y_i)$

8:     **end if**

9: **end for**

10: **return** $\overline{T}_f, \overline{T}_b, \overline{D}_f$ and $\overline{D}_b$

Pseudo code for lane decoder. Algorithm 2 introduces the detail process for lane decoder, showing how to recover all possible lanes based on the predicted segmentation map, transfer map and distance map.

Algorithm 2 Lane decoder

Input: d : the step length

 r: the minimal distance between two foreground points in key point map

 $f(\cdot, r)$: the function to perform Point-NMS on segmentation map to get a sparse key point map

 $h(\cdot)$: the function to perform IOU-NMS on the predict set of lanes

 S: the predicted segmentation map

 T: the predicted transfer map

 D: the predicted distance map

Output: L: the set of lanes in the image I

1: key point map $K = f(S, r)$

2: initialize L' as an empty set

3: **for all** key point $p \in K$ **do**

4: initialize two point sets: $G_1 = \{p\}$ and $G_2 = \{p\}$

5: initialize two points: $p_1 = p$ and $p_2 = p$

6: **for** $i = 1$ to $\frac{D_f(p)}{d}$ **do**

7: get the forward transfer vector $T_f(p_1)$ from T_f

8: update p_1: $p_1 = p_1 + T_f(p_1)$

9: update G_1: $G_1 = G_1 \cup \{p_1\}$

10: **end for**

11: **for** $i = 1$ to $\frac{D_b(p)}{d}$ **do**

12: get the backward transfer vector $T_b(p_2)$ from T_b

13: update p_2: $p_2 = p_2 + T_b(p_2)$

14: update G_2: $G_2 = G_2 \cup \{p_2\}$

15: **end for**

16: get the lane $l_p = \{G_1\} \cup \{G_2\}$

17: update L': $L' = L' \cup \{l_p\}$

18: **end for**

19: the final results of predicted lanes $L = h(L')$

20: **return** the set of lanes L
