Appendix Overview

This appendix is a supplement to the main paper. In Sec. A, more details of the network structure of our proposed ElasticLaneNet are illustrated. In Sec. B, more details in the implementation of our ElasticLaneNet are provided. Some specific comparisons on the three datasets that we use are provided. Section C gives further information of the explicit implementation of ElasticLaneNet^{pw}. Section D provides more results details and discussion, including the labels of Fig. 7, more discussion of CULane's results, and the further improvement direction are illustrated here. Section E shows some simple diagrams of the different types of models mentioned in Sec. 2. In Sec. F, the settings of other models in the comparison experiments on SDLane are presented. The open source of this work is on `https://github.com/yxfengl/ElasticLaneNet`.

A. More Details of the Network Structure

This section provides more details of the network structure of our ElasticLaneNet shown in Fig. 2 in Sec. 3 in the main text. In Fig. 2, the backbone consists of an Encoder ResNet34 and a Feature Pyramid Network (FPN), in which the down-sampling processes are indicated by the downward red arrows, the up-sampling processes by the green arrows, and the features concatenation processes by the gray arrows; the main stream is from the backbone to the *ELMM*, while the auxiliary steps include the Transformer Bottleneck layer (TB), classification sub-network (CSN), the range sub-network (RSN) and the auxiliary feature refinement (AFR). These auxiliary processes are framed (or connected) by dotted lines (or arrows), which means they can be removed.

After applying CSN and RSN (CRSN for short), in addition to using the loss functions $\mathcal{L}_{exist}$ (lane existence loss) and $\mathcal{L}_{range}$ (the binary-cross entropy loss), the outputs are taken element-wise multiplication with the outputs from *ELMM*, i.e. Pred$_1$, to jointly train and learn the implicit *ELM*s $\Psi_{p_1}^k$'s, $k = 1, 2, ..., N$, with the size of $N \times M \times w$. In AFR, the auxiliary loss $\mathcal{L}_{aux}$ is applied on Pred$_i$, namely $\Psi_{p_i}^k$, $i = 2, 3$. The feature fusion (FF) in AFR module means adopting convolution on the concatenation of F_2 and F_3 with P_1, the last layer from FPN, to obtain Pred$_1$. F_2 and F_3 here are the features up-sampled from P_2 and P_3, having the same size as P_1, i.e., 40×100.

B. More About Implementation and Datasets

B.1. Parameters Setting

In the EIE loss, the hyper-parameter α is set 0.5, and σ in $H_\sigma(\phi)$ is set according to the sample number M, *i.e.* $M = 36$ with $\sigma = 3$ for SDLane and TuSimple, $M = 18$ with $\sigma = 5$ for CULane.

During our training, the batch size is set to be 24 to 32, the optimizer is AdamW [21], the learning rate is about 3×10^{-4} for both three datasets. The data augmentation includes resize, flip, channel shuffle, random brightness or saturation change, affine transformation, etc., same as the setting in [37].

B.2. Data Preparation and Model Evaluation

In the TuSimple experiment, we perform Ground Truth (GT) sorting (based on the lanes' slope, same as [27, 37]) because the GT orders in TuSimple are random.

In the experiment of SDLane, we first reorganize the data structure into CULane format [24], and use the same official evaluation metrics as those in CULane to evaluate the performance. The evaluation code of these two datasets is the same as that in [27], which is a python re-implementation of the official C++ code.

Before entering the neural network, the input images are cropped at a fixed scale on top of the images that without lane. The TuSimple and CULane are cropped according to the prior work [37]. SDLane is cut to a size of $1920 * 660$.

B.3. More Details on Datasets Description

In this paper, three datasets are applied for evaluation: SDLane [12], CULane [24] and TuSimple [29].

SDLane [12] is a newly proposed dataset with up to 7 lanes and a variaty of high complexity of lane structures, including intersections, Y-shape forks, confluence roads, dense lanes, widely distributed curves, etc.

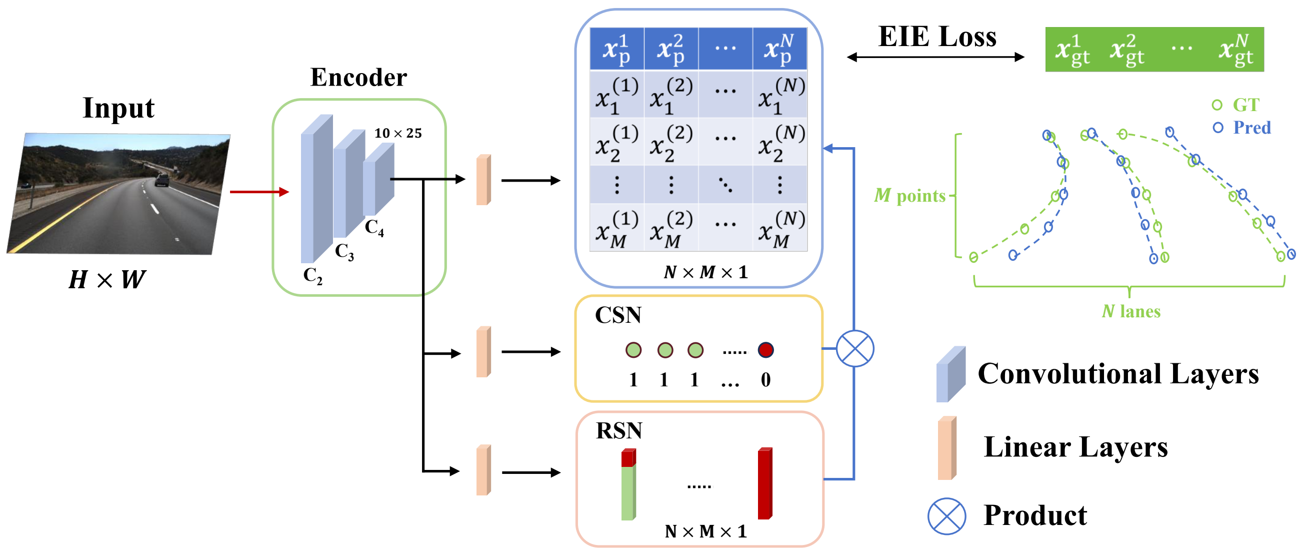
In CULane [24], the driving scenes are divided into nine categories, which include several complex scenes such as crowded, night, shadows and dazzling images. While any drivable areas are considered to have lanes, *e.g*. complex lane structures on the crossroad, while CULane regards the crossroad as no lane (category "Cross"). In CULane, the detection of crossroads detection almost becomes "Cross" type identification. In addition, there is a class of images in CULane, "No line", where the lanes, *i.e*. the driving areas need to be detected correctly, unlike the "Cross" class, which needs to be predicted to have no lane lines.

TuSimple [29] is a dataset of highways in good weather. There are some curve lanes bending at the end portions, and the curvature is much smaller than those in the SDLane.

The details of these datasets are summarized in Tab. 8.

C. More Details of ElasticLaneNet^{pw}

In this section, we describe more details of the explicit approach ElasticLaneNet^{pw}, which is an alternative EIE loss based implementation of ElasticLaneNet. Compared to the implicit implementation of ElasticLaneNet presented in the main text, from which we obtains the lane points from the zero contours of the output *ELM*s, the



Input
$H \times W$
Encoder
10×25
C_4
C_3
C_2
x_p^1 x_p^2 $\cdots$ x_p^N
$x_1^{(1)}$ $x_1^{(2)}$ $\cdots$ $x_1^{(N)}$
$x_2^{(1)}$ $x_2^{(2)}$ $\cdots$ $x_2^{(N)}$
$x_M^{(1)}$ $x_M^{(2)}$ $\cdots$ $x_M^{(N)}$
$N \times M \times 1$
EIE Loss
x_{gt}^1 x_{gt}^2 $\cdots$ x_{gt}^N
GT
Pred
M points
N lanes
CSN
1 1 1 ... 0
RSN
$N \times M \times 1$
Convolutional Layers
Linear Layers
Product

Figure a. Network architecture of ElasticLaneNet^{pw}.

Datasets	SDLane [12]	CULane [24]	TuSimple [29]
Scene	Highway/Urban	Highway/Urban	Highway
Amount	43K	133.2K	13.2K
Resolution	1208×1920	590×1640	720×1280
Curves	$\geq 90\%$	$< 2\%$	$\sim 30\%$
Lanes	$\leq$7	$\leq$4	$\leq$5
Forks	$\surd$	$\times$	$\times$
Crossroads	$\surd$	no lane	$\times$

Table 8. Details of experiment datasets.

ElasticLaneNet^{pw} predicts the x-coordinates of N lanes $\left(\boldsymbol{\gamma}_p^1, \ldots, \boldsymbol{\gamma}_p^N\right)$ directly from the network, *i.e.* $\left(\boldsymbol{x}_p^1, \ldots, \boldsymbol{x}_p^N\right)$, where $\boldsymbol{x}_p^k = \left(x_1^{(k)}, x_2^{(k)}, \ldots, x_M^{(k)}\right)^\top$; see Fig. a. The ground truth lanes are $\boldsymbol{\gamma}_{gt}^k$, and coordinates on them are $\left(\boldsymbol{x}_{gt}^k, \boldsymbol{y}_{gt}^k\right), k = 1, \ldots, N.$

As mentioned in Sec. 3.2, the elastic interaction energy (EIE) of two groups of pair-wise open curves is

$$E = \frac{1}{8\pi} \int_\gamma \int_{\gamma'} \frac{\boldsymbol{dl} \cdot \boldsymbol{dl}'}{r}, \tag{5}$$

where vector $\boldsymbol{dl}$ represents line element on curves γ with tangent direction $\boldsymbol{\tau}$, *i.e.* $\boldsymbol{dl} = \boldsymbol{\tau} dl$, dl is the arc length, γ' denotes the curves with another parameter, *i.e.* $\gamma = \gamma(s)$ and $\gamma' = \gamma(s')$, s (s') is the parameter. The coefficient $\frac{1}{8\pi}$ is set before the EIE so that the Fourier transform of EIE has no coefficient: $\mathcal{L}_{eie} = \sum_{m,n} \sqrt{m^2+n^2}\, |d_{mn}|^2$.

In order to apply an explicit method, the energy can be rewritten using the relationship of $\boldsymbol{dl} = \boldsymbol{\tau}\delta(\gamma)\mathrm{d}x\mathrm{d}y$, as

$$E(x,y) = \frac{1}{8\pi} \int_{\mathbf{R}^2} \delta(\gamma)\mathrm{d}x\mathrm{d}y \int_{\mathbf{R}^2} \frac{\boldsymbol{\tau} \cdot \boldsymbol{\tau}'}{r} \delta(\gamma')\, \mathrm{d}x'\mathrm{d}y', \tag{6}$$

where $\delta(\cdot)$ is a Delta function, $\boldsymbol{\tau}$ and $\boldsymbol{\tau}'$ are unit tangent vectors with different parameterized curve. In the implementation, a regularized delta function is required to smear out the singularities.

The velocity field of γ as well as the negative gradient descent direction of the EIE is:

$$\begin{aligned} \gamma_t(x,y) &= -\frac{\delta E}{\delta \gamma} \\ &= \left(\frac{1}{4\pi} \int_{\mathbf{R}^2} \frac{\boldsymbol{r} \cdot \boldsymbol{n}_{\gamma'}}{r^3} \delta(\gamma')\mathrm{d}x'\mathrm{d}y' \right) \boldsymbol{n}, \end{aligned} \tag{7}$$

where $\boldsymbol{n}_{\gamma'}$ is the normal vector of parameterized curve $(x(s'), y(s'))$, $\boldsymbol{n}$ is the unit normal vector of curve on (x,y) over the whole image domain.

In Eq. (7), the γ consists of the pair-wise ground truth γ_{gt} and the prediction lane γ_p. The prediction lane γ_p has opposite orientation to the ground truth γ_{gt} as mentioned in Sec. 3.2. The velocity of the prediction is:

$$\begin{aligned} &\gamma_{p_t}(x,y) \\ &= -\left(\frac{1}{4\pi} \int_{\mathbf{R}^2} \frac{\boldsymbol{r}}{r^3} \cdot (\boldsymbol{n}_{\gamma'_{gt}} \delta(\gamma'_{gt}) - \alpha \boldsymbol{n}_{\gamma'_p} \delta(\gamma'_p))\mathrm{d}x'\mathrm{d}y' \right) \boldsymbol{n}_{\gamma_p}, \end{aligned} \tag{8}$$

where α is the same hyper-parameter as mentioned in Eq. (2).

In order to simplify the calculation, the regularized delta function chosen here is $\delta_\sigma(x - x_m) =$

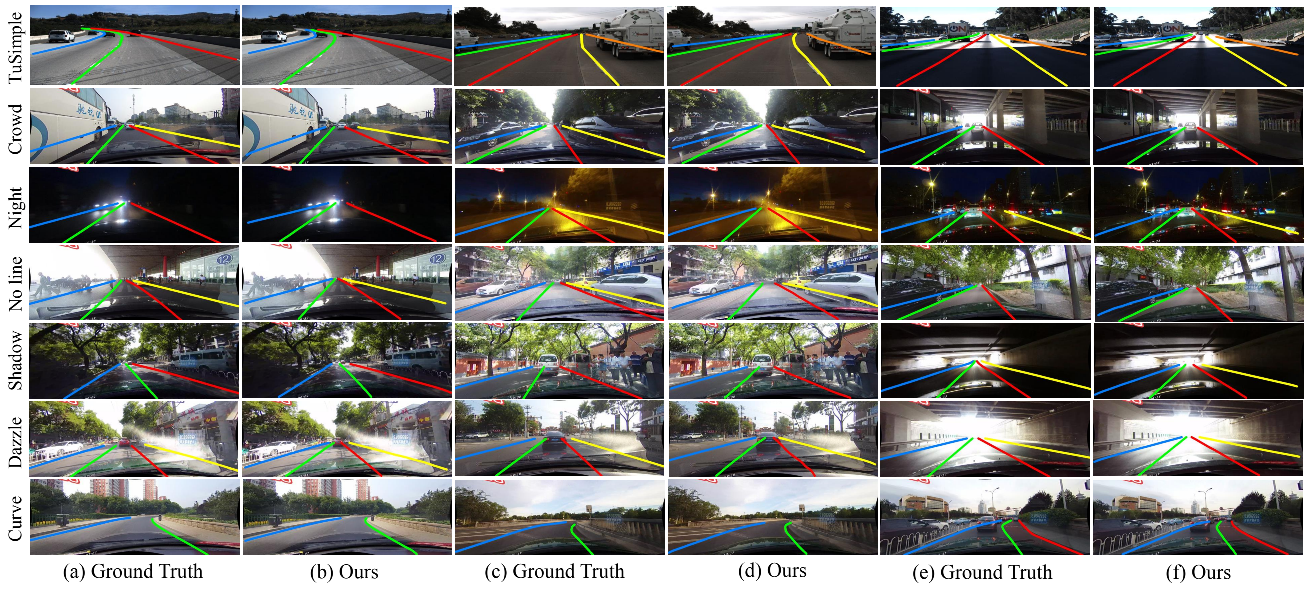


Figure b. The Ground Truths of the predictions on TuSimple and CULane in Fig. 7.

$\begin{cases} \frac{1}{2\sigma}, & \text{if } x \in \mathrm{U}(x_m, \sigma) \\ 0, & \text{if } x \notin \mathrm{U}(x_m, \sigma) \end{cases}, m = 1, \ldots, M$, where x_m is one of the x-coordinates on a lane. Therefore, the Delta function applied here can be the derivative of a smooth Heaviside function $H_\sigma(\cdot) - 0.5$ mentioned in Sec. 3.1, satisfying equation $\nabla(H_\sigma(\phi) - 0.5) = -\delta_\sigma(\gamma)\boldsymbol{n}$. Thus the calculation of EIE loss becomes the same as Eqs. (1) and (2) in Sec. 3.2.

The velocity field from Eq. (8) becomes:

$$\begin{aligned} v(x,y) &= \gamma_{p_t}(x,y) \\ &= -\frac{1}{4\pi}\int_{\mathbf{R}^2} \frac{\boldsymbol{r}\cdot\nabla\left(G_t - \alpha\Psi_p\right)(x,y)}{r^3}\,\mathrm{d}x\mathrm{d}y, \end{aligned} \tag{9}$$

which can be calculated efficiently via FFT, like Eq. (3).

We guess the degradation of ElasticLaneNet^{pw} on TuSimple and CULane, and the weak performance on geometry-diverse dataset SDLane is because of the non-smooth initial delta functions caused by the randomly initial $x-$sampling. On the contrary, implicit representation of *ELM* can be immediately attracted to the map of ground truth from a more general initial condition. This explicit method may be able to obtain better approximation results by improving the initial sampling distribution or choose a smooth regularized delta function instead.

D. More Results Discussion

D.1. Ground Truth Supplement

In order to save space, the Ground Truths are not provided in the Fig. 7 in text. The supplement of the GT of the results in Fig. 7 are in Fig. b.

D.2. Results Discussion on CULane

Some differences between our predictions and GT

In order to make further improvement on the dataset CULane, we compare our results and the GT via visualization. A small portion of results in three of nine categories in CULane scenes are displayed; see Fig. c. The rows 1-2 are "Night", the rows 3-4 are "Cross", the row 5 is "No line", and the row 6-7 include all types above.

The 1st and 5th rows are examples of some continuous frames, our predictions are more consistent, but they are different from the label-changing GT. Both ours and the GT look acceptable and reasonable. Besides, the predicted road at (f) in the 2nd row is narrower than the GT at (e). In fact, there is no clear line on the road and it is difficult to visually tell which ones (ours or GT) are better.

The images in the class of "Cross" (row3-4) are defined as no lane in CULane. Our model predicts False Positive (FP) results on "Cross" type in CULane sometimes, where some of these images do have lanes and look similar to other types of driving scene. It may because that the *ELM* trained via EIE loss is sensitive to geometry features in the input images, thus ElasticLaneNet can infer more geometry details related to the labeling patterns in the dataset. As a result, ElasticLaneNet can predict the lanes on the crossroad even when no label is provided in CULane. In the jointly training scheme of ElasticLaneNet, and the CSN module is responsible to eliminate the FP in majority. Developing a more suitable CSN module or scheme to distinguish the "Cross" type image adapting to CULane may increase the $F1$ score on it. On the other hand, the crossroads actually

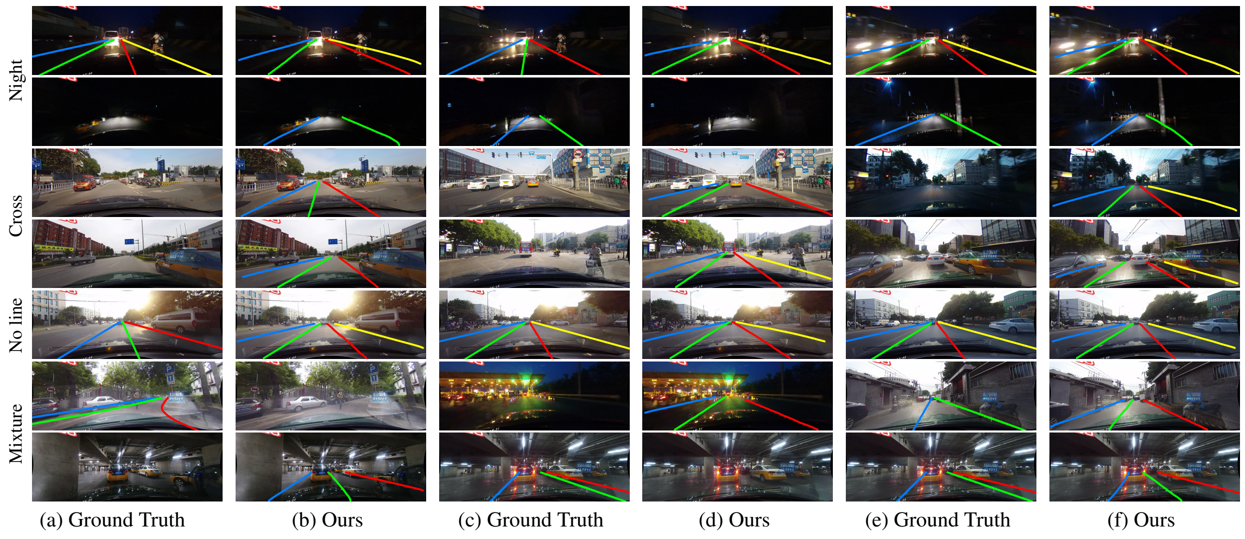


Figure c. More comparison of ElasticLaneNet on CULane dataset that are different from the ground truth according to different categories, which are "Night" (row 1-2), "Cross" (row 3-4), "No line" (row 5) and the mixture of "Night", "No line" and "Cross" (row 6-7).

have many complex lane structures (Y-shaped and merge lanes, L-shape turns, crossing, etc.). When these lanes need to be detected (such as in SDLane), our model is state-of-the-art (SOTA).

The 2nd and 7th rows show some near frames and the 6th row is a mixture of "Night" and "No line". It seems that both our predictions and the GT are not very self-consistent here. Part of the reasons might because the CULane covers a lot of scenes and many scenes are mixture types, and manual lane marking is sometimes subjective, it is difficult to formulate a very uniform labeling standard.

The situations above are some cases when our results are different from the GT. As we can successfully predict the lanes when the label of this image is not given (especially the "Cross" type in CULane), this phenomenon suggests that our approach may have the application potential in semi-supervised learning when the data is partially labeled, which requires the model to have a strong capability on drawing conclusion on labeled data features and on producing pseudo labels on unlabeled data, or to learn the consistency features from data.

Results Discussion

As ElasticLaneNet is a geometry-flexible model using implicit map (*ELM*) to represent lanes, it has large degrees of freedom (DOF) to express shapes and structures of lanes, but might bring some instability (mentioned in Sec.3 in [12]), especially when the simple structure (straight and parallel) lanes are in majority. Besides, the thresholds of computing TP in Acc in TuSimple (TPR on one lane >0.85) and the IoU in CULane (IoU >0.5) allow some slightly curved lanes to be predicted as straight lanes rather than count them as FP [28], thus most of the images in CULane and TuSimple can be handled by less DOF models when evaluating on the official metrics above. For instance, the 3rd order polynomial $x = ay^3 + by^2 + cy + d$ with more parameters and DOF can express more curves than the lower order polynomial, but slightly trade off on TuSimple's Acc measure, the second-order polynomial scores higher than the third-order one [28] without showing its superior on curves. These might be part of the reasons that the advantage of our model is not significant on CULane which contains less then 2% curves.

Failure Cases and Future Directions

In addition to the occasional lane existence error, other examples of instability caused by high DOF mentioned above can be seen in the 2nd row in Fig. c, which are not as straight as those in GT. Therefore, developing a more suitable CSN module adapting to the CULane, and more robust training scheme that further improve the stability of *ELM* are the possible future directions.

E. Simple Diagrams of Different Types of Lane Models

In this section, simple diagrams of different types of lane representations mentioned in Sec. 2 are shown, including segmentation-based, parameter-based, anchor-based, row-wise based on coarse grid maps (CGM) methods; see Fig. d. In (a), the coordinates of the left-most lane need to be determined via dense post-processing from pixels mask (the green one). Figures (b) and (d) show the lane point x_i of the left-most lane on the row i are calculated by param-

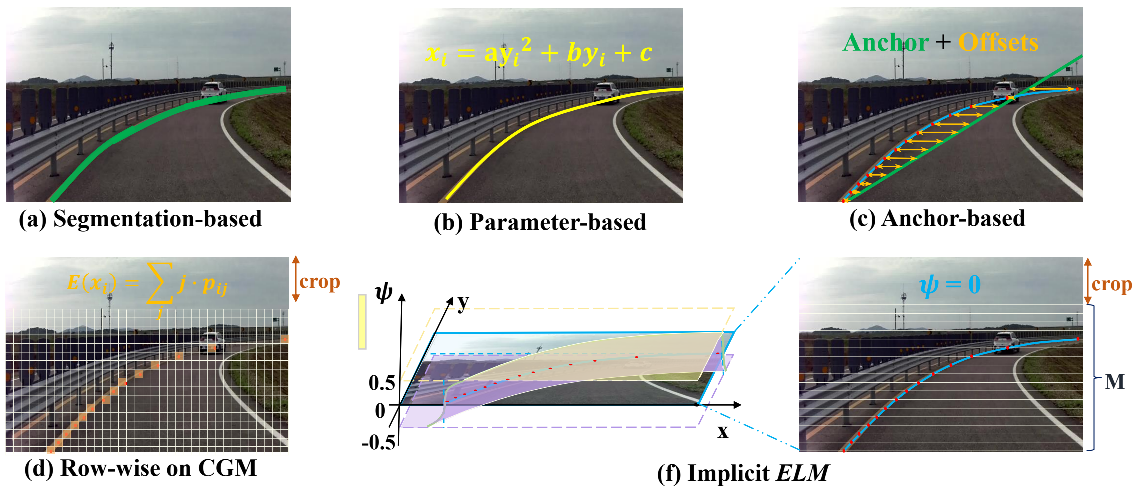
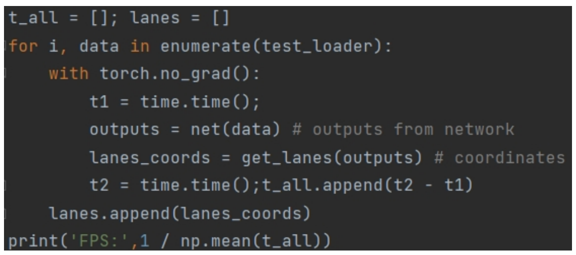


Figure d. The simple diagrams of the models in the Sec. 2.

eterizing the curve (*e.g*. 2nd order polynomial) and the row expectation, respectively, where j represents the column index. In (c), the lane points of the left-most lane are located by the sum of a chosen prior anchor and the predicted offsets according to it. In (f), the lane is on the zero-contour on the proposed *ELM*.

F. Configuration Settings of Compared Models

In Sec. 4.3.1, we perform comparison study on highly structure diverse lane data SDLane [12]. We re-train five of the SOTA methods from the last two years to convergence, including segmentation-based method LaneAF [1] and row-wise methods UFLDv2 [26], CondLaneNet [19], parameter-based method BezierLaneNet [5], anchor-based method CLRNet [37], and compare the results with those of our ElasticLaneNet; see Tab. 1 and Fig. 6. The pseudo code of FPS comparison is shown in Fig. e



```
t_all = []; lanes = []
for i, data in enumerate(test_loader):
    with torch.no_grad():
        t1 = time.time();
        outputs = net(data) # outputs from network
        lanes_coords = get_lanes(outputs) # coordinates
        t2 = time.time();t_all.append(t2 - t1)
    lanes.append(lanes_coords)
print('FPS:',1 / np.mean(t_all))
```

Figure e. Pseudocode of FPS comparison.

[1] is one of the latest segmentation-based methods with cluster post-processing according to affinity field predictions. Other setting remains the same as its original code for CULane.

[26], an efficient row-wise method proposed in year 2022, is an improved version of [25], which applies hybrid anchors of row and column to improve the predictions on side lanes that mostly are perpendicular to the y-sampling direction. The classification dimensions are set to be 100 on column anchors, 200 on row anchors, and the number of anchors are 80 columns and 72 rows.

[19] is an effective and popular row-wise method. We apply their RIM version designed for forks lanes. Other settings remain the same as the provided code.

[5] is a recent parameter-based method, which has a sensitive exploration capability on challenging lane cases. This method requires reconstructing the label to be parametric curves, e.g. bezier or polynomial curves. As SDLane has a high proportion of curves and a variaty of lane structures, we use 3 degree Bezier curves to reconstruct the label before training.

[37] is a state-of-the-art approach on TuSimple and CULane datasets using anchor-based model. We directly use the setting of the CULane's configuration file for SDLane, which performs better than TuSimple's configuration.