Rethinking Efficient Lane Detection via Curve Modeling
CVPR 2022

Zhengyang Feng1∗ , Shaohua Guo1*, Xin Tan2,1 , Ke Xu3 , Min Wang4 , Lizhuang Ma1,2,5†
1Shanghai Jiao Tong University 2East China Normal University 3City University of Hong Kong
4SenseTime Research 5MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University

Speaker: Po-Jui Su
01, January 2025

Outline

01
Introduction

02
Method

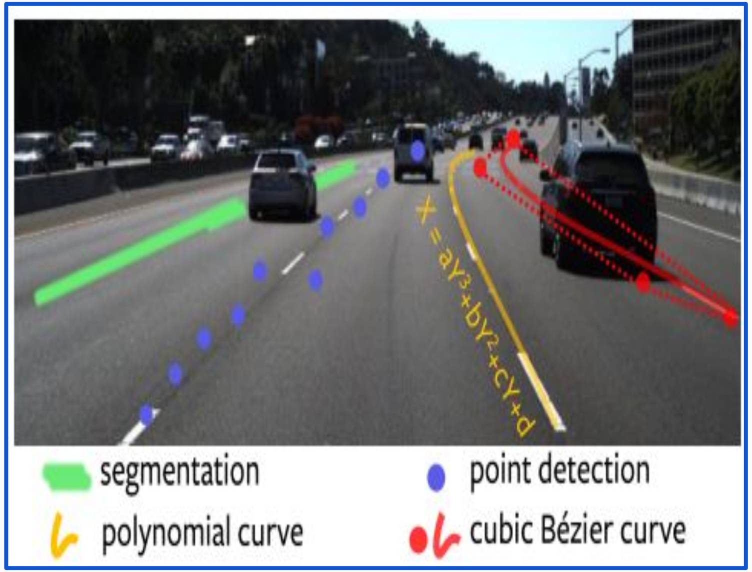
03
Experiments

04
Conclusion

01 Introduction

Lane detection tasks

Lane detection tasks involves identifying lane markings on roadways using image processing techniques, and it is currently widely used in autonomous driving technology.



Introduction: Current Approaches

Current Approaches:

- **Segmentation-based Lane Detection:**

  Segments lanes by classifying each pixel in the image and uses post-processing to aggregate these pixels into lane instances.

- **Point Detection-based Lane Detection**:

  Detects lanes as a sequence of discrete points along the vertical axis, usually **employing object detection frameworks (eg: DETR).**

- **Curve-based Lane Detection:**

  Represents lanes as holistic curves, **typically using polynomial equations** to model lane geometry and optimizing coefficients for precise curve fitting.

Introduction: Problem Definition

Challenges and Limitations in Existing Methods:

- **Segmentation-based Lane Detection:**

  Relies on per-pixel segmentation and heavy post-processing, **struggling with occlusions and challenging lighting conditions.**

- **Point Detection-based Lane Detection**:

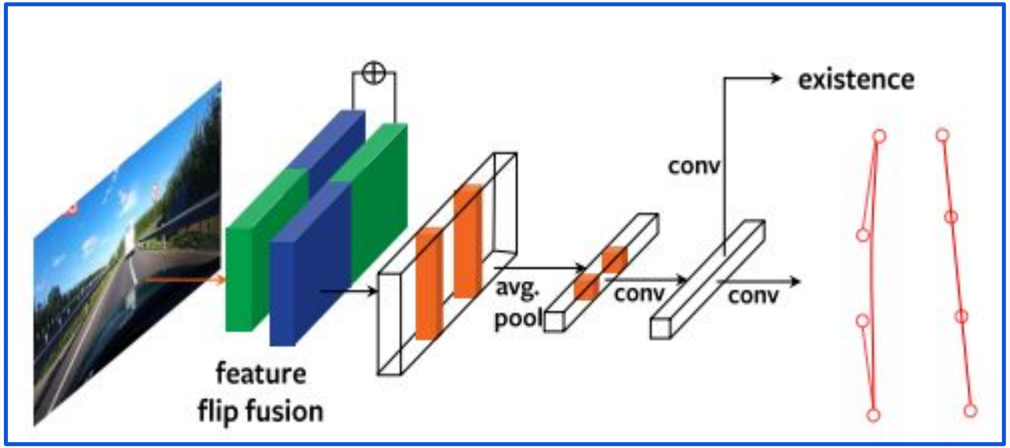
  Requiring Non-Maximum Suppression, the use of anchors and heuristics in point detection-based methods is **highly dataset-dependent, limiting their generalization.**

- **Curve-based Lane Detection:**

  Optimization is challenging due to slow convergence and computational overhead, leading to **lagging performance on complex datasets.**

Introduction: Purpose and Contribution

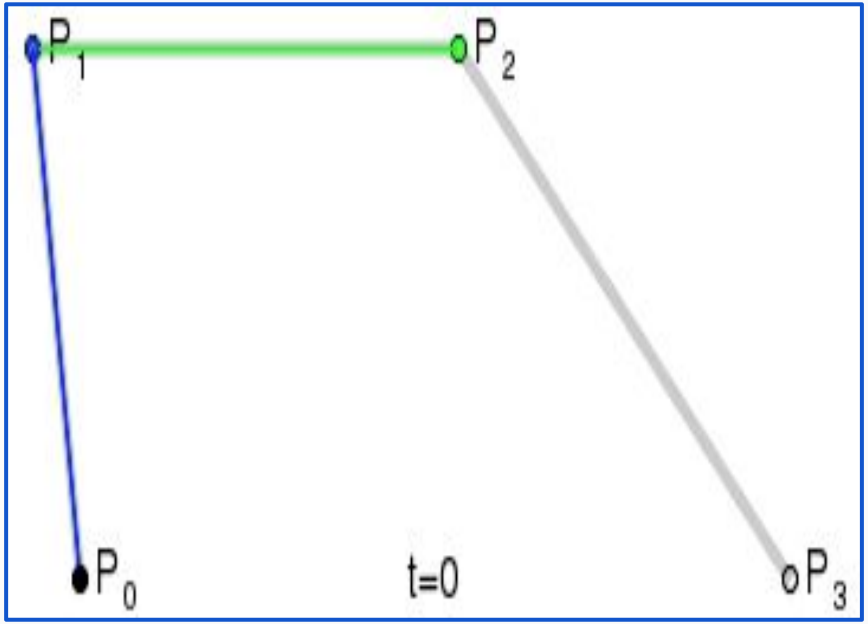
To overcome the limitations of existing methods, such as **high post-processing requirements**, **insufficient generalization and lagging performance**, this study proposes a **Bézier curve-based deep learning model (BézierLaneNet)** and introduces a **deformable convolution-based feature flip fusion module** to enhance the performance of existing approaches.



BézierLaneNet Architecture

02 Method

Bézier Curve and Considerations for the Value of n



Bézier Curve

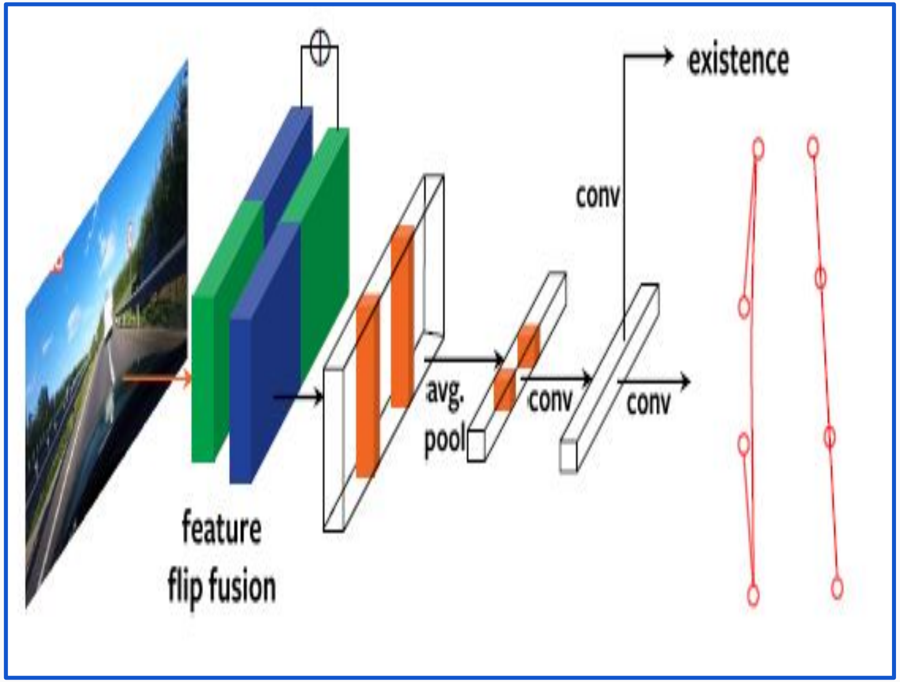
- Fitting Capability

- Degree of Freedom

- Efficiency and Real-Time Requirements

- Considering the above factors, the authors ultimately **set the value of n to 3.**

Architecture



- **Image and Feature Extraction:**

The image inputs into an encoder (ResNet) which **extracts high-level features** from the image.

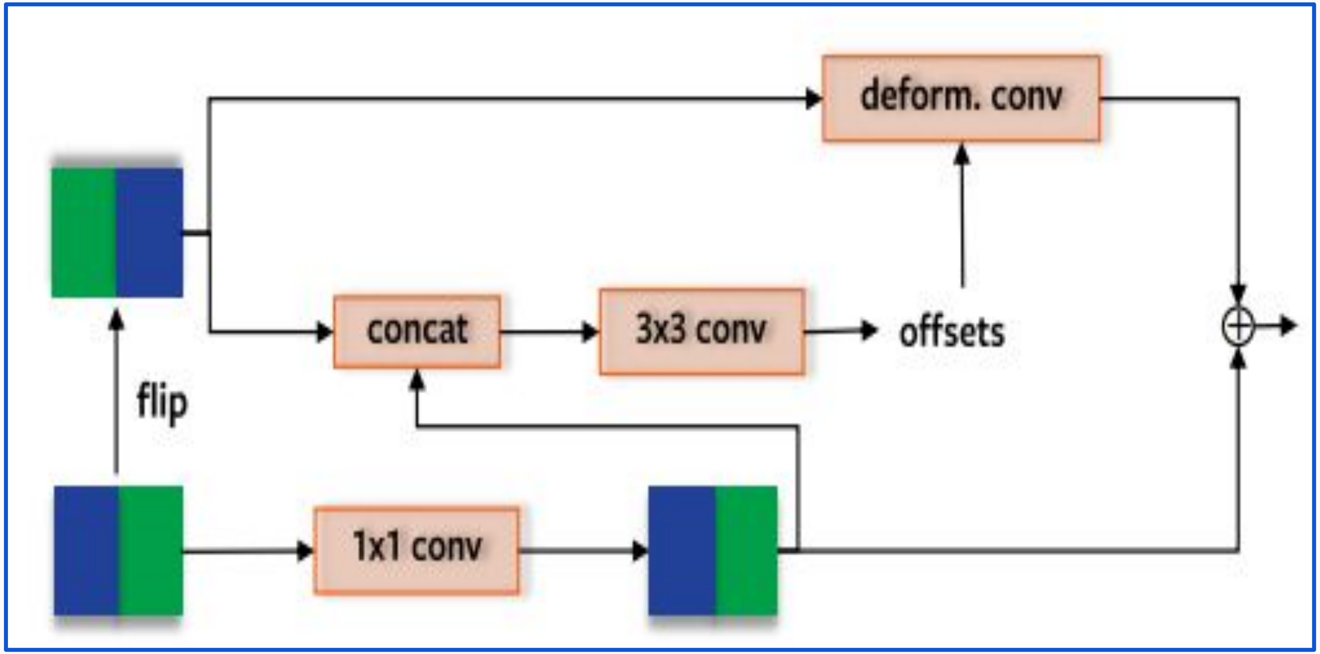
- **Feature Flip Fusion:**

Features are horizontally flipped and merged with original features to **enhance lane line symmetry** in the detection process.

- **Output Prediction:**

The processed features are used to predict lane presence through **classification** and **regression branches**, each containing a **1x1 convolutional layer.**

Feature Flip Fusion



Ground Truth Generation

$$\begin{bmatrix} b_{0,n}(t_0) & \cdots & b_{n,n}(t_0) \\ b_{0,n}(t_1) & \cdots & b_{n,n}(t_1) \\ \vdots & \ddots & \vdots \\ b_{0,n}(t_m) & \cdots & b_{n,n}(t_m) \end{bmatrix} \boxed{\begin{bmatrix} \mathcal{P}_0 \\ \mathcal{P}_1 \\ \vdots \\ \mathcal{P}_n \end{bmatrix}} = \boxed{\begin{bmatrix} k_{x_0} & k_{y_0} \\ k_{x_1} & k_{y_1} \\ \vdots & \vdots \\ k_{x_m} & k_{y_m} \end{bmatrix}}$$

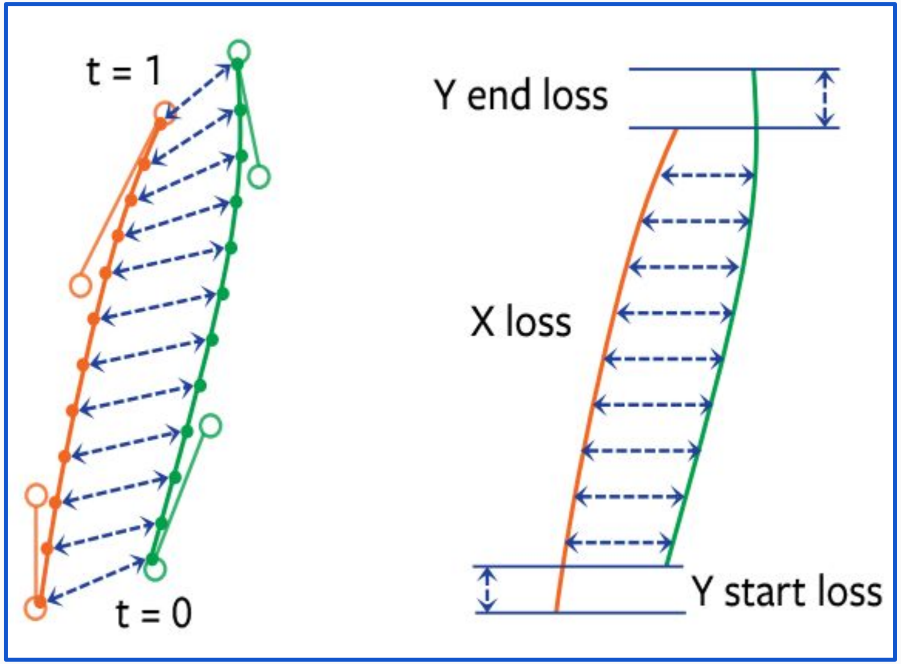
$$\boxed{\mathcal{B}(t) = \sum_{i=0}^{n} b_{i,n}(t)\mathcal{P}_i,\ 0 \le t \le 1,}$$

map groundtruth to Bézier curve(n=3)

P: corresponding control points, k: ground truth points

Distances Between Bézier Curves

$$\mathcal{L}_{reg} = \frac{1}{n}\sum_{t\in T}||\mathcal{B}(f(t)) - \hat{\mathcal{B}}(f(t))||_1,$$



Label and Prediction Matching

$$\hat{\pi} = \underset{\pi \in \Pi_G^N}{\arg\max} \sum_{i}^{G} Q_{i,\pi(i)},$$

$$Q_{i,\pi(i)} = \left(\hat{p}_{\pi(i)}\right)^{1-\alpha} \cdot \left(1 - L_1\left(b_i, \hat{b}_{\pi(i)}\right)\right)^{\alpha},$$

Equation Explanation:

- Match prediction curve and groundtruth
- Find G most possible curves from N predictions
- p: probability given by classifier
- L1: lane distance, alpha: weight coefficient(0.8 in this paper)

Training Loss

1. **Overall Loss:**

$$\mathcal{L} = \lambda_1 \mathcal{L}_{reg} + \lambda_2 \mathcal{L}_{cls} + \lambda_3 \mathcal{L}_{seg}$$

2. **Loss Function:**

- **L_{reg}: Quantifing curve differences by uniformly sampling across t values.** #1
- **L_{cls} : Classifying the existence of lane lines.** #2
- **L_{seg}: Classifying each pixel as lane region or non-lane region.** #3

3. **Weights for Loss: $\lambda_1 = 1$; $\lambda_2 = 0.1$; $\lambda_3 = 0.75$**

$$\mathcal{L}_{reg} = \frac{1}{n} \sum_{t \in T} ||\mathcal{B}(f(t)) - \hat{\mathcal{B}}(f(t))||_1,$$

$$\mathcal{L}_{cls} = -(y \log(p) + w(1 - y) \log(1 - p))$$

#1 Mentioned in the previous slide.
#2、3 Using Weighted Binary Cross Entropy

03 Experiments

Experiment Details

Dataset:

- TuSimple、CULane、LLAMAS

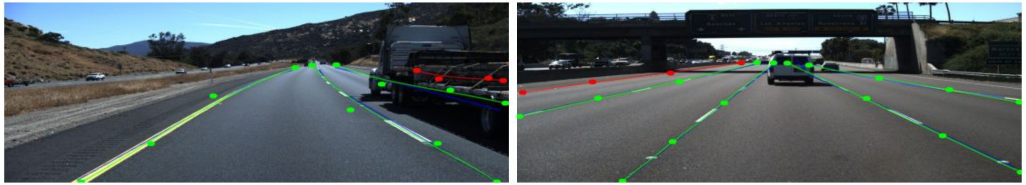
Evaluation Metrics:

- **F1 Score↑、Accuracy↑、FPR↓、FNR↓**

Training Configuration:

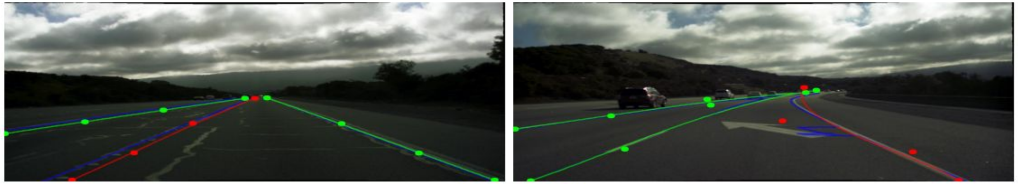
- **GPU:** 1× RTX 2080 Ti
- **Optimization:** Adam
- **Scheduler:** Cosine Annealing
- **Learning rate:** 6E-4
- **Weight Decay:** 1E-4
- **Batch size:** 20

Dataset	Train	Val	Test	Resolution	#Lines
TuSimple [1]	3268	358	2782	720×1280	≤ 5
CULane [22]	88880	9675	34680	590×1640	≤ 4
LLAMAS [3]	58269	20844	20929	717×1276	$\leq 4^*$



(a) TuSimple [1].

(b) CULane [22].



(c) LLAMAS [3].

Results on CULane and TuSimple Dataset

Method	CULane [22]											TuSimple [1]				
	Ep.	Total	Normal	Crowd	Night	No line	Shadow	Arrow	Dazzle light	Curve	Cross ↓	train+val	Ep.	Acc.	FPR ↓	FNR ↓
Segmentation-based																
Baseline (ResNet-18)*	12	65.30	85.45	62.63	61.04	33.88	51.72	78.15	53.05	59.70	1915		50	94.25	0.088	0.089
Baseline (ResNet-34)*	12	69.92	89.46	66.66	65.38	40.43	62.17	83.18	58.51	63.00	1713		50	95.31	0.064	0.062
Baseline (ResNet-101)*	12	71.37	90.11	67.89	67.01	43.10	70.56	85.09	61.77	65.47	1883		50	95.19	0.062	0.062
SCNN (ResNet-18) [22]*	12	72.19	90.98	70.17	66.54	43.12	66.31	85.62	62.20	65.58	1808		50	94.77	0.075	0.074
SCNN (ResNet-34) [22]*	12	72.70	91.06	70.41	67.75	44.64	68.98	86.50	61.57	65.75	2017		50	95.25	0.063	0.063
SCNN (ResNet-101) [22]*	12	73.58	91.10	71.43	68.53	46.39	72.61	86.87	61.95	67.01	1720		50	**95.69**	0.052	0.050
UFLD (ResNet-18) [26]**	50	68.4	87.7	66.0	62.1	40.2	62.8	81.0	58.4	57.9	1743	-	–	–	–	–
UFLD (ResNet-34) [26]**	50	72.3	90.7	70.2	66.7	44.4	69.3	85.7	59.5	69.5	2037	-	–	–	–	–
RESA (ResNet-18) [41]*	12	72.90	91.23	70.57	67.16	45.24	68.01	86.56	64.32	66.19	1679		50	95.24	0.069	0.057
RESA (ResNet-34) [41]*	12	73.66	91.31	71.80	67.54	46.57	72.74	86.94	64.46	67.31	1701		50	95.15	0.069	0.059
RESA (ResNet-101) [41]*	12	**74.04**	91.45	71.51	69.01	46.54	75.83	87.75	63.90	68.24	1522		50	95.56	0.058	0.051
Point detection-based																
FastDraw (ResNet-18) [25]	–	–	–	–	–	–	–	–	–	–	–	✓	7	94.9	0.061	0.047
CurveLanes-NAS-S [39]	12	71.4	88.3	68.6	66.2	47.9	68.0	82.5	63.2	66.0	2817	-	–	–	–	–
CurveLanes-NAS-M [39]	12	73.5	90.2	70.5	68.2	48.8	69.3	85.7	65.9	67.5	2359	-	–	–	–	–
CurveLanes-NAS-L [39]	12	74.8	90.7	72.3	68.9	49.4	70.1	85.8	67.7	68.4	1746	-	–	–	–	–
LaneATT (ResNet-18) [33]**	15	74.88	90.98	72.78	68.61	48.23	69.68	85.44	65.43	63.18	1163	✓	100	95.57	0.036	0.030
LaneATT (ResNet-34) [33]**	15	76.42	91.94	74.76	70.32	49.17	77.68	88.14	65.92	68.07	1323	✓	100	95.63	0.035	0.029
LaneATT (ResNet-122) [33]**	15	**76.79**	91.50	76.04	70.43	50.29	75.96	86.16	68.99	63.99	1265	✓	100	**96.10**	0.056	0.022
Curve-based																
PolyLaneNet (EfficientNet-B0) [32]**	–	–	–	–	–	–	–	–	–	–	–	✓	2695	93.36	0.094	0.093
LSTR (ResNet-18, 1×) [19]*	–	–	–	–	–	–	–	–	–	–	–		2000	95.06	0.049	0.042
LSTR (ResNet-18, 2×) [19]*	150	68.72	86.78	67.34	59.92	40.10	59.82	78.66	56.63	56.64	1166	-	–	–	–	–
BézierLaneNet (ResNet-18)	36	73.67	90.22	71.55	68.70	45.30	70.91	84.09	62.49	58.98	996		400	95.41	0.053	0.046
BézierLaneNet (ResNet-34)	36	**75.57**	91.59	73.20	69.90	48.05	76.74	87.16	69.20	62.45	888		400	**95.65**	0.051	0.039

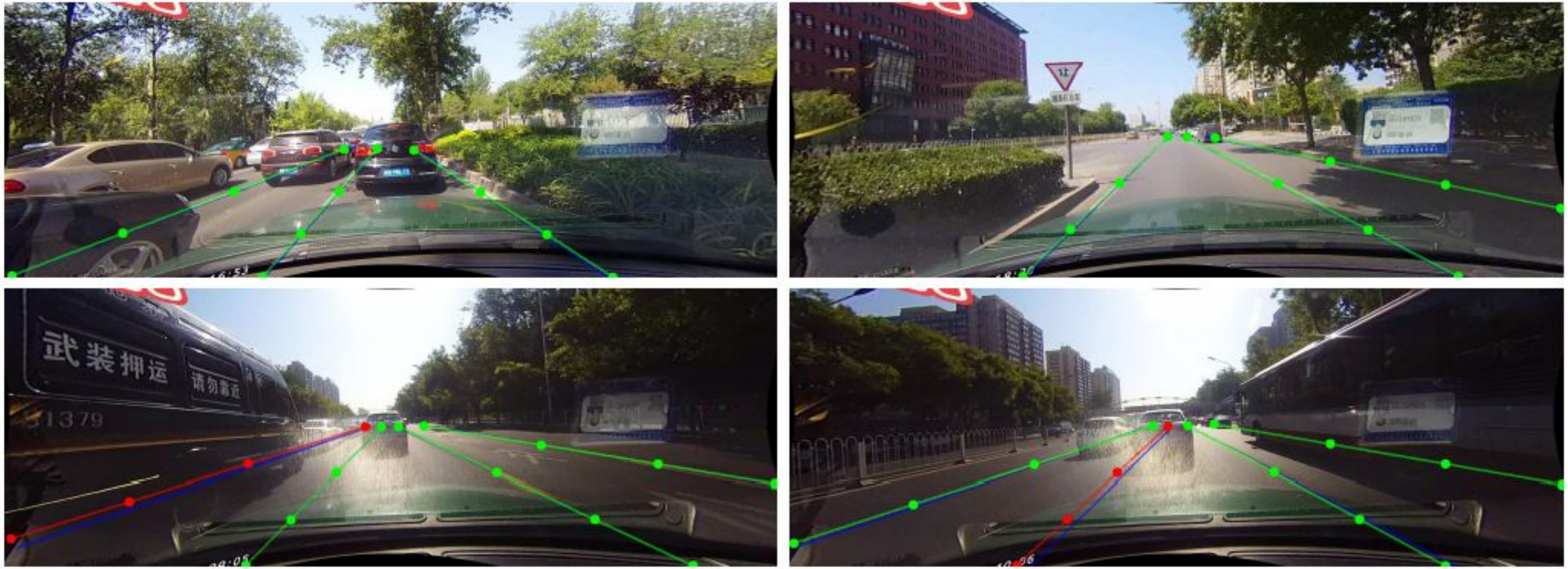
Results on LLAMAS Dataset

	LLAMAS [3]			
Method	**Ep.**	**F1**	**Precision**	**Recall**
Segmentation-based				
Baseline (ResNet-34)*	18	93.43	92.61	94.27
SCNN (ResNet-34) [22]*	18	94.25	94.11	94.39
Point detection-based				
LaneATT (ResNet-18) [33]**	15	93.46	96.92	90.24
LaneATT (ResNet-34) [33]**	15	93.74	96.79	90.88
LaneATT (ResNet-122) [33]**	15	93.54	96.82	90.47
Curve-based				
PolyLaneNet (EfficientNet-B0) [32]**	75	88.40	88.87	87.93
BézierLaneNet (ResNet-18)	20	94.91	95.71	94.13
BézierLaneNet (ResNet-34)	20	**95.17**	95.89	94.46

FPS and model size

Method	FPS ↑	Params (M) ↓
Segmentation-based (ignored post-processing time)		
Baseline (ResNet-101)	27	43.56
SCNN (ResNet-18) [22]	21	12.63
SCNN (ResNet-34) [22]	21	22.74
SCNN (ResNet-101) [22]	14	44.15
UFLD (ResNet-34) [26]	144	71.58
RESA (ResNet-18) [41]	68	6.61
RESA (ResNet-34) [41]	54	11.99
RESA (ResNet-101) [41]	25	31.46
Point detection-based (ignored NMS time in real images)		
LaneATT (ResNet-18) [33]	165	12.02
LaneATT (ResNet-34) [33]	117	22.13
LaneATT (ResNet-122) [33]	26	~~8.55~~ 85.5
Curve-based (entirely end-to-end)		
BézierLaneNet (ResNet-18)	**213**	**4.10**
BézierLaneNet (ResNet-34)	150	9.49

Experiment Results - Visualization



(b) CULane [22].

Ablation Study

CP	SP	Flip	Deform	Seg	F1
✓					63.74
	✓				68.89
	✓			✓	65.82
	✓	✓			70.28
	✓	✓	✓		72.96
	✓	✓		✓	73.97
	✓	✓	✓	✓	**75.41**

Table 7. Ablations. **CP**: Control point loss [20]. **SP**: The proposed sampling loss. **Flip**: The feature flip fusion module. **Deform**: Employ the deformable convolution in feature flip fusion. **Seg**: Auxiliary segmentation loss.

04 Conclusion

Comparison with Existing Methods

- **Curve-based Methods**

(1) BézierLaneNet runs over 2× faster than LSTR due to its fully convolutional, end-to-end design.

(2) It converges **4-5× faster** than LSTR, significantly reducing training time.

- **Segmentation-based Methods**

(1) BézierLaneNet outperforms segmentation-based methods in both **speed** and **accuracy**.

- **Point Detection-based Methods**

(1) Although Point Detection-based methods outperform BézierLaneNet in certain scenarios, **BézierLaneNet performs better in Dazzle Light scenarios, demonstrating greater stability.**

(2) It is approximately **30% faster** and requires **2.5× fewer parameters** compared to LaneATT.

Conclusion

- Introduced a novel lane detection framework leveraging **Bézier curves**, combined with a **feature flip fusion module** to utilize the symmetric property of road lanes, enhancing accuracy and robustness.

- Achieved SOTA performance across datasets with a **lightweight design (<10 million parameters)** and **real-time speed (>150 FPS)**, ensuring efficiency and practicality for real-world applications.