Lane Detection Transformer based on Multi-frame Horizontal and Vertical Attention and Visual Transformer Module

Han Zhang, Yunchao Gu ✉, Xinliang Wang, Junjun Pan, and Minghui Wang

Beihang University, XueYuan Road No.37, HaiDian District, BeiJing, China
{allenzhang,guyunchao,wangxinliang,pan_junjun,minghuiw}@buaa.edu.cn

Abstract. Lane detection requires adequate global information due to the simplicity of lane line features and changeable road scenes. In this paper, we propose a novel lane detection Transformer based on multi-frame input to regress the parameters of lanes under a lane shape modeling. We design a Multi-frame Horizontal and Vertical Attention (MHVA) module to obtain more global features and use Visual Transformer (VT) module to get "lane tokens" with interaction information of lane instances. Extensive experiments on two public datasets show that our model can achieve state-of-art results on VIL-100 dataset and comparable performance on TuSimple dataset. In addition, our model runs at 46 fps on multi-frame data while using few parameters, indicating the feasibility and practicability in real-time self-driving applications of our proposed method.

Keywords: Autonomous driving, Lane detection, Transformer

1 Introduction

Autonomous driving has been developing rapidly in recent years and has received full attention from academia and industry. In perception task—the "eyes" of autonomous driving, lane detection plays a significant role in understanding road environment. It relates to the self-positioning, observance of traffic rules, and subsequent decisions on control of autonomous vehicles.

Lane detection is challenging mainly due to two reasons. Firstly, the lanes[1] are slender in shape, single in structure, and rare in appearance clues. In addition, lanes may disappear due to various reasons such as wear and tear, shadow occlusion, road congestion, terrible weather conditions, or dazzling light. However, humans can easily determine the location of lane lines based on vehicle alignment, road shape, visible local lane lines information, etc. Inspired by the human visual system, people found that the global information containing additional visual clues is preferable to detecting the lane lines.

[1] For academic consistency, we use "lane" to denote lane line in this paper.

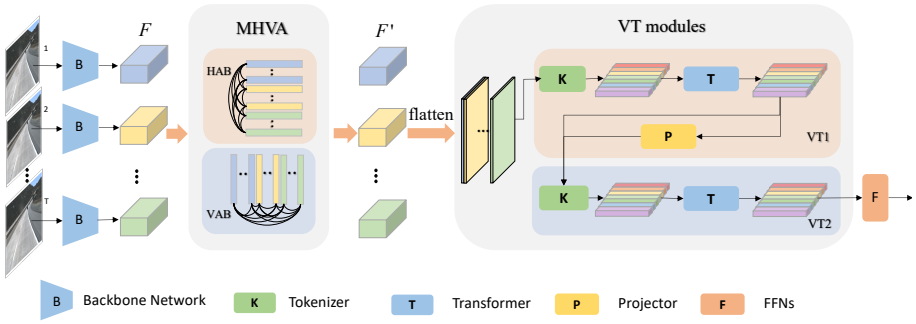


Fig. 1: Overall Framework. Continuous frames are compressed to 1/8 of origin through the backbone network. Then the compressed features are fed into the MHVA to pass messages between rows or columns. The VT modules are used to generate "lane tokens" and enable interaction between lane instances. Finally, the parameters are obtained through FFNs with the tokens as input

Many methods [12, 24, 16, 2, 7] focus on obtaining sufficient global features from current frame. Among them, LSTR [7] proposes to use the most conventional Transformer to capture global information, but it does not take full advantage of the structural features of lane lines and the pixel-wise attention mechanism is computationally intensive. SCNN [12], on the other hand, designs a special CNN to capture spatial relationships of pixels across rows and columns. However, using the convolution operation which is good at processing local features, it needs a large number of local information transfers(e.g. slice-by-slice transfer, multiple iterations of convolution modules) to achieve global information interaction. That is the reason why it is computationally intensive and inefficient in terms of information interaction. The RESA [24] on top of it does not solve the problem fundamentally.

The importance of global features to lane detection can not be ignored, but it is obviously unrealistic and ill-considered to expect deep learning models which only use individual frames as input and completely ignore more visual information of previous frames in dynamic driving can infer complete and accurate detection results. Although there are some lane methods[25, 23, 22] use multi-frame input data, they are all segmentation-based. These methods need to classify each pixel, which are indirect and lead to higher computational cost. Besides, they also need extra post-processing steps to extract lane information.

To solve the above problems, we propose a novel end-to-end model using multiple frames as input, having the customized structure suitable for lane detection based on Transformer. To extract more global information from frames, we design a Transformer-Based structure called Multi-frame Horizontal and Vertical Attention (MHVA) module. It can solve the dilemma of SCNN, due to its advantage of establishing efficient and direct information interaction via attention module across multiple frames. Here are the two main differences from previous

work. First, MHVA builds the interaction among row and column features separately via direct attention, which adopts the divide-and-conquer idea to reduce the computation and fully makes use of horizontal and vertical features. Second, MHVA can be easily extended to detection based on multiple frames.

Besides, we use the VT module in our method. We notice a certain continuity between lanes, such as they are parallel and always intersect at vanishing points. Inspired by [19], we introduce the concept of "lane token" into lane detection to get lane instance features. By utilizing the VT module, we can assign the features obtained from MHVA to specific lanes, and establish relationship between lanes to realize information interaction. Finally, the regression task is built on tokens to get the parameters of each lane.

The main contributions of our method can be summarized as follows:

– We propose a novel curve-fitting lane detection method using multi-frame information and achieve instance-level lane detection.

– We design a lane detection Transformer based on MHVA and VT modules capturing more global information for parametric regression.

– The experiments on two public datasets verify the effectiveness of our method. In addition, we achieve the state-of-the-art results on the VIL-100 dataset and competitive performance on the TuSimple dataset, both with real-time speed.

2 Related Work

Early lane detection approaches are based on traditional computer vision methods. However, traditional methods have poor robustness in complex scenarios due to their dependence on highly-specialized and hand-crafted features. Recently, with the spring breeze of deep learning, the lane detection approaches based on it have sprung up. They become the mainstream solutions relying on powerful learning ability. We taxonomize these methods into four categories.

Segmentation-Based Method This kind of method regards lane detection as a segmentation problem. Some semantic segmentation methods [25, 10, 1] only distinguish the pixels of lanes from background, while others [9, 12, 24, 13, 8, 2] regard each lane as an individual classification category by detecting a fixed number of lanes. Methods [11, 3, 21] based on instance segmentation take each lane as an instance of a lane category.

Based on multi-frame input, [25] and [22] only segment lanes from background and both need post-processing to distinguish lanes. Different from them, MMANet [23] is an instance lane detection method using multi-frame input. But it is inefficient and requires a redundant extraction step. Instead of using segmentation-based paradigm, we choose to regress the parameters of curves.

Row-wise-classification-Based Method This kind of method is based on the prior domain knowledge that the number of intersection points between a horizontal line and each other lanes is most one. Thus, it formulates lane detection

as a row-wise classification problem by selecting the cell position for intersection points. Finally, a complete lane can be composed of these intersection points. This divide-and-conquer idea reduces the calculation cost, but they also need a post-processing step to extract lanes. [20] and [14] are representative models of this method.

Anchor-Based Method Inspired by Faster-RCNN [15], Line-CNN [6] proposes the first anchor-based lane detection method. Its core design LPU (line proposal unit) predicts lines from the original straight proposal line. However, due to the lack of ability to capture global information, this method is inefficient. Based on [6], LaneATT [16] proposes an anchor-based attention mechanism to capture global information.

Curve-fitting-Based Method Lane detection is regarded as a parametric regression problem which use polynomial curves to express lanes in curve-fitting-based method. PolyLaneNet [17] connects a regression part after backbone network. Although it is efficient, the lack of global information leads to a gap with other methods in terms of performance. On this basis, R. Liu *et al.* [7] change the lane modeling method and use Transformer to learn more global features. They are both based on single-image input and discard the visual information in previous frames. Using continuous multi-frame input, we propose a more powerful lane detection Transformer to capture more global information.

3 Approach

We propose a novel lane detection method based on curve fitting, whose framework is shown in Fig. 1. It receives several time-ordered RGB images taken from a camera mounted in the vehicle as input and outputs the parameters of the predicted lanes. It consists of a backbone network, a MHVA module, several VT modules, and feed-forward networks (FFNs) for parametric regression. Given several continuous images in order, the backbone network first extracts a high-level feature map F. Then, the feature map F and positional embedding E are fed into the MHVA module to get the enhanced feature map F'. After received "lane tokens" through VT modules, FFNs will regress the parameters on them. Hungarian fitting loss are used to train our network.

3.1 Lane Shape Model

Following the lane shape model in LSTR [7], we regress the cubic polynomial parameters related to the internal and external parameters of the camera and the angle between the camera with the ground plane. In the image coordinate system, the modeling formula is as follows:

$$x = \frac{k}{(y-f)^2} + \frac{m}{y-f} + n + b \times y - b', \tag{1}$$

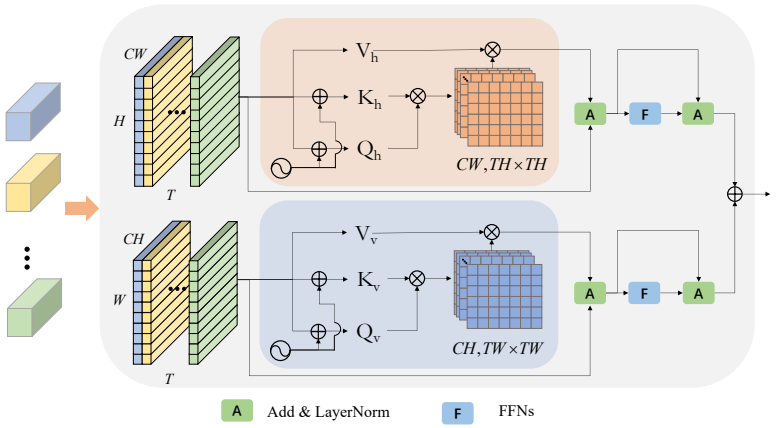


Fig. 2: MHVA. It has a HAB branch for interaction among rows and a VHB for interaction among columns. The final output is the sum of the results of the two branches

where (x, y) is position coordinate, n and b' are constants terms that cannot be integrated, and k, f, m, b are variable parameters.

For one lane l, it relates to eight parameters $(k, f, m, n, b_l, b_l^{'}, \alpha_l, \beta_l)$. The first four parameters are shared parameters among all lanes, and b_l and $b_l^{'}$ are particular parameters of lane l. α_l and β_l are vertical starting offset and ending offset respectively. Kindly refer to [7] for more details.

3.2 The MHVA Module

Transformer is popular in computer vision due to its effectiveness in capturing long-distance dependence. Therefore, to capture global content information from a multi-frame feature map, the intuitive idea is to build attention connection at pixel level by Transformer just as in [18]. However, when only focusing on lanes, we construct a more efficient module called Multi-frame Horizontal and Vertical Attention (MHVA) based on the prior knowledge of lane structure and frame continuity.

When we input T frames, the feature map $F \in \mathbb{R}^{T\times C\times H\times W}$ is obtained from the backbone network. Instead of building the pixel level connections, MHVA models the relationships among all row features and all column features respectively. It decomposes the $THW \times THW$ connection into $TH \times TH$ and $TW \times TW$ connections. The two branches of MHVA are Horizontal Attention Branch(HAB) and Vertical Attention Branch(VAB). Fig. 2 shows the architecture of MHVA.

Position Embedding To supplement precise position information in three dimensions (temporal, horizontal, and vertical dimensions), we generate fixed positional encoding features with the same size as F. We change the position

embedding in the original Transformer to a 3D manner. For $F \in \mathbb{R}^{T\times C\times H\times W}$, the position embedding with size $T\times\frac{C}{3}\times H\times W$ in three dimensions respectively will be generated and concatenated in channel-wise dimension. Specifically, we use Eq. 2 to calculate the embedding's i-th channel of coordinate pos in each dimension:

$$PE(pos,i) = \begin{cases} \sin(pos\cdot\omega_k), & i = 2k, \\ \cos(pos\cdot\omega_k), & i = 2k+1, \end{cases} \tag{2}$$

where $\omega_k = \frac{1}{10000}^{\frac{2k}{C/3}}$.

Horizontal Attention Branch(HAB) The HAB enables message passing of horizontal features. A self-attention operation is performed on all row features. For $F \in \mathbb{R}^{T\times C\times H\times W}$, the row num is TH and the feature channel size is CW. Afterward, the feature map F and positional embedding E will be integrated to $TH\times CW$ and fed into attention module, with query, key, and value tensors (Q_h, K_h, V_h) of the same dimension $TH\times d\_model$ as output. Then, a $TH\times TH$ attention matrix A_h is obtained as Eq. 3:

$$A_h = softmax(\frac{Q_h K_h^T}{\sqrt{d\_model}}). \tag{3}$$

where A_{ij} represents the similarity between features of frame $[i/h]$, row $[i \bmod h]$ and frame $[j/h]$, row $[j \bmod h]$. Intuitively, the value of adjacent rows or corresponding rows in adjacent frames will be higher. Finally, we obtain a refined feature map using the attention matrix A_h and V_h tensor as follows:

$$F_h = A_h V_h. \tag{4}$$

Vertical Attention Branch(VAB) The VAB models the relationships between column features similar to that of HAB's. Differently, the feature map F and positional embedding E are resized to $TW\times CH$, and the attention matrix A_v is $TW\times TW$. Finally, F_v is given by the multiplication of A_v and V_v.

F_h and F_v will be resized to $T\times C\times H\times W$ and added together to get the final feature map F'.

3.3 The VT Module

Inspired by [19], we define the "lane tokens" to represent the lane instance, and use the Visual Transformer (VT) module to learn the compact lane instance features from the high-level features. A VT module involves three steps. The first step is to group pixels into lane instances to generate a serial of corresponding lane tokens. Then, a transformer module is performed to the lane tokens to model the relationships between them. Finally, we reproject the lane tokens to pixel level to obtain an augmented feature map. If multiple VT modules are used continuously, the module VT_i will build on the reprojected feature map of the previous VT module VT_{i-1}. In last VT module, the third step is omitted directly to regress parameters on lane tokens.

Tokenizer The function of tokenizer is to convert feature maps into compact sets of lane tokens. Formally, we resize feature map F' to $THW \times C$ and denote it as input by $X \in \mathbb{R}^{N \times C}(N = THW)$. The lane tokens are denoted as $T \in \mathbb{R}^{L \times C}$, where L is the fixed maximum number of lanes. The core mechanism for calculating tokens is attention. Firstly, an attention map $A \in \mathbb{R}^{N \times L}$ is generated where A_{ij} represents the contribution of one pixel $p_i \in \mathbb{R}^C$ to the lane instance j. Next, we compute the weighted averages of pixels in X to form lane tokens by multiplying A and X. Formally,

$$T = A^T X. \tag{5}$$

There are two ways to generate the attention map A for modules in different locations as shown in Fig 3. For the first VT module, we generate it as follows:

$$A = softmax_{_N}(XW_A), \tag{6}$$

the $W_A \in \mathbb{R}^{C \times L}$ in Eq. 6 forms lane instance groups from X, and the *softmax* operation converts the values into normalized attention values.

While for the subsequent VT modules, we use the output lane tokens T_{last} of its previous module to guide the extraction of current new lane tokens. Formally,

$$\begin{aligned} W_R &= T_{last} W_{T \to R}, \\ A &= softmax_{_N}(XW_R), \end{aligned} \tag{7}$$

in which $W_{T \to R} \in \mathbb{R}^{C \times C}$, $W_R \in \mathbb{R}^{C \times L}$.

Transformer After Tokenizer, we need to establish the relationship between lanes to realize information interaction. To dynamically model interactions, a standard transformer with minor changes is used, in which self-attention generates the input-dependent weights.

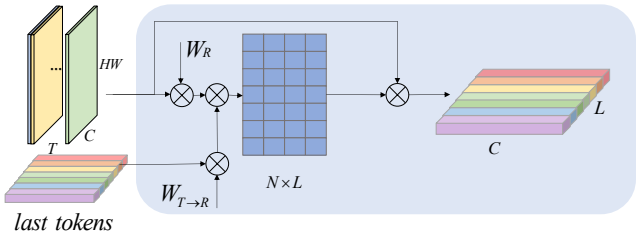
$$\begin{aligned} T'_{out} &= T_{in} + softmax_{_L}(T_{in} W_K (T_{in} W_Q)^T) T_{in}, \\ T_{out} &= T'_{out} + \delta(T'_{out} F_1) F_2, \end{aligned} \tag{8}$$

where T_{in}, T'_{out}, $T_{out} \in \mathbb{R}^{L \times C}$ are lane tokens, W_K and W_Q are key weight and query weight with size $C \times C$, and F_1, $F_2 \in \mathbb{R}^{C \times C}$ are linear weights. $\delta(\cdot)$ is the *ReLu* function.

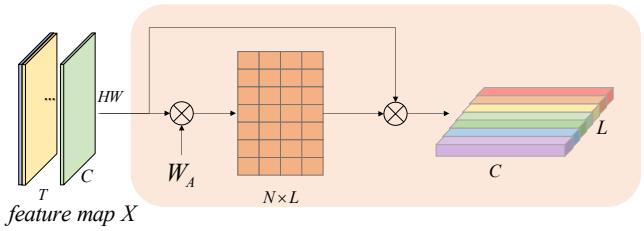
Projector When we overlay multiple VT modules, the role of the projector in the current VT is to generate the input feature map for the next VT module. When using only one VT module, there is no need to generate feature maps for the subsequent modules, parametric regression will be operated at lane tokens.

$$X_{out} = X_{in} + softmax_{_L}((X_{in} W_Q)(T W_K)^T) T, \tag{9}$$

where X_{in}, $X_{out} \in \mathbb{R}^{N \times C}$. W_Q, $W_K \in \mathbb{R}^{C \times C}$ are the weights for query and key tensors. The product of key and query determines how to project information encoded in lane tokens to the pixel level feature map.



(a) Tokenizer without previous tokens



(b) Tokenizer with previous tokens

Fig. 3: The two kinds of Tokenizer. The N is equal to THW

3.4 FFNs

Given the lane tokens $T \in \mathbb{R}^{L\times C}$, we use three branches to predict the parameters. In the classification branch, we use a linear operation to generate $C \in \mathbb{R}^{L\times 2}$, which indicates the probability of lane instances represented by each lane token. On another two branches, we use three-layer perceptrons to predict four shared parameters and four lane-specific parameters respectively. Those shared parameters will be averaged between lanes.

3.5 Loss Function

Our proposed method infers the predicted parameters of lane instances, which are stochastic in order. Thus we match them with the ground truth lanes first by utilizing the Hungarian algorithm. The regression loss is optimized based on the above matching result.

The L predicted curves are denoted as $\mathcal{H} = \{h_i | h_i = (c_i, m_i)\}_{i=1}^{L}$, where m is the set of eight parameters and $c \in \{0, 1\}$ represents the possibility of existence of lanes. The ground truth labels of lanes are represented by a set of key points $s = (x, y)_{r=1}^{R}$, where $y_{i+1} > y_i$, and the ground truth curves are denoted as $\mathcal{L} = \{l_i | l_i = (c_i', s_i')\}_{i=1}^{L}$. Since L is larger than the number of ground truth lanes, $\mathcal{L}$ will be padded with non-lane values.

Bipartite Matching In order to find a bipartite matching between the predicted parameters and the ground truth lanes, a permutation of L elements δ is

searched with the lowest cost:

$$\hat{\delta} = \arg\min_{\delta} \sum_{i}^{L} \mathcal{D}(l_i, h_{\delta(i)}), \tag{10}$$

where $\mathcal{D}$ is a pair-wise matching cost between the i-th ground truth lane and a predicted curve with index $\delta(i)$.

For the prediction curve with index δ_i, the probability of class c_i' is denoted as $p_{\delta(i)}(c_i')$, and the prediction key points $s_{\delta(i)}$ can be calculated by Eq. 1 based on parameters $m_{\delta(i)}$. The matching cost $\mathcal{D}$ is defined as:

$$\begin{aligned} \mathcal{D} = &- \omega_1 p_{\delta(i)}(c_i') + \mathbb{1}(c_i' = 1)\omega_2 L_1(s_i', s_{\delta(i)}) \\ &+ \mathbb{1}(c_i' = 1)\omega_3 L_1(\alpha_i', \alpha_{\delta(i)}, \beta_i', \beta_{\delta(i)}), \end{aligned} \tag{11}$$

where $\mathbb{1}$ is the indicate function, ω_1, ω_2 and ω_3 are the effect parameters, and L_1 is the mean absolute error.

Regression Loss After getting the matching results, we calculate the regression loss as follows:

$$\begin{aligned} L = \sum_{i=1}^{L} &-\omega_1 p_{\delta(i)}(c_i') + \mathbb{1}(c_i' = 1)\omega_2 L_1(s_i', s_{\delta(i)}) \\ &+ \mathbb{1}(c_i' = 1)\omega_3 L_1(\alpha_i', \alpha_{\delta(i)}, \beta_i', \beta_{\delta(i)}). \end{aligned} \tag{12}$$

4 Experiments

In this section, we evaluate the performance of our proposed method with the widely-used metrics on two public datasets TuSimple[2] and VIL-100 [23]. Afterward, we provide a detailed ablation study to prove the rationality of our design and the selected hyperparameters.

4.1 Dataset

VIL-100 VIL-100 is the first video instance-level lane detection dataset with all frames annotated. It contains 100 videos, 100 frames per video within 10 seconds, in total 10000 labeled frames. Among 100 videos, 97 are collected by the monocular forward-facing camera, and the remaining three videos are from the Internet. Besides, it contains one normal scenario and nine challenging scenarios. The training set has 80 videos and the rest belongs to the testing set.

TuSimple The TuSimple dataset is a widely-used dataset collected on America's highway with constant illumination and weather in the daytime. It consists of 6408 sequences, each of which contains 20 continuous frames collected in one second. We take the annotated 3626 clips for training and adopt the remaining 2782 clips for testing.

[2] https://github.com/TuSimple/TuSimple-benchmark.

4.2 Evaluation Metrics

On the TuSimple dataset, we use its three official metrics *Accuracy*, *FP* and *FN*. *Accuracy* refers to the average number of correct points per image. The standard for "correct points" is that when the vertical coordinate is the same, the horizontal distance between the predicted point and the ground truth point is smaller than 20 pixels. *Accuracy* can be defined in the following way:

$$Accuracy = \sum_{i=1}^{n} \frac{C_i}{S_i}, \tag{13}$$

where C_i is the number of correct points and S_i is the total number of ground-truth points in frame i, n is the number of total frames.

The false positive (FP) and false negative (FN) are computed as:

$$FP = \frac{F_{pred}}{N_{pred}}, \tag{14}$$

$$FN = \frac{M_{pred}}{N_{gt}}, \tag{15}$$

where F_{pred} is the wrongly predicted lanes, M_{pred} represents the number of missed lanes, N_{pred} is the total number of predicted lanes, N_{gt} is the total number of ground-truth lanes.

On the VIL-100 dataset, in addition to the above three metrics, we also use the *F1* metric. *F1* is a region-based metric based on the intersection-over-union (IoU). For one lane instance, we can get the predicted key points using the predicted parameters. A binary mask can be obtained by connecting these points with a line of 30 pixels in width. Then, if the IoU between the predicted mask and the ground truth label is larger than 0.5, the predicted lane can be considered as true positive (tp), otherwise, it is a false positive (fp). The missed lanes are denoted as fn. The metric *F1* is formulated as:

$$F1 = \frac{2 \times precision \times recall}{precision + recall}, \tag{16}$$

$$precision = \frac{tp}{tp + fp}, \tag{17}$$

$$recall = \frac{tp}{tp + fn}. \tag{18}$$

4.3 Implementation Details

The input data for our model are 5 continuous frames with size 360 × 640. Same data augmentation methods are used for these continuous frames, such as shadow, flipping horizontally, rotating, and color jittering.

The backbone network we use is a reduced ResNet18 which is the same as [7]. The output channels of four blocks are changed from "64, 128, 256, 512" to

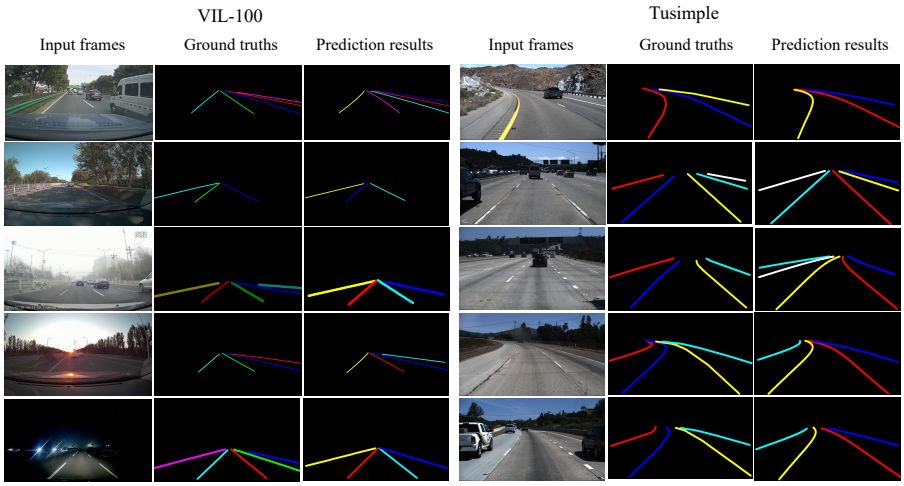


Fig. 4: Visualization results for VIL-100 and TuSimple dataset. The colors are just to distinguish different lane instances

"16, 32, 64, 128" and the downsampling factor is set to 8. We use the multi-head self-attention in all attention modules and the head number is set to 2. The number of VT modules is 2. The fixed number of predicted curves L is set to 7 and the loss coefficients $\omega_1, \omega_2, \omega_3$ are set to 3, 5 and 2 respectively.

We train our model on one TitanX GPU with batch size 8 in PyTorch. For both datasets, we use Adam optimizer with the base learning rate of 0.0001. The learning rate policy is StepLR with stepsize 300. The numbers of total training epochs are 1000 for TuSimple and 1200 for VIL-100 dataset separately.

4.4 Results

In this section, we treat LSTR as the baseline model since it is also a curve-fitting method based on Transformer. To show our performance, we compare our method with other state-of-the-art methods. Apart from the video instance lane detection method MMANet [23], the input of other methods are almost one image. However, because MMANet has not been trained on TuSimple, we only compare with it on VIL-100 dataset. Besides the above-mentioned four metrics, the FPS and the total number of parameters are also compared.

VIL-100 Table 1 shows the performance on VIL-100 dataset. Our proposed method outperforms the mentioned state-of-the-art lane detectors on four above-mentioned metrics. Compared with baseline method LSTR, ours shows significant improvement on four metrics, especially the *FP* is reduced by half. Compared with MMANet, the performance of our model raises 0.7% on *F1*, 0.5% on *Accuracy*, 2.9% on *FP*, and 1.0% on *FN* with 15 $\times$ fewer parameters and 5

Table 1: Quantitative comparisons on VIL-100 dataset. The FPS is evaluated using a single batch of inputs on the TitanX GPU, which means that our method is performed using a 5-frame batch. The PP means the requirement of post-processing

Methods	F1(%)	Accuracy(%)	FP(%)	FN(%)	FPS	Para	PP
LaneNet [11]	72.1	85.8	12.2	20.7	36	1.48	Y
SCNN [12]	49.1	90.7	12.8	11.0	16	20.72	Y
ENet-SAD [2]	75.5	88.6	17.0	15.2	14	0.98	Y
UFSA [14]	31.0	85.2	11.5	21.5	-	-	Y
LSTR [7]	70.3	88.4	16.3	14.8	**48**	**0.77**	N
MMA [23]	83.9	91.0	11.1	10.5	9	57.91	Y
Ours	**84.6**	**91.5**	**8.2**	**9.5**	46	3.87	N

× faster speed. The large improvement on the challenging dataset fully demonstrates the effectiveness of our proposed method.

Besides quantitative comparisons, we show our results qualitatively through visualization in Fig. 4. We visualize the performance of our method in various challenging scenarios, such as shadows, bright light, darkness, congestion and foggy weather. Based on the visualization results, we notice that our method can adequately capture global features and detect lanes well even when the visual information of lanes is very scarce.

TuSimple The results on TuSimple are shown as Table 2. It can be noted that our method achieves comparable results compared with other methods. The visualization results are shown in the Fig 4. We can notice that our method performs well, even can detect lanes not labeled in the ground truth(shown in row 3).

Compared to LSTR, our method does not improve significantly. We guess that the root cause may lie in the limitations of the dataset itself. Firstly, as mentioned in [5, 16, 6], it is a relatively simple dataset because the highway scenario is easier than street scenes and not congested. Thus, it's hard to expose the model's advantages in realistic complex scenes and has more saturated results. Besides, the road scene with no congestion and stable illumination results in the high similarity of multi-frame input data, which makes the use of multi-frame input meaningless. Combined with the results of the VIL-100 dataset, the results on the TuSimple dataset confirm the effectiveness of multi-frame input for global content information in the complex real driving environment.

4.5 Ablation Study

Because the TuSimple dataset cannot clearly reflect the difference among methods, we conduct ablation experiments on the VIL-100 dataset.

Effectiveness of the Multi-frame Mechanism To verify the effectiveness of multiple frames, we improve the Transformer in LSTR to adapt to multi-frame

Table 2: Quantitative comparisons on TuSimple dataset. * means our re-implemented results on TuSimple

Method	Accuracy(%)	FP(%)	FN(%)
SCNN [12]	96.53	6.17	1.80
Enet-SAD [2]	96.64	4.67	5.18
UFSA [14]	95.82	19.05	3.92
Line-CNN [6]	**96.87**	4.42	**1.97**
PINet [4]	96.75	**3.10**	2.50
LaneATT [16]	95.57	3.56	3.10
PolyLaneNet [17]	93.36	9.42	9.33
LSTR* [7]	96.03	3.26	3.44
Ours	96.17	3.50	3.38

feature maps. Specifically, at the core self-attention part, the attention between pixels in a single frame is replaced by the relationship modeling among all pixels in multiple frames, which is similar to the video instance segmentation method [18]. In addition to this, single-frame input experiments are also conducted on our method.

As shown in Table 3, compared with original LSTR(method A), method B with multi-frame input outperforms it by 0.43% on *F1*, 0.89% on *Accuracy*, 3.49% on *FP*. On our model, the multi-frame continuous input brings an improvement of 5.49% on *F1*, 2.33% improvement on *Accuracy*, a reduction of 3.43% on *FP*, and a reduction of 4% on *FN*.

To exclude the influence of complexity, we experiment on our method with 5 same frames as input and the results are shown as method D. We found that compared to single frame input, all metrics are worse. This result excludes the influence of complexity and fully demonstrates the effectiveness of multi-frame input.

Effectiveness of Our Model Based on method B, we generate the method C, replacing the Transformer encoder part with our MHVA module. As shown in Table 3, we find that method C outperforms B by 9.81% on *F1*, 1.1% on *Accuracy*, 3.53% on *FP*, and 3.97% on *FN*. It shows that the strategic information passing is more effective than blind full-pixel information interaction, which indicates the rationality and effectiveness of our MHVA design.

The difference between method C and our method is the decode part. We replace the Transformer decoder with VT modules. Compared with method C, we can see that the addition of the VT module in method F has largely improved the values of metrics. The value of *F1* increases 4.1%, and *Accuracy* increases 1.1%, while *FN* decreases 2.07%, and *FP* decreases 1.9%.

Besides, based on single frame input, the comparison of method D and method A can prove the superiority of our model. We can notice that our model can bring an improvement of 4.05% on *F1*, 0.81% improvement on *Accuracy*, a reduction of 4.5% on *FP*, and a reduction of 0.76% on *FN*.

Table 3: Quantitative results of different models with two kinds of input

methods	model	input	F1(%)	accuracy(%)	FP(%)	FN(%)
A	LSTR	single	70.30	88.40	16.30	14.30
B		5 frames	70.73	89.31	13.93	14.37
C	+MHVA	5 frames	80.54	90.43	10.44	11.44
D	our	single	79.51	89.21	11.80	13.54
E		5 same	76.85	89.05	14.17	14.34
F		5 frames	84.59	91.54	8.37	9.54

Table 4: Quantitative results of our method with different sampling numbers

Frames	F1(%)	Accuracy(%)	FP(%)	FN(%)
3 (frames)	83.68	90.68	9.47	10.82
5 (frames)	**84.59**	91.54	**8.37**	9.54
7 (frames)	84.34	**91.80**	8.41	**9.34**

Selection of Sampling Number To test the influence of the choice of different sampling numbers, we experiment with different frame sampling numbers 3, 5 and 7. The results are shown in Table 4. It can be found that the result of 5 frames outperforms that of 3 frames significantly, and is comparable with the result of 7 frames under smaller GPU memory and less inference time. Therefore, we choose 5 frames as the final sampling number of input data.

5 Conclusions

In conclusion, we propose a novel lane detection Transformer using multiple frames as input. Based on curve fitting, it can detect lanes directly and efficiently. Besides, the customized MHVA can capture more global information in two directions, and the VT modules are very effectual in improving detection result. Our method can achieve real-time results despite the use of multi-frame information, which enables the deployment in practical applications.

Bibliography

[1] Ghafoorian, M., Nugteren, C., Baka, N., Booij, O., Hofmann, M.: El-gan: Embedding loss driven generative adversarial networks for lane detection. In: proceedings of the european conference on computer vision (ECCV) Workshops. pp. 0–0 (2018)

[2] Hou, Y., Ma, Z., Liu, C., Loy, C.C.: Learning lightweight lane detection cnns by self attention distillation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1013–1021 (2019)

[3] Jung, S., Choi, S., Khan, M.A., Choo, J.: Towards lightweight lane detection by optimizing spatial embedding. arXiv preprint arXiv:2008.08311 (2020)

[4] 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)

[5] Lee, M., Lee, J., Lee, D., Kim, W., Hwang, S., Lee, S.: Robust lane detection via expanded self attention. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 533–542 (2022)

[6] 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)

[7] 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)

[8] Liu, Y.B., Zeng, M., Meng, Q.H.: Heatmap-based vanishing point boosts lane detection (2020)

[9] Lo, S.Y., Hang, H.M., Chan, S.W., Lin, J.J.: Multi-class lane semantic segmentation using efficient convolutional networks. In: 2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP). pp. 1–6. IEEE (2019)

[10] Mamidala, R.S., Uthkota, U., Shankar, M.B., Antony, A.J., Narasimhadhan, A.: Dynamic approach for lane detection using google street view and cnn. In: TENCON 2019-2019 IEEE Region 10 Conference (TENCON). pp. 2454–2459. IEEE (2019)

[11] 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)

[12] 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)

[13] Pizzati, F., Allodi, M., Barrera, A., García, F.: Lane detection and classification using cascaded cnns. In: International Conference on Computer Aided Systems Theory. pp. 95–103. Springer (2019)

[14] 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)
[15] Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems **28** (2015)
[16] 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)
[17] 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)
[18] Wang, Y., Xu, Z., Wang, X., Shen, C., Cheng, B., Shen, H., Xia, H.: End-to-end video instance segmentation with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8741–8750 (2021)
[19] Wu, B., Xu, C., Dai, X., Wan, A., Zhang, P., Yan, Z., Tomizuka, M., Gonzalez, J., Keutzer, K., Vajda, P.: Visual transformers: Token-based image representation and processing for computer vision. arXiv preprint arXiv:2006.03677 (2020)
[20] 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: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 1006–1007 (2020)
[21] Zhang, H., Gu, Y., Wang, X., Wang, M., Pan, J.: Sololanenet: Instance segmentation-based lane detection method using locations. In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). pp. 2725–2731. IEEE (2021)
[22] Zhang, J., Deng, T., Yan, F., Liu, W.: Lane detection model based on spatio-temporal network with double convolutional gated recurrent units. IEEE Transactions on Intelligent Transportation Systems p. 1–13 (2021). https://doi.org/10.1109/tits.2021.3060258, http://dx.doi.org/10.1109/TITS.2021.3060258
[23] Zhang, Y., Zhu, L., Feng, W., Fu, H., Wang, M., Li, Q., Li, C., Wang, S.: Vil-100: A new dataset and a baseline model for video instance lane detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 15681–15690 (2021)
[24] Zheng, T., Fang, H., Zhang, Y., Tang, W., Yang, Z., Liu, H., Cai, D.: Resa: Recurrent feature-shift aggregator for lane detection. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 3547–3554 (2021)
[25] Zou, Q., Jiang, H., Dai, Q., Yue, Y., Chen, L., Wang, Q.: Robust lane detection from continuous driving scenes using deep neural networks. IEEE transactions on vehicular technology **69**(1), 41–54 (2019)