PersFormer: 3D Lane Detection via Perspective Transformer and the OpenLane Benchmark Appendix

Li Chen[1*†], Chonghao Sima[1*], Yang Li[1*], Zehan Zheng[1], Jiajie Xu[1], Xiangwei Geng[1], Hongyang Li[1,2†], Conghui He[1], Jianping Shi[3], Yu Qiao[1], and Junchi Yan[1,2]

[1] Shanghai AI Laboratory [2] Shanghai Jiao Tong University
[3] SenseTime Research
{lichen,simachonghao,liyang,lihongyang}@pjlab.org.cn
yanjunchi@sjtu.edu.cn

A More Related Work

A.1 Lane Detection Benchmarks

For example, [12,9,32] annotate lanes and lane markings in pixel-level so they are best suitable for semantic segmentation task. [27,3] collect data on highways with light traffic only, which is not challenging and has a large gap between the evaluation and real-world performance for up-to-date algorithms. [19,30] consider more scenarios under different weather and traffic conditions; however, no-segment character limits their applicability for future applications, such as lane tracking or temporal lane detection. The recently released VIL-1000 [33] is specifically designed for video instance lane detection, and yet it does not provide tracking ID annotation across the segments. At the same time of our proposing OpenLane dataset, there's another large-scale realistic 3D lane dataset, named ONCE-3DLanes [31], that annotates lane layout in 3D space. The difference between OpenLane and ONCE-3DLanes falls into three aspects. First is the dataset statistics. The number of frames contained is quite the same, where OpenLane has 200K in total and ONCE-3DLanes has 211K. The annotation quality differentiates a lot, as OpenLane has more than 25% of frames with more than 6 lanes, while ONCE-3DLanes only has less than 10% of frames under the same setting. Second is the problem setting. OpenLane provides camera extrinsics as Waymo Open Dataset, while ONCE-3DLanes lacks of this information. Meanwhile, OpenLane provides segments annotation as scene tags, where ONCE-3DLanes doesn't. This could be used in video task and expand the potential usage of OpenLane. Third is the diversity of lane annotation. In OpenLane, the lane annotation not only contains the 3D position of such a lane, but also several attributes and tracking id. In ONCE-3DLanes, only the 3D position information is provided. Due to the difficulty of collecting 3D information for lanes, current 3D lane detection algorithms mainly focus on synthetic data [7]. It is small-scale and exists the domain gap between simulation and realistic scenarios.

* Equal contribution. † Correspondence author.

A.2 2D Lane Detection

Early lane detection approaches rely on traditional computer vision techniques, such as filtering [2,14], clustering [28], *etc.* With the advent of deep learning, CNN-based methods significantly outperform hand-crafted algorithms. A typical way is to treat lane detection as a semantic segmentation problem [12,19,18,8,1]. Binary segmentation [18] needs post-clustering process for lane instance discrimination, while multi-class segmentation [12,19,8] usually limits the maximum detection results in one frame. Moreover, the pixel-wise classification takes large computation resources. To overcome this, several work propose lightweight yet effective grid based [21,16,10,22] or anchor based [5,13,30,23,25] methods. The grid-based approach detects lanes in a row-wise way, whose resolution is much lower than the segmentation map. The model outputs the probability for each cell if it belongs to a lane, and a vertical post-clustering process is still needed to generate the lane instances. Anchor-based approaches adopt the idea from classical object detection, focusing on optimizing the offsets from predefined line anchors. In this circumstance, how to define anchors is a critical problem. Chen *et al.* [5] adopts vertical anchors, which cause great difficulty for curving lane prediction. Some work [13,25,23] design anchors as a slender tilt shape, while the huge amount of different anchors to improve the detection accuracy would influence the computational efficiency. Nevertheless, considering their incredible performance on public datasets, we adopt the anchor-based formulation and carefully re-design anchors to achieve both high accuracy and efficiency.

B Algorithm

We summarize the details of PersFormer here. We introduce the backbone, overall structure and the unified anchor design. Later we break down the loss function into pieces.

B.1 Backbone

The backbone module is slightly different from previous work [6,7], as we need to consider 2D/3D branches together. We use EfficientNet [26] as our backbone, and extract a specific layer as our following module's input. Later we provide two designs, using FPN [15] or not. After using several convolution layers, the backbone module outputs 4 different scaled front-view feature maps. Their resolutions are 180×240, 90×120, 45×60, 22×30. Each front-view feature map is then transformed to BEV-space feature map with the help of Perspective Transformer, resulting in 4 BEV feature maps.

B.2 Anchor Details

In this section, we present details of our anchor design, including angles, numbers of anchors and how we associate ground truth lanes with anchors in 2D and 3D.

As introduced in the main body of the paper, we first set anchors in BEV space. Following Gen-LaneNet [7], the starting positions X^i_{bev} are evenly placed along x-axis with the spacing of 8 pixels. However, we differentiate it from the incline angle φ. Gen-LaneNet sets straight-forward (parallel to y-axis) only, which makes it hard to predict lanes with large curvatures or perpendicular lanes. Towards this problem, we put 7 anchors at each X^i_{bev} with different angles, *i.e.*, $\varphi \in \{\pi/2, \arctan(\pm 0.5), \arctan(\pm 1), \arctan(\pm 2)\}$. Note that the angles are in terms of grid coordinates, which is not equal to the absolute values when grids are not square. Moreover, we project all the BEV anchors to image space with average camera height and pitch angle of the dataset, leading to corresponding 2D anchors.

The association between ground truth lanes and anchors is based on the average distance similar to the loss calculation process, instead of assigning the closest anchor at Y_{ref} to ground truths as [6,7]. The Y_{ref} is set very close to ego-vehicle, *i.e.*, $5m$ in Gen-LaneNet, which makes it better predict lanes in close area while having unsatisfactory performance in the far distance. In our experiments, we assign the anchor with minimum *edit distance* to ground truth lanes in both 2D and 3D tasks. The distance is calculated at fixed y positions: $(5, 10, 15, 20, 30, 40, 50, 60, 80, 100)$ for 3D anchors, and 72 equally sampled heights for 2D anchors.

B.3 Loss Function

We give the details of loss function here. As introduced in the main body of the paper, given the pre-defined y value of the N_d samples along y-axis, the 3D detection head outputs a set of points for each anchor i as following:

$$(\mathbf{x}^i, \mathbf{z}^i, \mathbf{vis}^i_{\text{bev}}) = \{(x^{(i,k)}, z^{(i,k)}, \text{vis}^{(i,k)}_{\text{bev}})\}_{k=1}^{N_d} \tag{1}$$

The y values are $(5, 10, 15, 20, 30, 40, 50, 60, 80, 100)$ in the BEV space, and the size of the BEV space is $20m \times 100m$. Similar to 3D setting, given the pre-defined v value of the N_d samples along v-axis in front view, the 2D prediction is:

$$(\mathbf{u}^i, \mathbf{vis}^i_{\text{uv}}) = \{(u^{(i,k)}, \text{vis}^{(i,k)}_{\text{uv}})\}_{k=1}^{N_d} \tag{2}$$

The loss is a combination of the 2D lane detection, 3D lane detection and intermediate segmentation with learnable weights (α, β, γ) accordingly:

$$\mathcal{L} = \sum_i \alpha \mathcal{L}_{\text{2D}}(c^i_{\text{2D}}, \mathbf{u}^i, \mathbf{vis}^i_{\text{fv}}) + \beta \mathcal{L}_{\text{ 3D}}(c^i_{\text{3D}}, \mathbf{x}^i, \mathbf{z}^i, \mathbf{vis}^i_{\text{bev}}) + \gamma \mathcal{L}_{\text{seg}}(S_{\text{pred}}), \tag{3}$$

where $c^i_{(\cdot)}$ is the predicted lane category in 2D and 3D domain respectively. For $\mathcal{L}_{\text{ 3D}}$, it consists of classification loss, regression loss and visibility loss. The classification loss is a cross-entropy loss, which is as follow:

$$\mathcal{L}_{\text{3D-cls}} = \mathcal{L}_{CE}(c^i_{\text{3D-pred}}, c^i_{\text{3D-gt}}) \tag{4}$$

The regression loss is a L_1 loss, which is as follow:

$$\mathcal{L}_{\text{3D-reg}} = \mathcal{L}_{L_1}(\{\mathbf{x}^i, \mathbf{z}^i\}_{\text{pred}}, \{\mathbf{x}^i, \mathbf{z}^i\}_{\text{gt}}) \tag{5}$$

The visibility loss is a binary cross-entropy loss, which is as follow:

$$\mathcal{L}_{\text{3D-vis}} = \mathcal{L}_{BCE}(\mathbf{vis}^i_{\text{pred}}, \mathbf{vis}^i_{\text{gt}}) \tag{6}$$

The 2D loss functions are similar to the 3D ones, except they are in 2D form:

$$\begin{aligned} \mathcal{L}_{\text{2D-cls}} &= \mathcal{L}_{CE}(c^i_{\text{2D-pred}}, c^i_{\text{2D-gt}}) \\ \mathcal{L}_{\text{2D-reg}} &= \mathcal{L}_{L_1}(\{\mathbf{u}^i\}_{\text{pred}}, \{\mathbf{u}^i\}_{\text{gt}}) \\ \mathcal{L}_{\text{2D-vis}} &= \mathcal{L}_{BCE}(\mathbf{vis}^i_{\text{pred}}, \mathbf{vis}^i_{\text{gt}}) \end{aligned} \tag{7}$$

The segmentation loss is a binary cross-entropy loss as well, which is as follow:

$$\mathcal{L}_{\text{seg}} = \mathcal{L}_{BCE}(S_{\text{pred}}, S_{\text{gt}}) \tag{8}$$

C Details on OpenLane Benchmark

In this section, we present more details on dataset statics, our annotation criterion, visualization examples, algorithms we adopted when generating the dataset.

C.1 Dataset Statistics

OpenLane has 1,150 segments with train/validation/test splits of 798/202/150, respectively. Since the test sets are kept for its online leaderboard evaluation, we annotate the other 1,000 segments, *i.e.*, 200K frames at a frequency of 10 FPS, and keep the original train/validation partition for fair comparison with other tasks, such as object detection.

We compute the statistics in OpenLane and visualize them. The overall number of segments with different scene tags is given in Tab. 1. It implies great diversity in data collection and raises higher requirements on the robustness of algorithms. The weather distribution is visually presented in Fig. 1. It shows the benchmark covers various weather conditions and well holds the consistency in the train/validation split. The distribution of the number of lanes in each frame is shown in Fig. 2. About 25% frames of OpenLane have more than 6 lanes, which exceeds the maximum number in most lane datasets. Fig. 3 shows the distribution of lane categories. Single white solid and dash lanes, double yellow solid lanes take up almost 90% of the total lanes. This is imbalanced and yet it falls into a long-tail distribution problem, which is common in realistic scenarios. Fig. 4 presents the distribution of altitude difference per frame. Only around 20% frames are relatively flat with absolute height variation less than $0.5m$, whereas the difference is more than $1m$ in over 50% of OpenLane. This data further demonstrates the necessity of 3D lane detection. The above statistics and examples below demonstrate that OpenLane is the most challenging one compared to existing lane detection datasets.

Table 1. Statics of scenario tags. Scene tags are annotated in terms of segments

Tags		Train	Val.	All
Weather	Clear	515	145	660
	Partly cloud	131	28	159
	Overcast	33	8	41
	Rainy	107	18	125
	Foggy	12	3	15
Scene	Residential	270	69	339
	Urban	234	56	290
	Suburbs	259	64	323
	Highway	30	6	36
	Parking lot	5	7	12
Hours	Daytime	653	167	820
	Night	88	22	110
	Dawn/Dusk	57	13	70

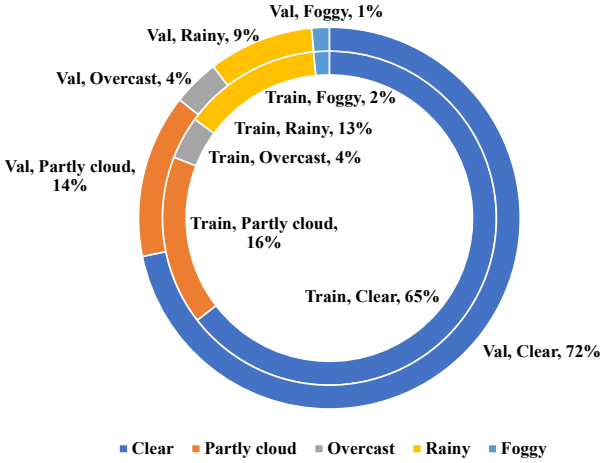

Fig. 1. Distribution of weather tags in training and validation sets. The data is collected under different weathers and split into training and validation with great balance

C.2 Annotation Criterion

We aim at introducing how we annotate lanes, scene tags and CIPO levels in this section. Details such as data structures, folder hierarchy will be provided in the dataset releasing page in the future.

Lanes. Our principle for the 2D lane detection task is to find all visible lanes inside left and right road edges. Following this philosophy, we carefully annotate lanes in each frame. However, due to the complexity of scenarios, there exist some special cases we seek to illustrate here. (1) Lanes are often occluded by objects or invisible because of abrasion but they are still valuable for the real application. Thus we annotate lanes if parts of them are visible, meaning lanes with one side being occluded are extended or lanes with invisible intermediate

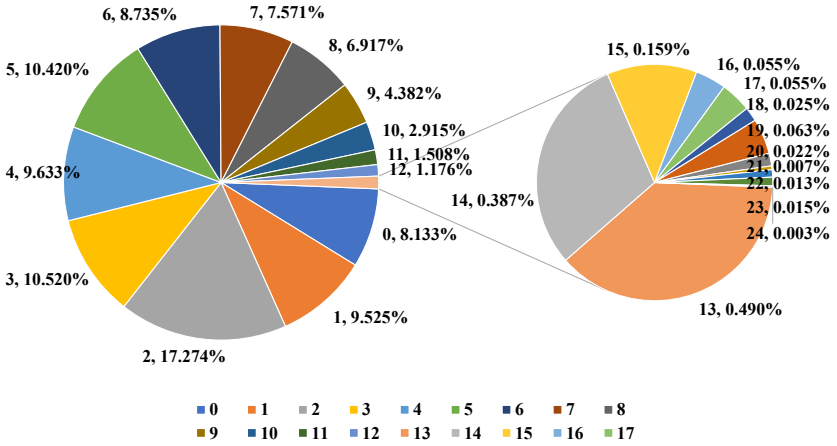

Fig. 2. Distribution of lane numbers per frame. The maximum number is 24, and 25% frames have more than 6 lane

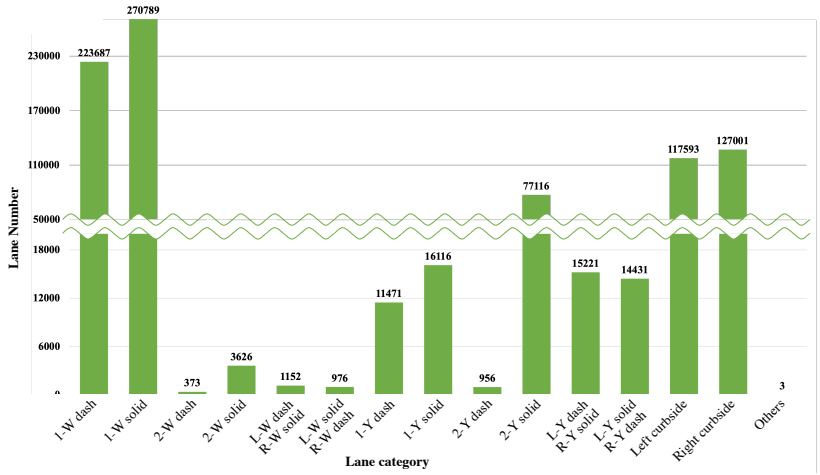

Fig. 3. Distribution of the lane category. Here we abbreviate single in *1*, double in *2*, white in *W*, yellow in *Y*, left in *L*, and right in *R*. Thus *1-W dash* means the category of single white dash lanes

parts are completed according to the context, as shown in Fig. 5. (2) It is very common that the number of lanes changes, especially when lanes have complex topologies such as fork lanes in merge and split cases. Traditional lane datasets usually omit these scenarios for simplicity, while we keep them all and further choose them out of the whole dataset for evaluation. Fork lanes are annotated as separate lanes with a common starting point (split) or ending point (merge) - two close adjacent lanes are desired for the lane detection methods. (3) We further

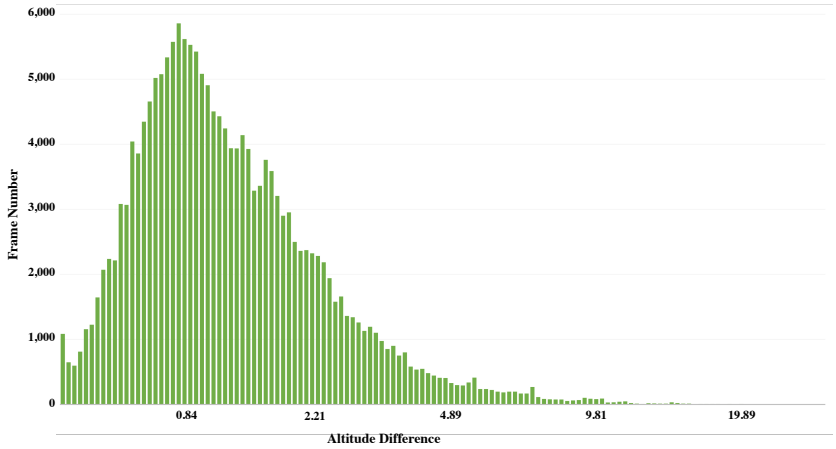

Fig. 4. Altitude difference per frame. Note the x-axis is approximately in a log scale and its unit is m

annotate each lane as one of the 14 lane categories, *i.e.*, single white dash, single white solid, double white dash, double white solid, double white dash solid (left white dash with right white solid), double white solid dash (left white solid with right white dash), single yellow dash, single yellow solid, double yellow dash, double yellow solid, double yellow dash solid (left yellow dash with right yellow solid), double yellow solid dash (left yellow solid with right yellow dash), left curbside, right curbside. Note that traffic bollards are considered as curbsides as well if they are not temporally placed. (4) Different from all the other lane datasets, we annotate a tracking ID for each lane which is unique across the whole segment. We believe this could be helpful for video lane detection or lane tracking tasks. We also assign a number in 1-4 to the most important 4 lanes based on their relative position to the ego-vehicle. Basically, the left-left lane is 1, the left lane is 2, the right lane is 3, and the right-right lane is 4.

All valid 2D ground truths are transformed to 3D annotations by the generation method in Sec. 4.2 of the main body (Generation of High-quality Annotation), except those without LiDAR points scanning through. Thus the criterion above applies to 3D lanes as well.

Scene tags. We label each segment with 3 scene tags, *i.e.*, weather, scene and hours. We hope these labels can help researchers to investigate the robustness of their models under various scenarios. The statics are shown in Tab. 1. Specifically, the dataset covers 5 different kinds of weather, clear, partly cloud, overcast, rainy and foggy. Note that we classify the video as partly cloud or foggy when there are clouds or fog in the sky respectively, otherwise it will be categorized as overcast. The scene, or the location, includes 5 categories, *i.e.*, residential, urban, suburbs, highway and parking lot. And the hours are divided into 3 parts: daytime, night, dawn/dusk.

Closest-in-path object (CIPO). CIPO is usually defined as the closest object in ego lane, which refers to a single vehicle only. However, there are cases that vehicles on left/right lanes are intended to cut in which are crucial as well, or there may not be any qualified vehicles in ego lane. To cover the complex scenarios, we categorize objects, mainly including vehicles, pedestrians and cyclists, into 4 different CIPO levels. (1) The most important one, which is closest to ego vehicle within the required reaction distance and has over 50% part of it in the ego lane. Level 1 contains one object at most. (2) Objects are annotated as Level 2 when their bodies interact with the real or virtual lines of ego lane. They are typically in the process of cut-in or cut-out, which hugely influences ego-vehicle decision-making. (3) We consider objects mainly within the reaction distance or drivable area, or those in left/ego/right lanes more specifically. Thus we annotate Level 3 with objects in the above area and having occlusion rate less than 50%. Note that vehicles in the opposite direction can be in this CIPO level as well. (4) The remainings are labeled as Level 4, which means they are almost unlikely to impact the future path at this moment. They are mainly objects in lanes with far distance, objects out of drivable area, or parked vehicles in our dataset. Examples are provided in Fig. 6.

C.3 3D lane Generation

Fig. 7 shows the intermediate results of the generation process of 3D lane labels. However, the above process could have a few problems in some cases, especially in the last step, *i.e.*, smoothing and fitting. Multiple filtering and fitting algorithms are adopted to realize it, while all of them require a set of sorted points. Due to the large curvature, the one-to-one mapping probably does not stand either in x or y direction, thus we could not sort the points directly. Towards this problem, for each image with this circumstance, we simply find an angle to rotate the whole points set, do the filtering and fitting process in the temporary coordinate and rotate back in the end. This method is illustrated in Fig. 8.

D Experiments

D.1 Evaluation Metrics.

For both 3D lane datasets, we follow the evaluation metric designed by GenLaneNet [7], with small modifications[1] and additional category accuracy on OpenLane dataset. The matching between prediction and ground truth is built upon *edit distance*, where one predicted lane is considered to be a true positive only if 75% of its covered y-positions have a point-wise distance less than the max-allowed distance ($1.5m$). Then, with the percentage of matched ground-truth lanes as recall and the percentage of matched prediction lanes as precision, we use F-score to report the regression performance of such a model. Since OpenLane dataset has category information per lane, we present the accuracy

[1] Please see in OpenLane page: https://github.com/OpenPerceptionX/OpenLane.

Fig. 5. Visualization example of lane annotation in OpenLane dataset

upon the matched lanes to show classification performance. We only report the accuracy of PersFormer on OpenLane dataset, as other 3D methods do not support classification task. For the 2D task, the classical metric in CULane [19] is adopted.

D.2 Implementation Details

To fairly compare with other methods [7,6,17], we retain many model settings of image resolution and BEV scale. We resize the original image to 360×480 as model input, project it to BEV space with a resolution of 208×108. We use PyTorch [20] to implement the model. The batch size is set to 8; the number of training epochs is set to 100. We re-implement 3D-LaneNet and Gen-LaneNet on OpenLane dataset for a fair comparison. Following previous experience on training vision transformer [4,34,29], we use Adam optimizer [11] with base learning rate of 2×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and weight decay of 10^{-4}. All of these mod-

Fig. 6. Visualization example of CIPO and Scene tags annotation in OpenLane dataset

els are trained on 8 NVIDIA Tesla V100 GPUs. More details about environment setup can be referred to our GitHub repository once accepted.

D.3 More Experimental Results

In this section, we present more experimental results, mainly in 3D comparison on ONCE-3DLanes, additional ablations and more qualitative examples.

3D Comparisons on ONCE-3DLanes. We provide additional experimental results on ONCE-3DLanes dataset [31], as it's another real-world 3D lane dataset concurrently presented. ONCE-3DLanes also uses F-Score as the evaluation metric, and more details can be found in their repo ONCE-3DLanes. In Tab. 2, PersFormer gets the highest F-Score on the validation set, outperforming its proposed method SALAD [31] over **10%**. One thing worth noticing is that ONCE-3DLanes does not provide camera extrinsics, therefore PersFormer predefine a set of extrinsic parameters to fit the model setting. The camera height

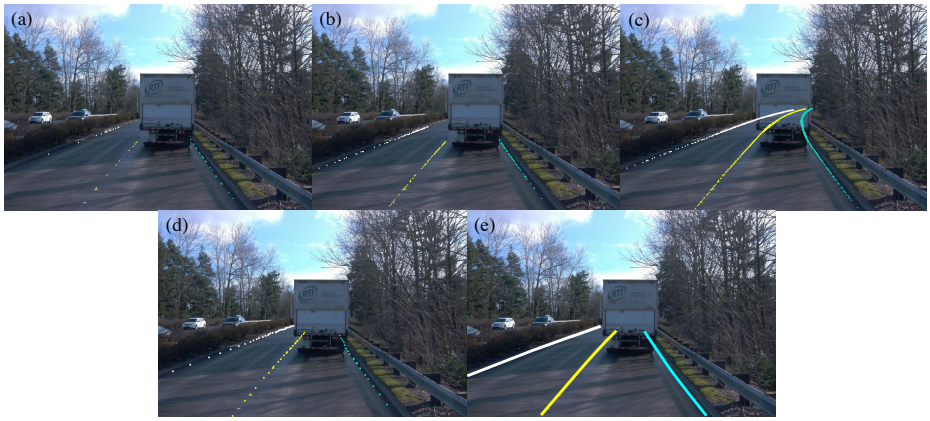

Fig. 7. 3D lane generation pipeline. (a) Original point clouds inside a certain threshold of 2D lane annotations are reserved, which is relatively sparse; (b) Positions of points on the 2D annotation are interpolated to get a dense point set; (c) 3D lane points in the same segment are spliced into long, high-density lanes; (d) We remove those too far as they are invisible, while reasonable extensions are desired; (e) A smooth and fitting process is applied to get the final 3D lane annotation

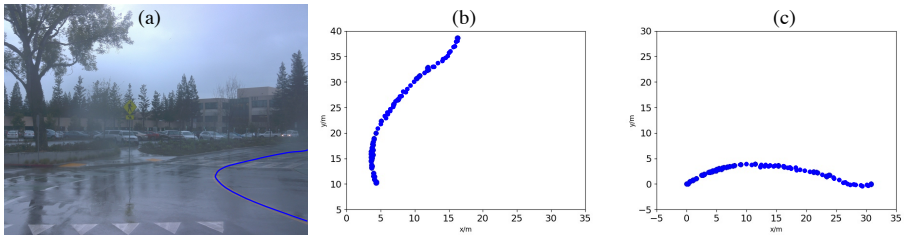

Fig. 8. Illustration of 3D lane generation problem with large curvatures. (a) The original image and the 2D lane; (b) Unsorted 3D points set of the lane in (a), which a filtering algorithm is not applicable directly; (c) A simple translation and rotation can result in a one-to-one mapping of x and y

Table 2. New results on the new benchmark (CVPR22) ONCE-3DLanes [31]. * denotes results from the paper [31]

Method	F-Score(%)	Precision(%)	Recall(%)	CD error(m)
3D-LaneNet* [6]	44.73	61.46	35.16	0.127
Gen-LaneNet* [7]	45.59	63.95	35.42	0.121
SALAD* [31]	64.07	75.90	55.42	0.098
PersFormer (ours)	**74.33**	**80.30**	**69.18**	**0.074**

is set to be $1.5m$ and pitch to be 0.5. This does not affect the evaluation results since it is just to fit the IPM process in PersFormer.

Table 3. Ablative Study on PersFormer Design. IPM prior plays a vital role in guiding the generation of BEV feature compared to naive one-to-one mapping and learned reference-target mapping. Using MSDeformAttn from Deformable DETR [34] to map multi-scale front-view feature to multi-scale BEV feature is competitive, and the self-attention module of BEV query is important in Transformer-style structure

Exp.	Naive 1-1	Learned	Multi-to-Multi	Self Attn.	IPM Prior	3D F-Score
1	✓					36.15
2		✓				13.45
3			✓			51.35
4				✓		47.18
5					✓	52.68

Ablations. We provide an additional ablative study on the structure of the feature transformation module on a subset of OpenLane (∼300 segments) in Tab. 3. We argue that the IPM-based cross attention is a necessity in PersFormer, as we compare it with two initial designs, naive one-to-one mapping and the learned mapping. The naive one-to-one mapping simply scales every location in the BEV space to the corresponding location in the front view space, not considering camera parameters (Exp.1). A more "aggressive" way to simulate the mapping is directly learning from the front view feature with several fully-connected layers (Exp.2). Neither of them could catch up with the performance of IPM-based mapping, indicating the importance of such a prior in generating BEV feature. We further attempt to adopt Multi-scale Deformable Attention from [34] to implement a several-for-one feature mapping from multi-scale front view feature to multi-scale BEV feature (Exp.3), just like Deformable DETR. The result slightly falls behind our final design (Exp.5), probably due to the influence of tuning of hyper-parameters and the impact of the small-scale feature on the large-scale feature. Finally, we try to remove the classical self attention module in ordinary Transformer design (Exp.4), showing that the self attention module is all there for a reason in Transformer-style structure.

Visualization. We provide qualitative results compared with SOTA 3D lane detection methods in different evaluation scenarios on OpenLane dataset in Fig. 9,10. Results on Apollo 3D synthetic dataset are shown in Fig. 11. We can observe that PersFormer could achieve higher accuracy and capture more lanes to reconstruct the scenes on both datasets.

E License of Assets

OpenLane dataset is based on the Waymo Open Dataset [24] and therefore we distribute the data under Creative Commons Attribution-NonCommercial-ShareAlike license and Waymo Dataset License Agreement for Non-Commercial Use (August 2019). You are free to share and adapt the data, but have to give

appropriate credit and may not use the work for commercial purposes. All code of PersFormer and OpenLane toolkit is under Apache License 2.0.

The pretrained ResNet model weights are under the MIT license. We integrate part of the code of Deformable-DETR [34] and Gen-LaneNet [7] which are under Apache License 2.0. We also use part of the code of LaneATT [25] which is under the MIT license.

F Outlook

As OpenLane is built upon Waymo Open Dataset [24], a road-object joint detection framework is possible in the future. Moreover, BEV is the necessity in the future of autonomous driving, and how to design a better BEV representation remains to be explored. The proposed PersFormer may also be adapted to new tasks.

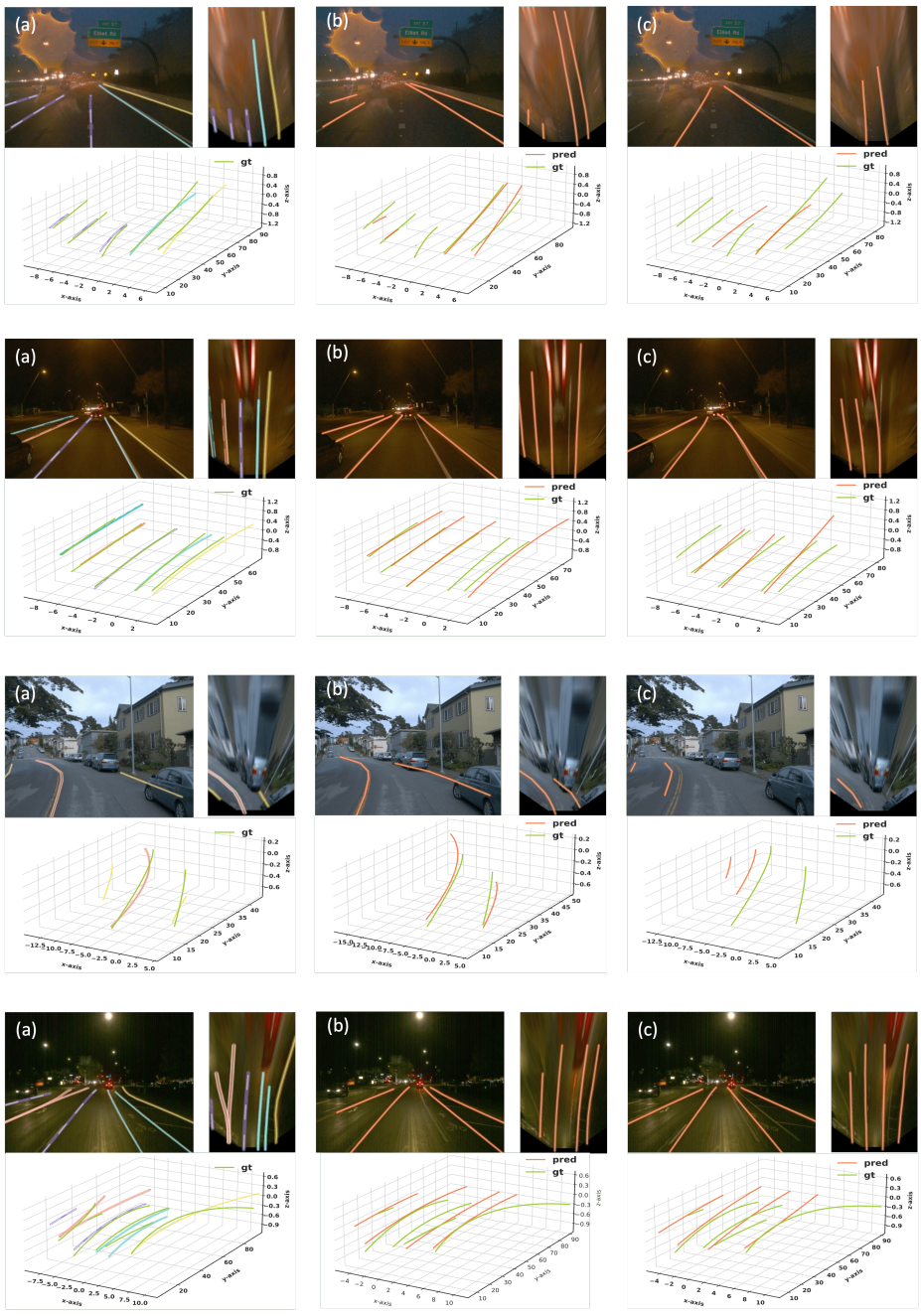

Fig. 9. Qualitative results of PersFormer(a), 3D-LaneNet(b) [6], and Gen-LaneNet(c) [7] on OpenLane. Night case and Up&Down case

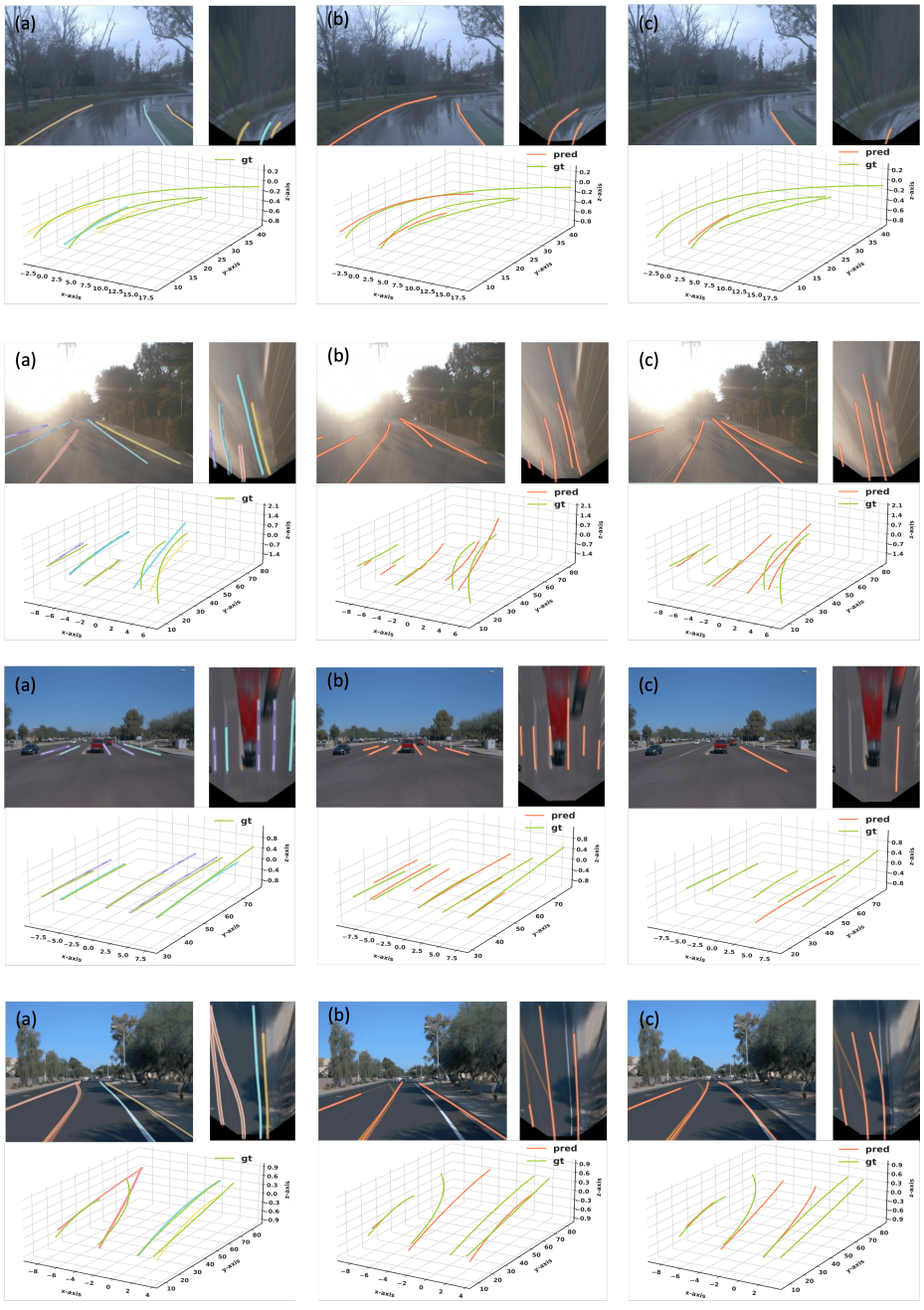

Fig. 10. Qualitative results of PersFormer(a), 3D-LaneNet(b) [6], and Gen-LaneNet(c) [7] on OpenLane. Extreme weather case, Intersection case and Merge&Split case

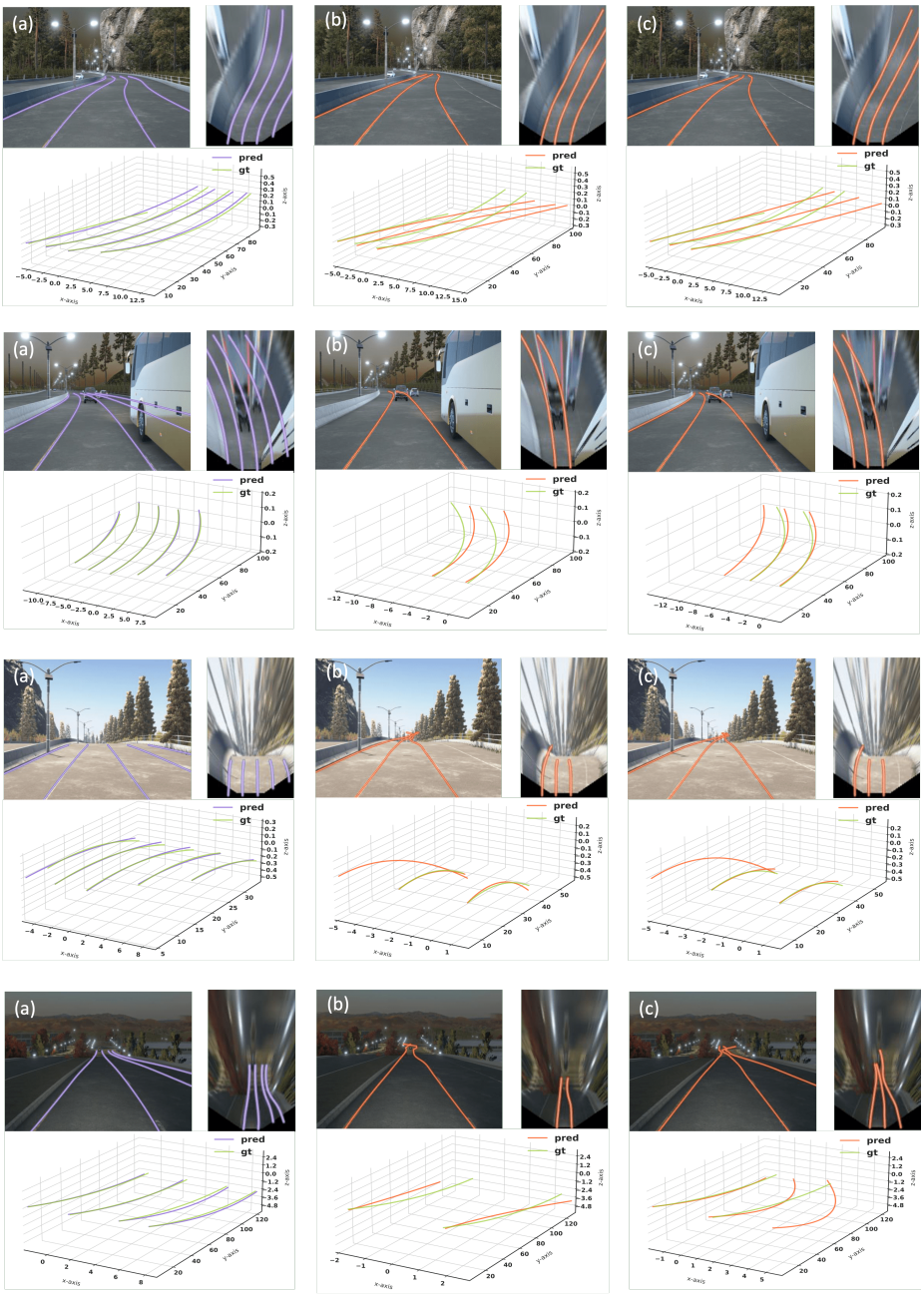

Fig. 11. Qualitative results of PersFormer(a), 3D-LaneNet(b) [6], and Gen-LaneNet(c) [7] on Apollo. Curve case and Up&Down case

References

1. Abualsaud, H., Liu, S., Lu, D.B., Situ, K., Rangesh, A., Trivedi, M.M.: Laneaf: Robust multi-lane detection with affinity fields. RA-L (2021) 2
2. Aly, M.: Real time detection of lane markers in urban streets. In: IV (2008) 2
3. Behrendt, K., Soussan, R.: Unsupervised labeled lane markers using maps. In: ICCV (2019) 1
4. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV (2020) 9
5. Chen, Z., Liu, Q., Lian, C.: Pointlanenet: Efficient end-to-end cnns for accurate real-time lane detection. In: IV (2019) 2
6. Garnett, N., Cohen, R., Pe'er, T., Lahav, R., Levi, D.: 3d-lanenet: End-to-end 3d multiple lane detection. In: ICCV (2019) 2, 3, 9, 11, 14, 15, 16
7. Guo, Y., Chen, G., Zhao, P., Zhang, W., Miao, J., Wang, J., Choe, T.E.: Gen-lanenet: A generalized and scalable approach for 3d lane detection. In: ECCV (2020) 1, 2, 3, 8, 9, 11, 13, 14, 15, 16
8. Hou, Y., Ma, Z., Liu, C., Loy, C.C.: Learning lightweight lane detection cnns by self attention distillation. In: ICCV (2019) 2
9. Huang, X., Wang, P., Cheng, X., Zhou, D., Geng, Q., Yang, R.: The apolloscape open dataset for autonomous driving and its application. TPAMI (2019) 1
10. Jayasinghe, O., Anhettigama, D., Hemachandra, S., Kariyawasam, S., Rodrigo, R., Jayasekara, P.: Swiftlane: Towards fast and efficient lane detection. In: ICMLA (2021) 2
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) ICLR (2015) 9
12. 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) 1, 2
13. Li, X., Li, J., Hu, X., Yang, J.: Line-cnn: End-to-end traffic line detection with line proposal unit. T-ITS (2019) 2
14. Li, Z.Q., Ma, H.M., Liu, Z.Y.: Road lane detection with gabor filters. In: ISAI (2016) 2
15. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017) 2
16. Liu, L., Chen, X., Zhu, S., Tan, P.: Condlanenet: a top-to-down lane detection framework based on conditional convolution. In: CVPR (2021) 2
17. Liu, R., Chen, D., Liu, T., Xiong, Z., Yuan, Z.: Learning to predict 3d lane shape and camera pose from a single image via geometry constraints. In: AAAI (2022) 9
18. Neven, D., De Brabandere, B., Georgoulis, S., Proesmans, M., Van Gool, L.: Towards end-to-end lane detection: an instance segmentation approach. In: IV (2018) 2
19. Pan, X., Shi, J., Luo, P., Wang, X., Tang, X.: Spatial as deep: Spatial cnn for traffic scene understanding. In: AAAI (2018) 1, 2, 9
20. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. NeurIPS (2019) 9
21. Qin, Z., Wang, H., Li, X.: Ultra fast structure-aware deep lane detection. In: ECCV (2020) 2
22. 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) 2

23. Su, J., Chen, C., Zhang, K., Luo, J., Wei, X., Wei, X.: Structure guided lane detection. In: IJCAI-21 (2021) 2
24. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: CVPR (2020) 12, 13
25. 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) 2, 13
26. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: ICML (2019) 2
27. TuSimple: https://github.com/TuSimple/tusimple-benchmark (2017) 1
28. Wang, J., Mei, T., Kong, B., Wei, H.: An approach of lane detection based on inverse perspective mapping. In: ITSC (2014) 2
29. Wang, Y., Guizilini, V.C., Zhang, T., Wang, Y., Zhao, H., Solomon, J.: Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In: CoRL (2022) 9
30. 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) 1, 2
31. Yan, F., Nie, M., Cai, X., Han, J., Xu, H., Yang, Z., Ye, C., Fu, Y., Mi, M.B., Zhang, L.: Once-3dlanes: Building monocular 3d lane detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 17143–17152 (June 2022) 1, 10, 11
32. Yu, F., Chen, H., Wang, X., Xian, W., Chen, Y., Liu, F., Madhavan, V., Darrell, T.: Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In: CVPR (2020) 1
33. 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: ICCV (2021) 1
34. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable DETR: Deformable transformers for end-to-end object detection. In: ICLR (2021) 9, 12, 13