Supplementary document of InsMapper

Zhenhua Xu, Kwan-Yee.K.Wong, and Hengshuang Zhao

The University of Hong Kong

1 Experiment

1.1 Experiment settings

In this paper, we employ a learning rate of 6×10^{-4} and a weight decay rate of 0.01. The experiments involve 100 instance queries ($N_I = 100$) and 20 point queries ($n_p = 20$), conducted using eight NVIDIA GeForce RTX 3090 GPUs, each equipped with 24GB of memory. For the ResNet backbones, we set the batch size to 4.

In our comparison experiments, we keep parameter settings of all methods exactly the same for fair comparison, such as batch size, network layers, number of input queries, input data resolution, etc.

1.2 Road elements

InsMapper detects four types of road elements essential for vectorized HD maps: road boundaries (polyline), lane splits (polyline), pedestrian crossings (polygon), and lane centerlines (polyline with topology). It should be noted that most past works either detect three types of road elements (i.e., road boundaries, lane splits and pedestrian crossings) [8–10] or only detect one road element (e.g., lane centerlines) [1, 14].

Currently, the detected HD map is an undirected graph to unify all elements. If directed vectorized HD maps are required for specific applications (e.g., centerlines need directions), InsMapper can be easily adapted by employing directed ground truth HD maps as labels to train the network.

1.3 Topology Evaluation metric

To provide a comprehensive evaluation, we report a topology-level evaluation score in this paper, namely the TOPO metric score [3–5, 12–15]. The TOPO metric has been used in several previous studies [3–5] to evaluate the correctness of lane and road-network graph topology. The TOPO metric first randomly samples vertices v_i^* from the ground truth graph G^*. It then finds corresponding matched vertices $\hat{v}_i$ in the predicted graph $\hat{G}$ based on the closest distance. Using v_i^* as a seed node, TOPO calculates a sub-graph $G^*_{v_i^*}$, where the distance between all vertices and v_i^* is smaller than a predetermined threshold. Similarly, we obtain the sub-graph $\hat{G}_{\hat{v}_i}$ by taking $\hat{v}_i$ as the seed node. Finally, we measure the graph similarity of the two sub-graphs using precision, recall, and F1-score. The TOPO metric is the mean similarity F1-score of all sampled vertex pairs $(v_i^*, \hat{v}_i)$.

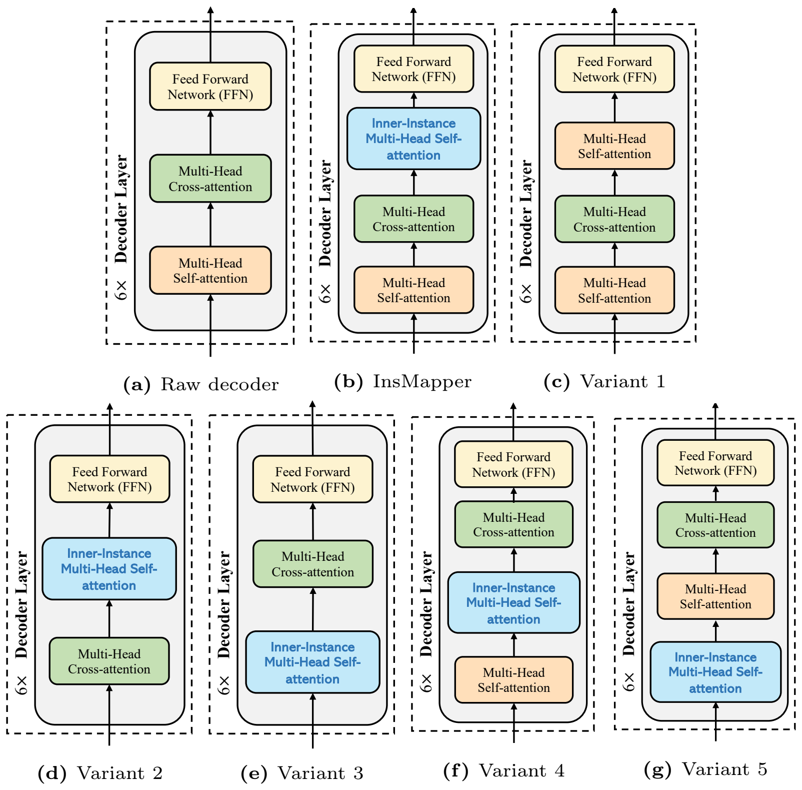
6× Decoder Layer
Feed Forward Network (FFN)
Multi-Head Cross-attention
Multi-Head Self-attention
Inner-Instance Multi-Head Self-attention
(a) Raw decoder
(b) InsMapper
(c) Variant 1
(d) Variant 2
(e) Variant 3
(f) Variant 4
(g) Variant 5

Fig. 1: Different transformer decoder designs. (a) Raw decoder of the deformable DETR. (b) InsMapper decoder. The proposed inner-instance self-attention module (blue) is incorporated. (c) Decoder variant 1. We replace the inner-instance self-attention module with a vanilla self-attention module. (d) Decoder variant 2. The self-attention module is removed from the InsMapper decoder. (e) Decoder variant 3. Swap the inner-instance self-attention module with the cross-attention module of the variant 2. (f) Variant 4. Place the inner-instance self-attention module before the cross-attention module. (g) Variant 5. Place the inner-instance self-attention module before the self-attention module. InsMapper decoder is the best design. Changing the structure of the proposed decoder will degrade the final performance.

2 Inner-instance feature aggregation

2.1 Ratio of random blocking

There is an ϵ possibility that the attention between inner-instance points is blocked (set to one, blocked). This blocking operation is inspired by [2] to enhance the robustness of the proposed module. ϵ controls the ratio of blocked attention masks. This concept is similar to dropout for better performance. However, dropout is applied after the *softmax* on the whole attention mask, while the blocking operation is before the *softmax* and it is operated only on the

Table 1: Quantitative results of ablation studies about the mask ratio.

ϵ	AP_{ped}	AP_{div}	AP_{bound}	AP_{center}	mAP	TOPO
0 (No masked attn)	42.48	50.66	53.22	41.54	46.98	51.49
25% (InsMapper)	**44.36**	**53.36**	52.77	42.35	**48.31**	51.58
50%	43.67	52.57	**53.98**	**42.95**	48.17	**51.59**
80%	42.27	52.49	53.64	41.34	47.41	49.20

Table 2: Quantitative results of ablation studies about transformer decoder designs.

Position	AP_{ped}	AP_{div}	AP_{bound}	AP_{center}	mAP	TOPO
Raw Decoder	39.41	50.04	52.13	40.10	45.42	49.19
Variant 1	40.50	50.78	52.17	41.46	46.23	49.51
Variant 2	40.96	47.85	51.11	40.45	45.09	47.98
Variant 3	40.92	46.40	51.94	38.91	44.54	46.90
Variant 4	39.09	50.09	51.56	40.21	45.24	48.16
Variant 5	44.04	49.59	52.18	41.78	46.90	49.52
InsMapper	**44.36**	**53.36**	**52.77**	**42.35**	**48.31**	**51.58**

inner-instance attention mask. Experimental results with different ϵ values are shown in Table 1.

2.2 Transformer Decoder Architectures

The transformer decoder of InsMapper may have multiple variants, which are visualized in Figure 1. There might be multiple kinds of attention layers and their positions can be altered. The evaluation results are reported in Table 2. Based on the results, the transformer decoder of InsMapper has the optimal design among these variants.

3 Instance-level class prediction

Although each instance contains n_p points, it should only have one predicted class for consistency. In MapTR, after obtaining point embeddings from the transformer decoder, it calculates a new instance-level embedding by taking the mean of all points in an instance. Then, the mean embedding is sent to the class head for class prediction. Differently, in InsMapper, we propose to use concatenation for class prediction, which preserves more information on points. Two class prediction methods are visualized

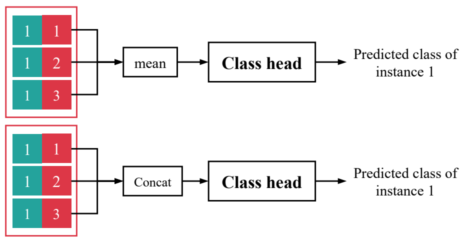

Fig. 2: Class head designs. MapTR leverages the mean of points for aggregation (upper). While InsMapper uses concatenation (lower).

in Figure 2. Experiment results are reported in Table 3. From the results, it is noted that concatenation achieves a slight performance gain. Thus, the concatenation method is used for class prediction in InsMapper.

Table 3: Quantitative results of ablation studies on the class head design.

Aggregation Method	AP_{ped}	AP_{div}	AP_{bound}	AP_{center}	mAP	TOPO
Mean	42.31	53.22	**53.70**	**43.16**	48.10	51.19
Concatenation (InsMapper)	**44.36**	**53.36**	52.77	42.35	**48.31**	**51.58**

4 Abandoned Designs

In our experiments, some designs are proved not effective for performance enhancement in our task. But we report them here which could be helpful for afterwards research.

4.1 Denosing DETR

In conventional object detection tasks, the denoising operation has been proven to effectively accelerate convergence and enhance overall performance [6, 7, 16]. However, this operation requires queries to be independent and identically distributed (i.i.d.). If this condition is not met, simply adding random noise to queries may not yield superior results. In our task, due to the strong correlation among inner-instance points, the denoising operation does not improve the vectorized HD map detection results. We report the outcomes of applying Dn-DETR in Table 4.

Table 4: Quantitative results of ablation studies about denoising DETR. For all metrics, a larger value indicates better performance.

Method	AP_{ped}	AP_{div}	AP_{bound}	AP_{center}	mAP	TOPO
InsMapper-Dn-DETR	43.74	51.34	**54.48**	42.33	47.97	50.37
InsMapper	**44.36**	**53.36**	52.77	**42.35**	**48.31**	**51.58**

In our experiments, we initially add random instance-level noise equally to all points within the same instance. Subsequently, we introduce random point-level noise to each point. Despite these modifications, we observe neither a significant improvement nor faster convergence. We attribute this unsatisfactory performance to the correlation between points.

4.2 Dynamic query generation

Another method to improve DETR in the query generation phase is dynamic query generation [2, 11, 16]. Unlike static query generation, where all queries are randomly initialized, dynamic queries are predicted based on the features obtained by the transformer encoder. In other words, the output of the transformer encoder can be utilized to generate queries with better initialization.

After obtaining the transformer encoder output F, we create grids to partition F. Each grid $F_{x,y}$ contains the local information of the input image around coordinate (x, y). Then, each grid is used to predict a dynamic query, represented as a 4-D bounding box. In conventional detection tasks, objects are typically not very large, so each grid can make satisfactory predictions of dynamic queries. However, in our vectorized HD map detection task, target objects are often very thin and very long, which cannot be effectively predicted by local grids. Consequently, dynamic queries do not yield improvements. We present the results on dynamic query generation in Table 5.

Table 5: Quantitative results of ablation studies about dynamic queries.

Method	AP_{ped}	AP_{div}	AP_{bound}	AP_{center}	mAP	TOPO
InsMapper with dynamic query	43.14	48.86	52.17	38.65	45.71	43.83
InsMapper	**44.36**	**53.36**	**52.77**	**42.35**	**48.31**	**51.58**

4.3 Inter-instance self-attention

We also attempted to exploit the inter-instance information to further enhance the vectorized HD map detection results. The inter-instance self-attention module is incorporated into the decoder layers, similar to the inner-instance self-attention module, as shown in Figure 3. First, we predict the adjacency matrix, representing the connectivity of the predicted HD map. Some lane centerline instances may intersect with each other. Adjacent instances are illustrated by colored grids. Then, for adjacent instances, the corresponding attention mask grids are set to zero (attention is allowed). Otherwise, the attention is blocked. In this way, the information exchange between points in adjacent instances is allowed to better leverage the point correlations. However, this design significantly increases resource consumption while not noticeably improving the final results. The experiment results are shown in Table 6.

We believe the reason is the sparsity of the adjacency matrix. Under most circumstances, only a few instances intersect with each other, so the adjacency matrix is very sparse, providing limited additional inter-instance information. Furthermore, it may affect the inner-instance self-attention module, which is the main reason for the performance gains of InsMapper.

Therefore, at this stage, inter-instance self-attention is not used in InsMapper. But this could be an interesting topic for future exploration.

Table 6: Quantitative results of ablation studies about inter-instance self-attention.

Inter-instance Self-attn	AP_{ped}	AP_{div}	AP_{bound}	AP_{center}	mAP	TOPO	FPS
Yes	42.40	53.21	**54.03**	**43.03**	48.17	**52.63**	6.3
No (InsMapper)	**44.36**	**53.36**	52.77	42.35	**48.31**	51.58	**7.7**

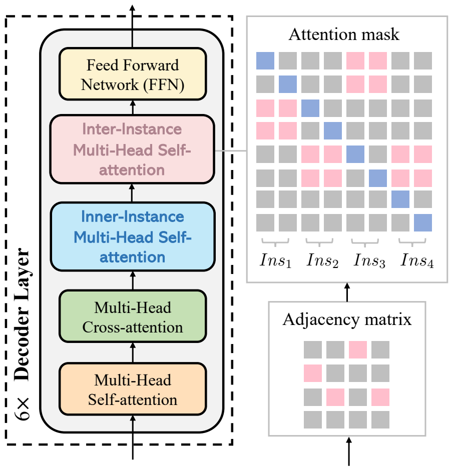
6× Decoder Layer
Feed Forward Network (FFN)
Inter-Instance Multi-Head Self-attention
Inner-Instance Multi-Head Self-attention
Multi-Head Cross-attention
Multi-Head Self-attention
Attention mask
Ins_1 Ins_2 Ins_3 Ins_4
Adjacency matrix

Fig. 3: Decoder of InsMapper with inter-instance self-attention module. In this module (the pink one), the attention between points of non-adjacent instances is blocked (grey grids). The attention is allowed for points of adjacent instances (pink grids), and diagonal grids (blue grids) to maintain the ego information of each point.

5 Additional qualitative visualizations

Qualitative visualizations on the Nuscenes validation set are shown in Figure 4 to Figure 7. The predicted map contains four classes, i.e., road boundaries (green), lane splits (red), pedestrian crossing (blue), and lane centerlines (pink). We visualize the vectorized HD map of the previous SOTA method MapTR, our proposed InsMapper, and the ground truth label. Models are trained with ResNet50 by 24 epochs.

References

1. Can, Y.B., Liniger, A., Paudel, D.P., Van Gool, L.: Structured bird's-eye-view traffic scene understanding from onboard images. In: ICCV (2021) 1
2. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: CVPR (2022) 2, 4
3. He, S., Balakrishnan, H.: Lane-level street map extraction from aerial imagery. In: WACV (2022) 1
4. He, S., Bastani, F., Abbar, S., Alizadeh, M., Balakrishnan, H., Chawla, S., Madden, S.: Roadrunner: improving the precision of road network inference from gps trajectories. In: SIGSPATIAL (2018) 1
5. He, S., Bastani, F., Jagwani, S., Alizadeh, M., Balakrishnan, H., Chawla, S., Elshrif, M.M., Madden, S., Sadeghi, M.A.: Sat2graph: road graph extraction through graph-tensor encoding. In: ECCV (2020) 1
6. Li, F., Zhang, H., Liu, S., Guo, J., Ni, L.M., Zhang, L.: Dn-detr: Accelerate detr training by introducing query denoising. In: CVPR (2022) 4
7. Li, F., Zhang, H., Xu, H., Liu, S., Zhang, L., Ni, L.M., Shum, H.Y.: Mask dino: Towards a unified transformer-based framework for object detection and segmentation. In: CVPR (2023) 4
8. Li, Q., Wang, Y., Wang, Y., Zhao, H.: Hdmapnet: A local semantic map learning and evaluation framework. In: ICRA (2022) 1
9. Liao, B., Chen, S., Wang, X., Cheng, T., Zhang, Q., Liu, W., Huang, C.: Maptr: Structured modeling and learning for online vectorized hd map construction. In: ICLR (2023) 1
10. Liu, Y., Wang, Y., Wang, Y., Zhao, H.: Vectormapnet: End-to-end vectorized hd map learning. In: ICML (2023) 1
11. Wang, Y., Zhang, X., Yang, T., Sun, J.: Anchor detr: Query design for transformer-based detector. In: AAAI (2022) 4
12. Xu, Z., Liu, Y., Gan, L., Hu, X., Sun, Y., Wang, L., Liu, M.: csboundary: City-scale road-boundary detection in aerial images for high-definition maps. arXiv:2111.06020 (2021) 1
13. Xu, Z., Liu, Y., Gan, L., Sun, Y., Liu, M., Wang, L.: Rngdet: Road network graph detection by transformer in aerial images. TGRS (2022) 1
14. Xu, Z., Liu, Y., Sun, Y., Liu, M., Wang, L.: Centerlinedet: Road lane centerline graph detection with vehicle-mounted sensors by transformer for high-definition map creation. In: ICRA (2023) 1
15. Xu, Z., Liu, Y., Sun, Y., Liu, M., Wang, L.: Rngdet++: Road network graph detection by transformer with instance segmentation and multi-scale features enhancement. RAL (2023) 1
16. Zhang, H., Li, F., Liu, S., Zhang, L., Su, H., Zhu, J., Ni, L.M., Shum, H.Y.: Dino: Detr with improved denoising anchor boxes for end-to-end object detection. In: ICLR (2023) 4

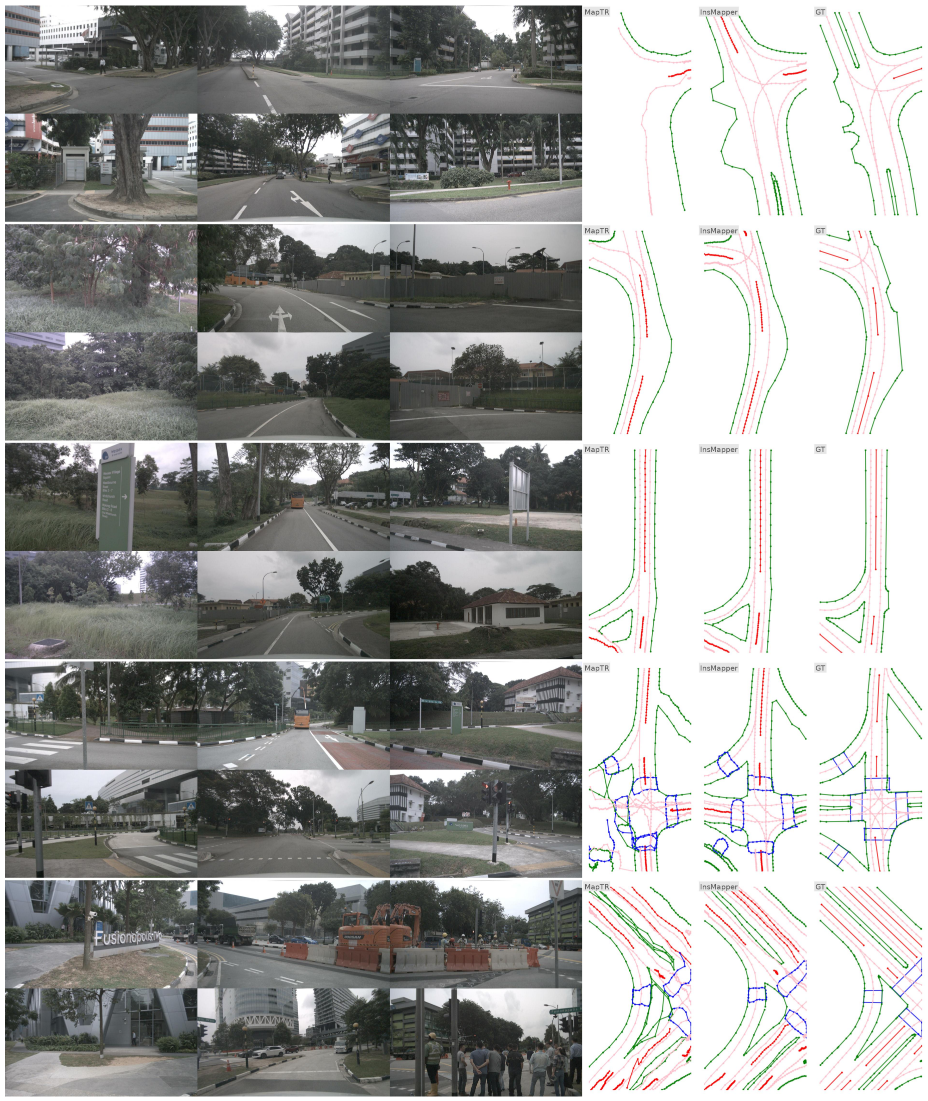

Fig. 4: Qualitative visualization. Left three columns are input 6 RGB camera images. For map columns, the first column presents MapTR's results, the second column features InsMapper's outcomes, and the final column depicts the ground truth map. The predicted map contains four classes, i.e., road boundaries (green), lane splits (red), pedestrian crossing (blue), and lane centerlines (pink).

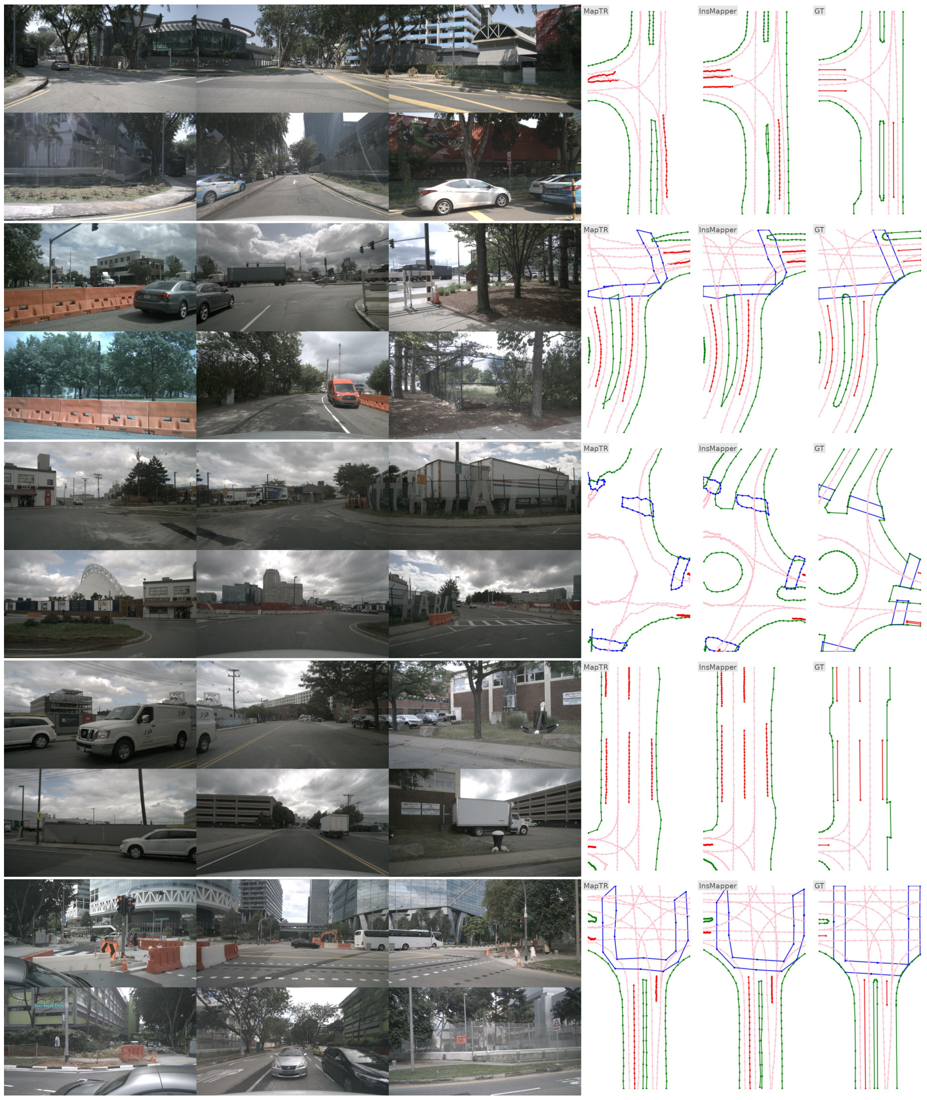

Fig. 5: Qualitative visualization. Left three columns are input 6 RGB camera images. For map columns, the first column presents MapTR's results, the second column features InsMapper's outcomes, and the final column depicts the ground truth map. The predicted map contains four classes, i.e., road boundaries (green), lane splits (red), pedestrian crossing (blue), and lane centerlines (pink).

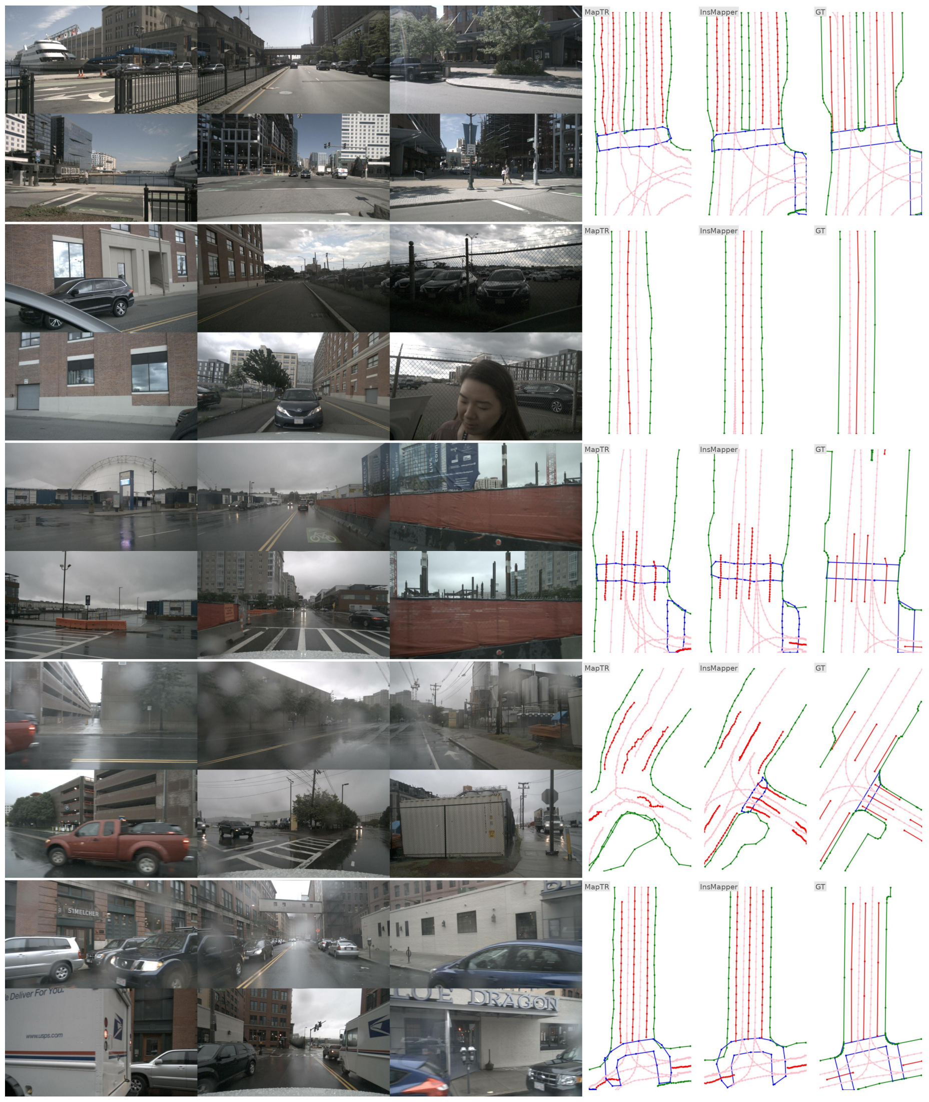

Fig. 6: Qualitative visualization. Left three columns are input 6 RGB camera images. For map columns, the first column presents MapTR's results, the second column features InsMapper's outcomes, and the final column depicts the ground truth map. The predicted map contains four classes, i.e., road boundaries (green), lane splits (red), pedestrian crossing (blue), and lane centerlines (pink).

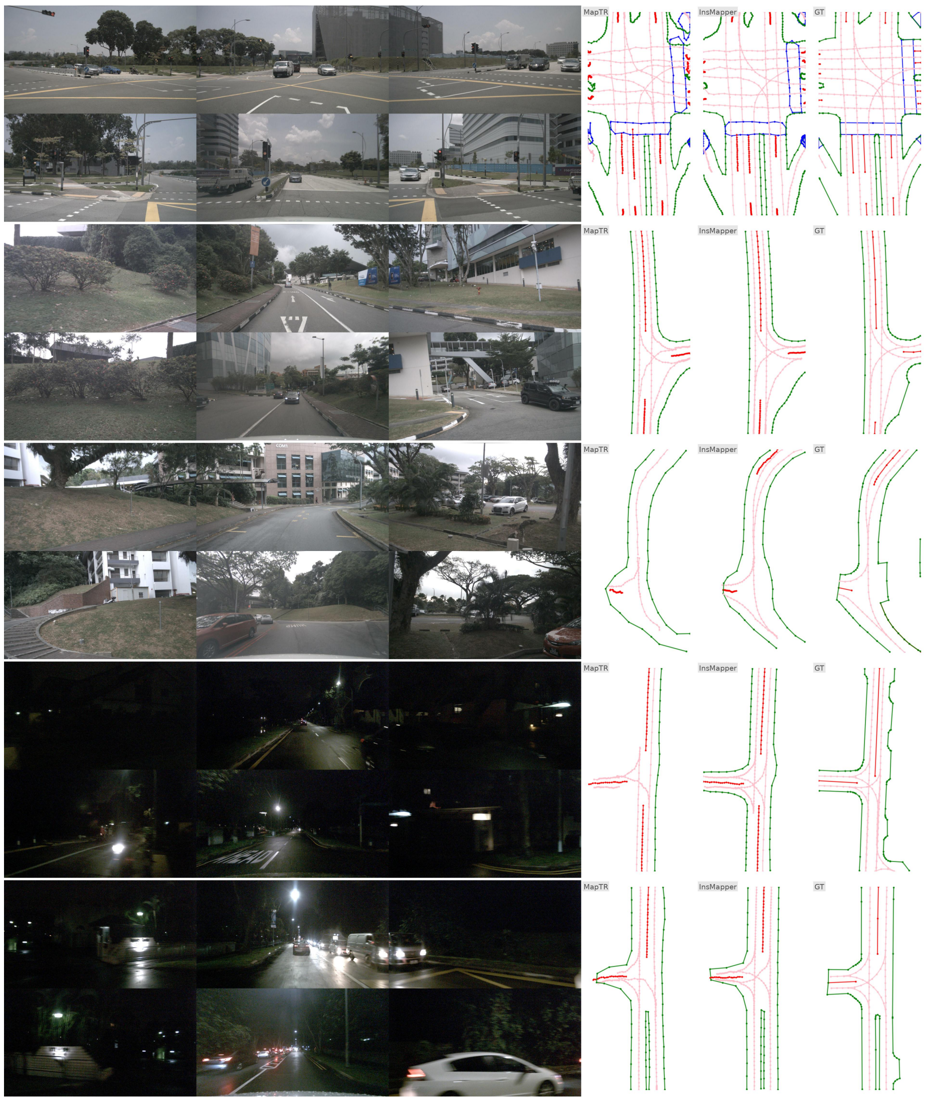

Fig. 7: Qualitative visualization. Left three columns are input 6 RGB camera images. For map columns, the first column presents MapTR's results, the second column features InsMapper's outcomes, and the final column depicts the ground truth map. The predicted map contains four classes, i.e., road boundaries (green), lane splits (red), pedestrian crossing (blue), and lane centerlines (pink).