Supplementary Material

In this supplementary material, we provide additional details which we could not include in the main paper due to space limitation. Specifically, we provide:

- Detailed illustrations on PivotNet design.
- Additional ablation studies.
- Extensive qualitative visualization results.

A. Detailed illustrations on model design

A.1. Code of the custom matching algorithm

We provide the python code of the custom matching algorithm as follows. The input *cost* refers to the distance array, where $cost[i][j]$ denotes the L_1 distance between the i-th point in the ground truth and the j-th point in the line prediction. $dp[i][j]$ denotes the lowest matching cost between the front-i points in the ground truth and the front-j points in the line prediction. mem_sort stores the minimum cost during traversal to avoid unnecessary sorting. $match\_res1$ and $match\_res2$ store the temporary matching results.

```
def pivot_dynamic_matching(cost: np.array):
  A, B = cost.shape
  assert A >= 2 and B >= 2, \
  "A line should contain two points at least"
  if A > B: # special case
    seq_dist = 0
    for j in range(B):
      seq_dist += cost[j][j]
    combination = list(range(B))
    return seq_dist, combination
  dp = np.ones((A, B)) * np.inf
  mem_sort = np.ones((A, B)) * np.inf
  match_res1 = [[] for _ in range(B)]
  match_res2 = [[] for _ in range(B)]
  # initialize
  for j in range(0, B-A+1):
    match_res1[j] = [0]
    mem_sort[0][j] = cost[0][0]
    if j == 0:
      dp[0][j] = cost[0][0]
  # update
  for i in range(1, A):
    for j in range(i, B-A + i+1):
      dp[i][j] = mem_sort[i-1][j-1] \
                 + cost[i][j]
      if dp[i][j] < mem_sort[i][j-1]:
        mem_sort[i][j] = dp[i][j]
        if i < A-1:
          match_res2[j] = match_res1[j-1] + [j]
      else:
        mem_sort[i][j] = mem_sort[i][j-1]
        if i < A -1:
          match_res2[j] = match_res2[j-1]
    if i < A-1:
      match_res1 = match_res2.copy()
      match_res2 = [[] for _ in range(B)]
  seq_dist =  dp[-1][-1]
  combination = match_res1[-2] + [B-1]
  return seq_dist, combination
```

A.2. Detailed illustrations of notations

Fig.5 illustrates the notations of *pivot dynamic matching* (PDM) in detail. As stated in Sec.3.2.3, given a ground truth sequence $S^p = \{v_n\}_{n=1}^{T}$, and a predicted line $\hat{S} = \{\hat{v}_n\}_{n=1}^{N}$, PDM searches the optimal T-combination β^* with the lowest sequence matching cost among all the T-combinations. T is the length of the ground truth sequence, and N is the predefined max number of points in a line prediction. For predicted lines with distinct point distribution, the optimal β^* is different, resulting in distinct splits of pivot sequence $\hat{S}^p$ and collinear sequence $\hat{S}^c$. Fig.5 shows examples of β^*, S^p, $\hat{S}$, $\hat{S}^p$ and $\hat{S}^c$.

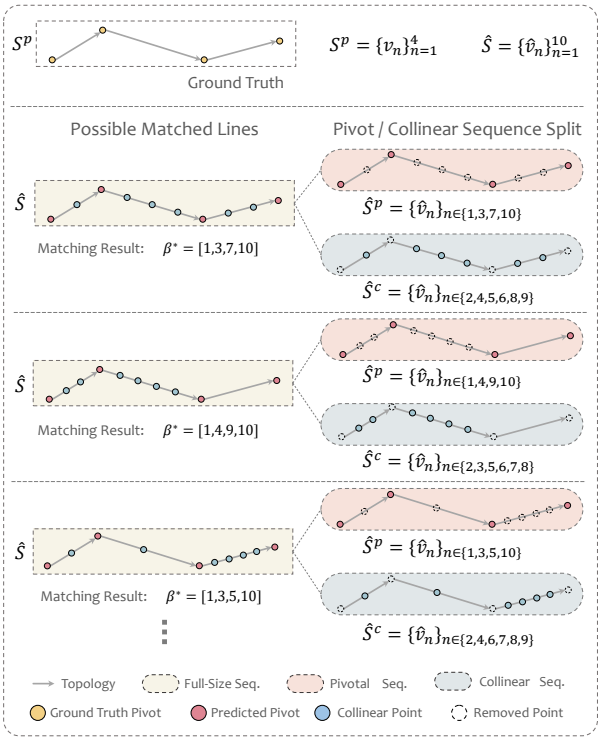

Figure 5: More illustrations of notations in pivot dynamic matching module. β^* is the optimal T-combination for the predicted lines with respect to the ground truth. The figure shows the case where $T = 4, N = 10$. The ground truth sequence S^p contains only pivot points while the predicted sequence $\hat{S}$ contains both pivot points and collinear points. $\hat{S}$ is split into a pivot sequence $\hat{S}^p$ and a collinear sequence $\hat{S}^c$ based on β^*, which is different for distinct point distribution. Predicted pivot points in $\hat{S}^p$ is in one-to-one correspondence to the ground truth sequence S^p, and $|S^p| = |\hat{S}^p| = T$, while the number of collinear points $|\hat{S}^c|$ is $N - T$.

A.3. Ground-truth pivot point generation

Pivot points in the paper are defined as the points in a map element that contribute to the overall shape and typ-

ically indicate a change in direction. Ground-truth pivot point generation can be considered as a *polyline simplification* problem, which has been studied for years [1, 3, 6, 7, 8]. Given a polyline connected by vertices, *polyline simplification* aims to find a similar polyline with fewer vertices, which we call pivot points. In this paper, we choose Visvalingam-Whyatt (VW) algorithm [8] to generate ground-truth pivot points. Given an ordered set of points, the importance of each interior point is determined by the area of triangle formed by it and its immediate neighbors. Then the point with the smallest triangle area is obtained. If the area is below the predefined threshold, the point is removed from the set. After that, the triangle area is calculated again to find the most unimportant point. This process is repeated until no triangle area is below the threshold. It is worth noting that the VW algorithm [8] is used for ground truth generation only and can be simply replaced by other pivot point generation methods like [1, 3, 6, 7].

B. Additional ablation studies

B.1. Impact of *BEV* encoder layer number

The impact of *BEV* encoder layer number is evaluated in Table 7. The performance of PivotNet improves with more encoder and decoder layers. Even with single encoder and decoder layer, PivotNet achieves satisfying performance, *i.e.,* 36.0% in mAP.

Layer Num.	$AP_{divider}$	AP_{ped}	$AP_{boundary}$	mAP
(1, 1)	38.2	33.7	36.0	36.0
(2, 4)	45.5	36.9	42.4	41.6
(4, 2)	46.3	38.2	41.8	42.1
(4, 4)	47.6	38.3	43.8	43.3
(6, 6)	**48.0**	**38.9**	**45.9**	**44.3**

Table 7: Impact of *BEV* encoder layer number. The gray row represents the setting we use in default.

B.2. Impact of the maximum instance number

We evaluate the effect of maximum instance number in Table 8. We define the maximum instance number based on the rule that it should be larger than the instance number in typical ground truth. In default, we choose maximum instance number of $(20, 25, 15)$ for $lane\text{-}divider$, $ped\text{-}crossing$, and $road\text{-}boundary$ respectively. We provide performances with other number settings in Table 8. The performance of PivotNet get saturated when the maximum instance number is set to $(20, 25, 15)$.

B.3. Impact of the maximum pivot point number

The impact of the maximum pivot point number is evaluated in Table 9. The maximum number of pivot points roughly represents the complexity of map elements that PivotNet is able to model. For a certain type of map element, too small maximum number will lead to insufficient modeling capability, yet too large maximum number will increase the learning burden of the model. Therefore, an appropriate value is required for trade-off.

Instance Num.	$AP_{divider}$	AP_{ped}	$AP_{boundary}$	mAP
(15, 20, 10)	46.5	37.1	42.8	42.2
(20, 25, 15)	**47.6**	38.3	**43.8**	**43.3**
(30, 30, 30)	47.6	**38.9**	43.2	43.2

Table 8: Impact of the maximun instance number. The gray row represents the setting we use in default.

Pivot Point Num.	10	20	30
$AP_{divider}$	**47.6**	47.5	42.6
Pivot Point Num.	30	50	60
$AP_{boundary}$	43.8	**45.4**	43.2

Table 9: Impact of the maximum pivot point number.

C. More qualitative visualization results

Fig.6-9 provide extensive visualization results for comparison with SOTA approaches [2, 4] on various environmental conditions. The visualization of HDMapNet [2] is reproduced with its public code, and that of MapTR [4] is generated by its released model checkpoint. Even in challenging road scenarios like intersections and dense roads, PivotNet produces accurate yet compact representations.

We visualize the predictions of the PivotNet with Swin-Tiny [5] backbone on nuScenes. The predictions on different environmental conditions are presented. Even at the most challenging night scenario, the predictions near the vehicle closely match the ground truth (see Fig. 9). Seen from visualization, the modeling of PivotNet is flexible and can describe map elements of arbitrary shape, including line segments, curves, and combinations of them.

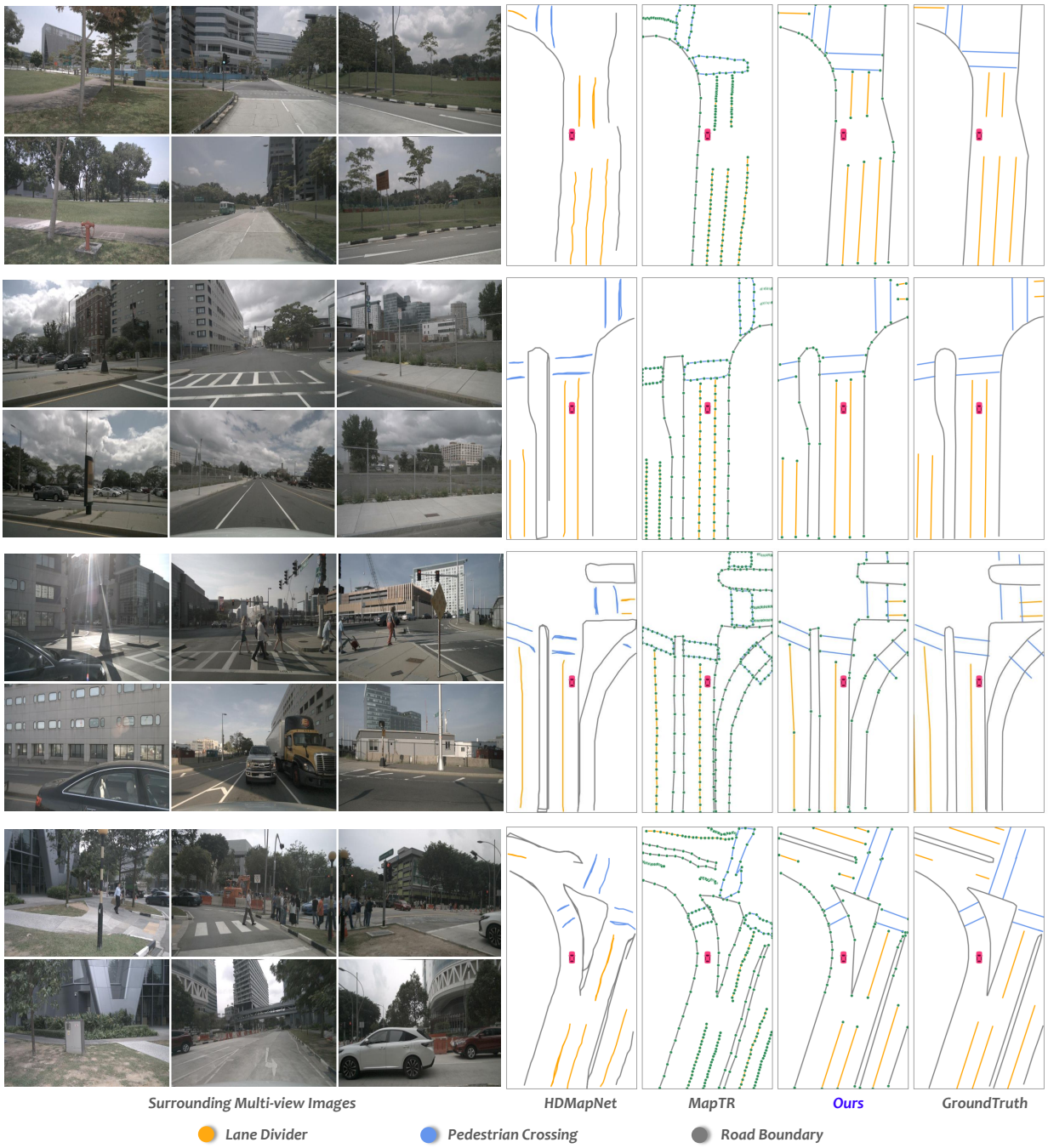

Figure 6: Visualization results under the weather condition of *sunny*.

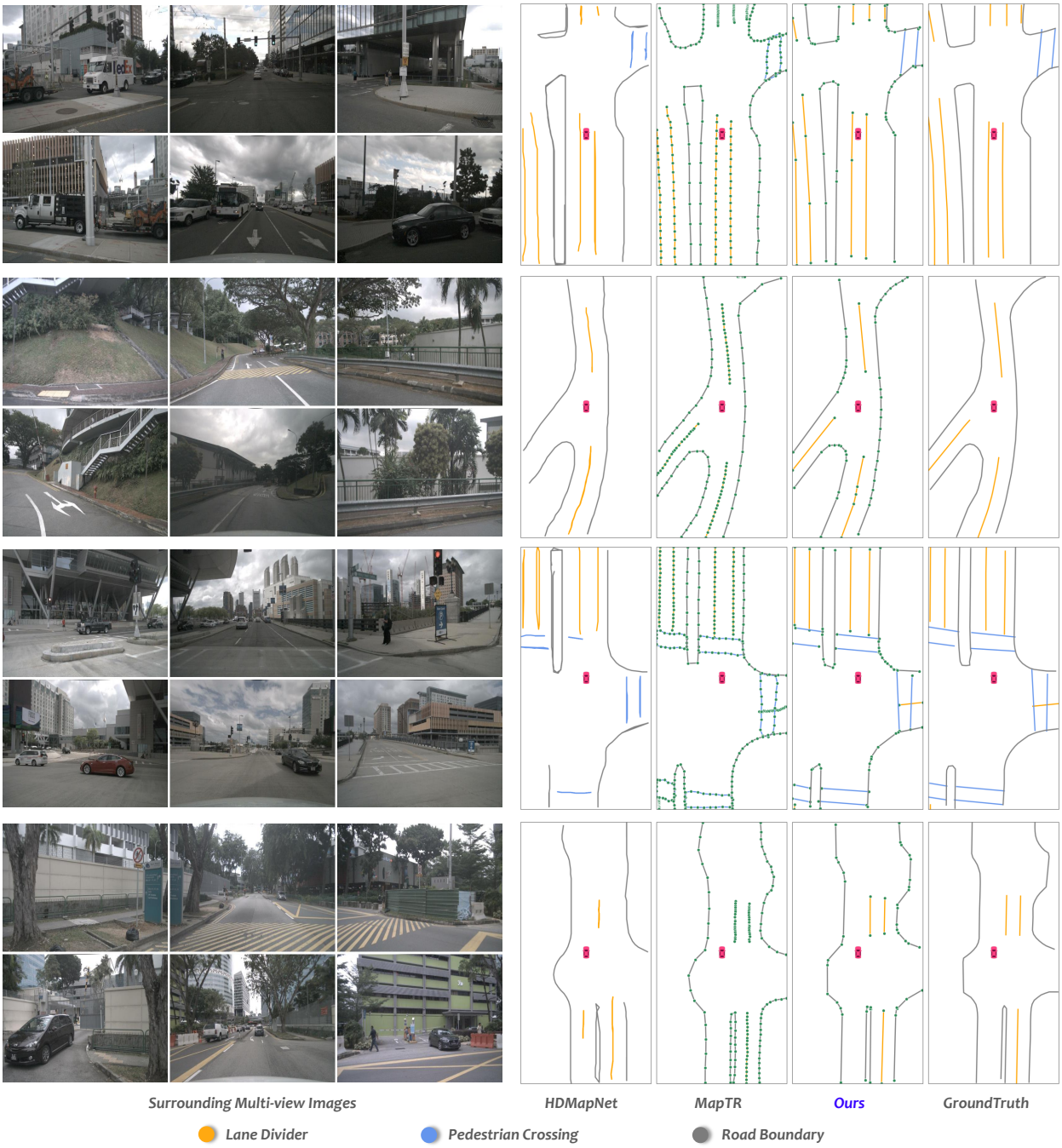

Figure 7: Visualization results under the weather condition of *cloudy*.

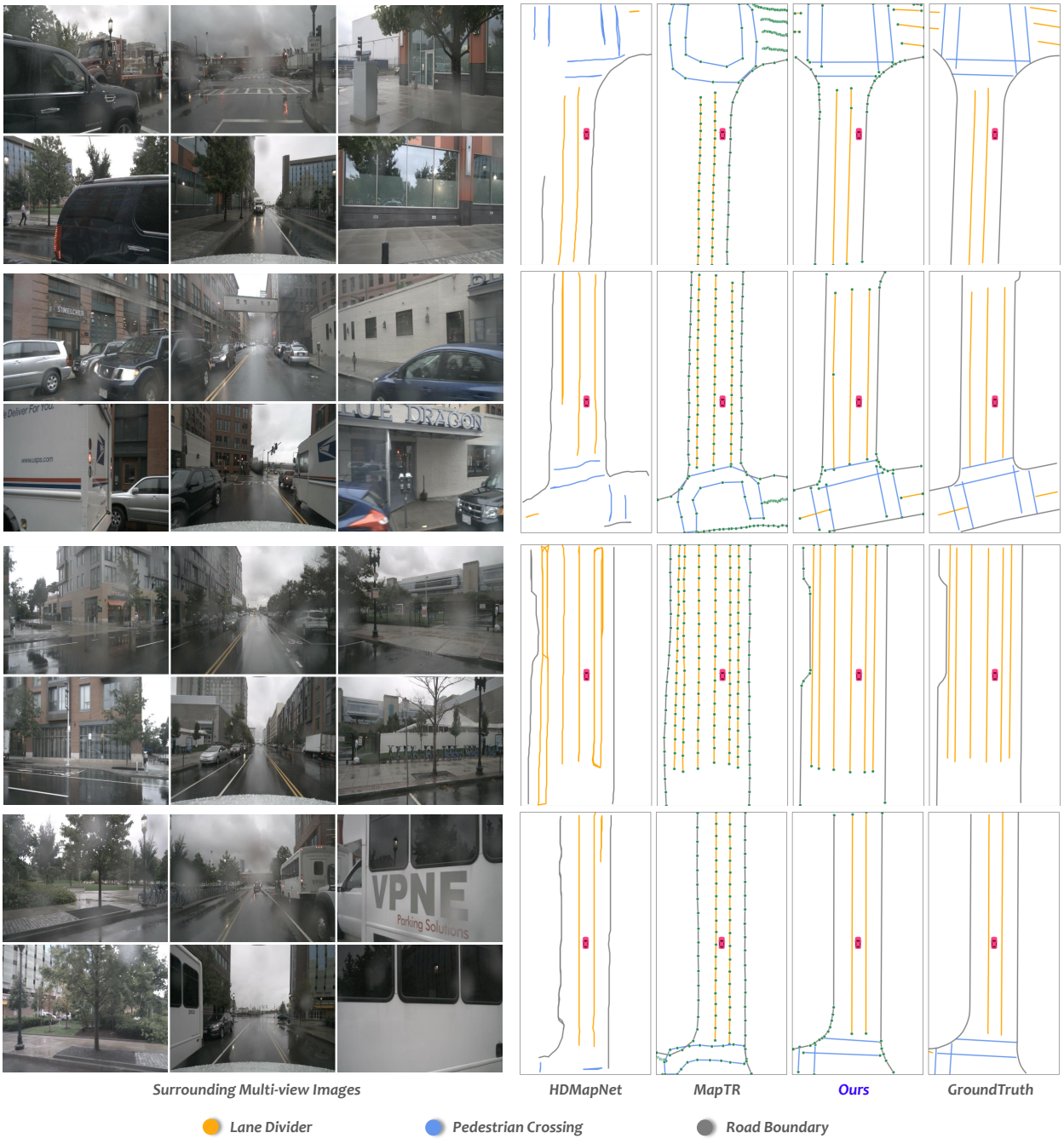

Figure 8: Visualization results under the weather condition of *rainy*.

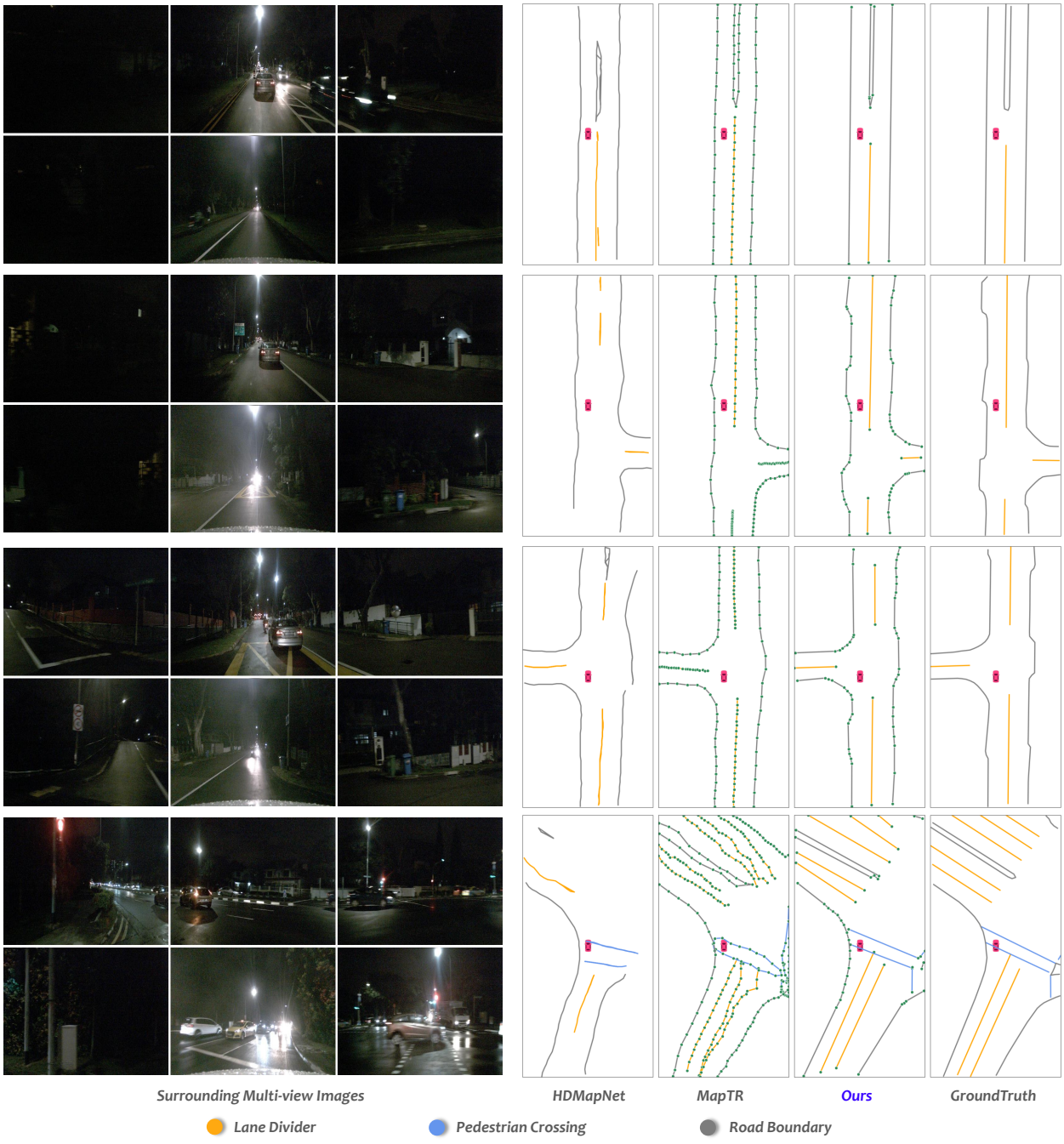

Figure 9: Visualization results under the lighting condition of *nighttime*.

References

[1] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, 10(2):112–122, 1973. 2

[2] Qi Li, Yue Wang, Yilun Wang, and Hang Zhao. Hdmapnet: An online hd map construction and evaluation framework. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4628–4634. IEEE, 2022. 2

[3] Zhilin Li. An algorithm for compressing digital contour data. *The Cartographic Journal*, 25(2):143–146, 1988. 2

[4] Bencheng Liao, Shaoyu Chen, Xinggang Wang, Tianheng Cheng, Qian Zhang, Wenyu Liu, and Chang Huang. Maptr: Structured modeling and learning for online vectorized hd map construction. In *International Conference on Learning Representations*, 2023. 2

[5] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 2

[6] Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer graphics and image processing*, 1(3):244–256, 1972. 2

[7] Helmut Ratschek, J Rokne, and M Leriger. Robustness in gis algorithm implementation with application to line simplification. *International Journal of Geographical Information Science*, 15(8):707–720, 2001. 2

[8] Maheswari Visvalingam and James D Whyatt. Line generalization by repeated elimination of points. In *Landmarks in Mapping*, pages 144–155. Routledge, 2017. 2