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.

```python
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):
        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 = [[0] for _ in range(B)]
    match_res2 = [[0] 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] = min(
                    mem_sort[i-1][j-1] + [0],
                    dp[i-1][j],
                    mem_sort[i][j-1] + [0]
                )
                if i < A - 1:
                    match_res2[j] = match_res1[j-1] + [0]
                    if j < A - 1:
                        match_res2[j] = match_res2[j-1]
                    else:
                        match_res2[j] = (match_res2[j] - match_res2[j-1])
                        if i < A - 1:
                            match_res2[j] = match_res2[j-1]
                        if i < A - 1:
                            match_res2[j] = match_res2[j-1]
                            if j < A - 1:
                                match_res2[j] = match_res2[j-1]
                        else:
                            match_res2[j] = match_res2[j-1]
            if j == 0:
                dp[0][j] = np.inf
            for j in range(1, B):
                dp[i][j] = min(dp[i][j], dp[i-1][j-1])
            dp[i][j] = dp[i][j] + [0]
            match_res1[j] = match_res2[j] + [0]
        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 $\beta^*$ 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 $\beta^*$ is different, resulting in distinct splits of pivot sequence $\hat{S}^p$ and collinear sequence $\hat{S}^c$. Fig. 5 shows examples of $\beta^*$, $S^p$, $\hat{S}$, $\hat{S}^p$ and $\hat{S}^c$.

![Figure 5: More illustrations of notations in pivot dynamic matching module. $\beta^*$ 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 $\beta^*$, 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 $|S^p| = |\hat{S}| = T$, while the number of collinear points $|\hat{S}^c|$ is $N - T$.](image)

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.

<table>
<thead>
<tr>
<th>Layer Num.</th>
<th>$AP_{\text{divider}}$</th>
<th>$AP_{\text{ped}}$</th>
<th>$AP_{\text{boundary}}$</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 1)</td>
<td>38.2</td>
<td>33.7</td>
<td>36.0</td>
<td>36.0</td>
</tr>
<tr>
<td>(2, 4)</td>
<td>45.5</td>
<td>36.9</td>
<td>42.4</td>
<td>41.6</td>
</tr>
<tr>
<td>(4, 2)</td>
<td>46.3</td>
<td>38.2</td>
<td>41.8</td>
<td>42.1</td>
</tr>
<tr>
<td>(4, 4)</td>
<td>47.6</td>
<td>38.3</td>
<td>43.8</td>
<td>43.3</td>
</tr>
<tr>
<td>(6, 6)</td>
<td><strong>48.0</strong></td>
<td><strong>38.9</strong></td>
<td><strong>45.9</strong></td>
<td><strong>44.3</strong></td>
</tr>
</tbody>
</table>

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-divider, ped-crossing, and road-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).

<table>
<thead>
<tr>
<th>Instance Num.</th>
<th>$AP_{\text{divider}}$</th>
<th>$AP_{\text{ped}}$</th>
<th>$AP_{\text{boundary}}$</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>(15, 20, 10)</td>
<td>46.5</td>
<td>37.1</td>
<td>42.8</td>
<td>42.2</td>
</tr>
<tr>
<td>(20, 25, 15)</td>
<td><strong>47.6</strong></td>
<td>38.3</td>
<td><strong>43.8</strong></td>
<td><strong>43.3</strong></td>
</tr>
<tr>
<td>(30, 30, 30)</td>
<td>47.6</td>
<td><strong>38.9</strong></td>
<td>43.2</td>
<td>43.2</td>
</tr>
</tbody>
</table>

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

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.

<table>
<thead>
<tr>
<th>Pivot Point Num.</th>
<th>$AP_{\text{divider}}$</th>
<th>$AP_{\text{ped}}$</th>
<th>$AP_{\text{boundary}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td><strong>47.6</strong></td>
<td>47.5</td>
<td><strong>43.4</strong></td>
</tr>
<tr>
<td>20</td>
<td>47.5</td>
<td>42.6</td>
<td><strong>43.4</strong></td>
</tr>
<tr>
<td>30</td>
<td>45.4</td>
<td><strong>43.4</strong></td>
<td>43.4</td>
</tr>
</tbody>
</table>

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


