Supplementary Document: PolyDiffuse: Polygonal Shape Reconstruction via Guided Set Diffusion Models

Jiacheng Chen **Ruizhi Deng** **Yasutaka Furukawa**
Simon Fraser University

The supplementary document is organized as follows:

- §1: The derivation of the Guided Set Diffusion Models based on the DDPM [4] formulation.
- §2: Additional implementation details, including:
  - Architectures and implementation details of the guidance networks.
  - Architectures and implementation details of the denoising networks.
  - Implementation details of the permutation loss $L_{\text{perm}}(\phi)$ for the guidance training.
  - Implementation details of the diffusion models framework.
- §3: More details and analyses on the augmented matching criterion for the mean AP metric to evaluate HD map reconstruction.
- §4: Extra experimental results including:
  - Additional ablation studies on the design choices of the denoising networks.
  - Additional qualitative results for floorplan and HD map reconstruction.

1 Derivation of Guided Set Diffusion Models (GS-DM)

In this section, we present more details of the derivation of the Guided Set Diffusion Models (GS-DM) proposed in §3 of the main paper.

1.1 Forward process

We derive Eq. 7, 8, and 9 in Sec. 3 of the main paper by induction. The trivial case of $t = 1$ can be easily verified. Let $\mathbf{x}^i_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\mathbf{x}^i_0 + \bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0, t-1, i) + \bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, t-1, i)\boldsymbol{\epsilon}^i_{t-1}$ for $t > 1$ and $\boldsymbol{\epsilon}^i_{t-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. By definition, we have

$$\begin{aligned}
\mathbf{x}^i_t =& \sqrt{\alpha_t}\mathbf{x}^i_{t-1} + \boldsymbol{\mu}_\phi(\mathbf{x}_0, t, i) + \sqrt{1-\alpha_t}\boldsymbol{\sigma}_\phi(\mathbf{x}_0, t, i)\boldsymbol{\epsilon}^i_t \\
=& \sqrt{\alpha_t}(\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}^i_0 + \bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0, t-1, i) + \bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, t-1, i)\boldsymbol{\epsilon}^i_{t-1}) + \\
& \boldsymbol{\mu}_\phi(\mathbf{x}_0, t, i) + \sqrt{1-\alpha_t}\boldsymbol{\sigma}_\phi(\mathbf{x}_0, t, i)\boldsymbol{\epsilon}^i_t \\
=& \sqrt{\bar{\alpha}_t}\mathbf{x}^i_0 + \sqrt{\alpha_t}\bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0, t-1, i) + \boldsymbol{\mu}_\phi(\mathbf{x}_0, t, i) + \\
& \sqrt{\alpha_t}\bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, t-1, i)\boldsymbol{\epsilon}^i_t + \sqrt{1-\alpha_t}\boldsymbol{\sigma}_\phi(\mathbf{x}_0, t, i)\boldsymbol{\epsilon}^i_t
\end{aligned} \tag{1}$$

where $\boldsymbol{\epsilon}^i_t$ is a standard Gaussian variable independent of $\boldsymbol{\epsilon}^i_{t-1}$. Since $\boldsymbol{\epsilon}^i_{t-1}$ and $\boldsymbol{\epsilon}^i_t$ are independent, $\sqrt{\alpha_t}\bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, t-1, i)\boldsymbol{\epsilon}^i_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\sigma}_\phi(\mathbf{x}_0, t, i)\boldsymbol{\epsilon}^i_t$ is a Gaussian random variable of zero mean and a variance of $\alpha_t\bar{\boldsymbol{\sigma}}^2_\phi(\mathbf{x}_0, t-1, i) + (1-\alpha_t)\boldsymbol{\sigma}^2_\phi(\mathbf{x}_0, t, i)$. Thus we have

$$\bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0, t, i) := \sqrt{\alpha_t}\bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0, t-1, i) + \boldsymbol{\mu}_\phi(\mathbf{x}_0, t, i) \text{ and} \tag{2}$$

$$\bar{\boldsymbol{\sigma}}^2_\phi(\mathbf{x}_0, t, i) := \alpha_t\bar{\boldsymbol{\sigma}}^2_\phi(\mathbf{x}_0, t-1, i) + (1-\alpha_t)\boldsymbol{\sigma}^2_\phi(\mathbf{x}_0, t, i), \tag{3}$$

which correspond to Eq.8 and Eq.9 of the main paper, and we further obtain

$$q(\mathbf{x}^i_t|\mathbf{x}^i_0) = \mathcal{N}(\mathbf{x}^i_t;\ \sqrt{\bar{\alpha}_t}\mathbf{x}^i_0 + \bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0, t, i),\ \bar{\boldsymbol{\sigma}}^2_\phi(\mathbf{x}_0, t, i)\mathbf{I}), \tag{4}$$

which is the Eq.7 of the main paper.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

1.2 Reverse process

Assuming we are given the guidance networks $\boldsymbol{\mu}_\phi$ and $\boldsymbol{\sigma}_\phi$ and the noise scales $\boldsymbol{\sigma}_t$s of the reverse process, we follow a style similar to DDPM [4] to derive a sampling step in the reverse process. At each sampling step in the reverse process, $p_\theta(\mathbf{x}_{t-1}^i|\mathbf{x}_t^i)$, the Gaussian distribution to sample $\mathbf{x}_{t-1}^i$ from is supposed to have the same mean as $q(\mathbf{x}_{t-1}^i|\mathbf{x}_t^i, \mathbf{x}_0)$ defined by the forward process, which is also Gaussian, to minimize their KL divergence. As the joint distribution of $\mathbf{x}_{t-1}^i$ and $\mathbf{x}_t^i$ conditioned on $\mathbf{x}_0$ is Gaussian, the mean of $q(\mathbf{x}_{t-1}^i|\mathbf{x}_t^i, \mathbf{x}_0)$ can be easily derived with the following closed form [9, Chapter 8.1.3]:

$$\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0^i + \bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0^i, t-1, i) + \frac{\sqrt{\alpha_t}\bar{\boldsymbol{\sigma}}_\phi^2(\mathbf{x}_0, t-1, i)}{\bar{\boldsymbol{\sigma}}_\phi^2(\mathbf{x}_0, t, i)}(\mathbf{x}_t^i - \sqrt{\bar{\alpha}_t}\mathbf{x}_0^i - \bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0, t, i)). \tag{5}$$

Since $\mathbf{x}_t^i = \sqrt{\bar{\alpha}_t}\mathbf{x}_0^i + \bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0, t, i) + \bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, t, i)\boldsymbol{\epsilon}^i$ for $\boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, Equation 5 can be rewritten as

$$\begin{aligned}
&\sqrt{\bar{\alpha}_{t-1}}\mathbf{x}_0^i + \bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0^i, t-1, i) + \bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, t, i)\frac{\sqrt{\alpha_t}\bar{\boldsymbol{\sigma}}_\phi^2(\mathbf{x}_0, t-1, i)}{\bar{\boldsymbol{\sigma}}_\phi^2(\mathbf{x}_0, t, i)}\boldsymbol{\epsilon}^i. \\
=&\frac{1}{\sqrt{\alpha_t}}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0^i + \bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, t, i)\frac{\alpha_t\bar{\boldsymbol{\sigma}}_\phi^2(\mathbf{x}_0, t-1, i)}{\bar{\boldsymbol{\sigma}}_\phi^2(\mathbf{x}_0, t, i)}\epsilon^i) + \bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0^i, t-1, i) \\
=&\frac{1}{\sqrt{\alpha_t}}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0^i + \bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, t, i)\frac{\bar{\boldsymbol{\sigma}}_\phi^2(\mathbf{x}_0, t, i) - (1-\alpha_t)\boldsymbol{\sigma}_\phi^2(\mathbf{x}_0, t, i)}{\bar{\boldsymbol{\sigma}}_\phi^2(\mathbf{x}_0, t, i)}\boldsymbol{\epsilon}^i) + \bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0^i, t-1, i) \\
=&\frac{1}{\sqrt{\alpha_t}}(\sqrt{\bar{\alpha}_t}\mathbf{x}_0^i + \bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, t, i)\boldsymbol{\epsilon}^i + \bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0^i, t, i) - \bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0^i, t, i) - \frac{(1-\alpha_t)\boldsymbol{\sigma}_\phi^2(\mathbf{x}_0, t, i)}{\bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, t, i)}\boldsymbol{\epsilon}^i) \\
&+ \bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0^i, t-1, i) \\
=&\frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t^i - \bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0^i, t, i) - \frac{(1-\alpha_t)\boldsymbol{\sigma}_\phi^2(\mathbf{x}_0, t, i)}{\bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, t, i)}\boldsymbol{\epsilon}^i) + \bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0^i, t-1, i).
\end{aligned} \tag{6}$$

As our formulation directly parameterizes $\bar{\sigma}_\phi$, we further simplify Eq. 6 by defining $\bar{\sigma}_\phi(\mathbf{x}_0, t, i) := \sqrt{1-\bar{\alpha}_t}C(\mathbf{x}_0, i)$ where $C(\mathbf{x}_0, i)$ is independent of the timestep t, thus implicitly defining $\boldsymbol{\sigma}_\phi(\mathbf{x}_0, t, i) = \bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, T, i) = C(\mathbf{x}_0, i)$. Such a parameterization makes $\{\bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, t, i)\}_t$ simply an interpolation between 0 and $\bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, T, i)$ and further simplifies Eq. 6 as

$$\frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t^i - \bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0^i, t, i) - \bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, t, i)\frac{1-\alpha_t}{1-\bar{\alpha}_t}\boldsymbol{\epsilon}^i) + \bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0^i, t-1, i). \tag{7}$$

During inference, we replace $\bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0, t, i)$ and $\bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, t, i)$ with $\bar{\boldsymbol{\mu}}_\phi(\hat{\mathbf{x}}_0, t, i)$ and $\bar{\boldsymbol{\sigma}}_\phi(\hat{\mathbf{x}}_0, t, i)$ respectively and replace $\boldsymbol{\epsilon}^i$ with the denoising network $\boldsymbol{\epsilon}_\theta^i$ to define the mean of sampling distribution $p_\theta(\mathbf{x}_{t-1}^i|\mathbf{x}_t^i)$ and derive a sampling step in the reverse process as

$$\mathbf{x}_{t-1}^i = \frac{1}{\sqrt{\alpha_t}}\left[\mathbf{x}_t^i - \bar{\boldsymbol{\mu}}_\phi(\hat{\mathbf{x}}_0, t, i) - \bar{\boldsymbol{\sigma}}_\phi(\hat{\mathbf{x}}_0, t, i)\frac{1-\alpha_t}{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_\theta^i(\mathbf{x}_t, t, \mathbf{y})\right] + \bar{\boldsymbol{\mu}}_\phi(\hat{\mathbf{x}}_0, t-1, i) + \boldsymbol{\sigma}_t\mathbf{z}^i, \tag{8}$$

which is the Eq.10 of the main paper.

2 Additional implementation details

In this section, we complement §4 of the main paper by providing the complete implementation details of PolyDiffuse for the two different tasks.

2.1 Guidance networks

The implementation details of the guidance networks are shared across the two tasks.

Model architectures: In §4 of the main paper, we parameterize $\bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0, T, i)$ and $\bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, T, i)$ with two Transformers $\mathrm{Trans}_{\bar{\boldsymbol{\mu}}_\phi}(\mathbf{x}_0, i)$ and $\mathrm{Trans}_{\bar{\boldsymbol{\sigma}}_\phi}(\mathbf{x}_0, i)$, and define $\bar{\boldsymbol{\mu}}_\phi(\mathbf{x}_0, t, i)$ and $\bar{\boldsymbol{\sigma}}_\phi(\mathbf{x}_0, t, i)$ with

Eq.14. The two Transformers are implemented with a DETR-style [1] Transformer decoder[1]: The Transformer decoder contains two shared attention-based decoder layers and two separate linear projection heads for $\bar{\boldsymbol{\mu}}_\phi$ and $\bar{\boldsymbol{\sigma}}_\phi$, respectively. Each decoder layer consists of an intra-element self-attention layer and a global self-attention layer, whose outputs are fused by addition. Each vertex in the input $\mathbf{x}_0$ becomes an input node of the Transformer decoder, and the node feature consists of three positional encodings [14]: 1) positional encoding of the X-axis coordinate, 2) positional encoding of the Y-axis coordinate, 3) positional encoding of the vertex index inside the element. The sequence of each element starts with a special dummy node (similar to the SOS token in language modeling), whose final encodings are used to compute the $\bar{\boldsymbol{\mu}}_\phi$ and $\bar{\boldsymbol{\sigma}}_\phi$ of the element. The dimension of the positional encoding is 128, and the hidden dimension of the Transformer decoder is 256.

Training details: The guidance training is summarized by Algorithm 1 of the main paper. We train the Transformer decoder for 3125 iterations with batch size 32. Adam optimizer is employed with a learning rate of 2e-4 and a weight decay rate of 1e-4.

2.2 Denoising networks

Since the implementations of the denoising networks are based on the corresponding state-of-the-art task-specific models, we describe them separately.

Floorplan reconstruction: The denoising network for the floorplan reconstruction task refers to RoomFormer [15] and its official implementation[2]. RoomFormer is a DETR-based [1] model consisting of a ResNet50 [3] image backbone, a Transformer encoder to process and aggregate image information, and a Transformer decoder with two-level learnable embeddings/cooridnates to predict a set of polygon vertices from the image information. The Transformer has 6 encoder layers and 6 decoder layers with an embedding dimension of 256, and deformable-attention [16] is employed for all cross-attention layers and the self-attention layers in the Transformer encoder.

We follow the same architecture as RoomFormer, and have discussed the key modifications to turn the model into a denoising function (§4 of the main paper) and our improvements to lift up its overall performance (§5 of the main paper). A minor modification omitted in the main paper is that we add an intra-element attention layer to each Transformer decoder layer to increase the model capacity, as we found the model converges obviously slower under the denoising formulation than the regression/detection formulation of RoomFormer. In each decoder layer, the outputs of the intra-element attention are added to the outputs of the original global attention. The intra-element attention increases the number of overall parameters by ~5%, and the corresponding ablation study is in §4.1 of this appendix, showing minor but recognizable improvements.

For training the denoising network, we keep the same setups as RoomFormer except that we increase the number of training iterations (ablation study is in §5 of the main paper). Adam optimizer is employed with a base learning rate of 2e-4 for all parameters, and the learning rate decays by a factor of 0.1 for the last 20% iterations.

HD map reconstruction: The denoising network for the HD map reconstruction task refers to MapTR [7] and its official implementation[3]. The overall design of MapTR is very similar to RoomFormer, where the keys are the "hierarchical" or "two-level" query embeddings for DETR-style Transformer and a loss based on "hierarchical bipartite matching". We use the same model config file provided in MapTR's official codebase and convert the model into a denoising function as described in §4 of the main paper. Similar to what we did to RoomFormer above, we add an intra-element attention layer to each Transformer decoder layer to facilitate convergence.

We employ an Adam optimizer with a base learning rate of 6e-4 and a weight decay factor of 1e-4. A cosine learning rate scheduler is used. To further facilitate convergence under the denoising formulation, we load the ResNet image backbone and Transformer encoder from the pre-trained MapTR, and set the initial learning rate of the image backbone to be 0.1 of the base learning rate.

When using MapTR as the proposal generator to produce the initial reconstruction for PolyDiffuse, we only consider predicted instances with a confidence score higher than 0.5 as MapTR's positive

[1] `https://github.com/facebookresearch/detr/blob/main/models/transformer.py`

[2] `https://github.com/ywyue/RoomFormer`

[3] `https://github.com/hustvl/MapTR`

predictions. PolyDiffuse takes and updates the positive predictions of MapTR and keeps the remaining low-confidence predictions unchanged. Only the positive predictions are visualized for the qualitative results of MapTR and PolyDiffuse.

2.3 Permutation loss

Directly computing the $L_{\text{perm}}(\phi)$ as defined in Eq.14 of the main paper requires finding $\mathbf{x}_0^* = \arg\max_{\mathbf{x}_0' \in \mathcal{P}^!(\mathbf{x}_0) \setminus \{\mathbf{x}_0\}} L_{\text{Triplet}}(\mathbf{x}_t, \mathbf{x}_0, \mathbf{x}_0')$ and $\mathbf{x}_t^* = \arg\max_{\mathbf{x}_t' \in \mathcal{P}^!(\mathbf{x}_t) \setminus \{\mathbf{x}_t\}} L_{\text{Triplet}}(\mathbf{x}_0, \mathbf{x}_t, \mathbf{x}_t')$, where the size of $\mathcal{P}^!(\mathbf{x}_0)$ and $\mathcal{P}^!(\mathbf{x}_t)$ is $N!$. This enumeration over all $N!$ permutations of $\mathbf{x}_0$ and $\mathbf{x}_t$ immediately becomes computationally prohibitive when N gets large.

To reduce the computational cost and make Eq.14 practically feasible, we propose to approximate $L_{\text{perm}}(\phi)$ with element-level triplet losses. Concretely, we first enumerate all pairs of elements $\mathbf{x}_0^i$ and $\mathbf{x}_t^j$ for $i, j = 1, \ldots, N$ to compute a $N \times N$ distance matrix D. The entry $D(i, j)$ is the minimum distance between two elements $\mathbf{x}_0^i$ and $\mathbf{x}_t^j$, considering all possible vertex-level permutations (*i.e.*, two variants for a polyline, and $2(N_i - 1)$ variants for a polygon with N_i vertices, similar to the "point-level matching" in MapTR [7]). We then replace the sample-level triplet loss in $L_{\text{perm}}(\phi)$ with N element-level triplet losses, and define the computationally feasible proxy loss $\hat{L}_{\text{perm}}(\phi)$ as:

$$\hat{L}_{\text{perm}}(\phi) = \sum_{i=1,\ldots,N} \max_{1 \le j \le N, j \neq i} L_{\text{Triplet}}(\mathbf{x}_t^i, \mathbf{x}_0^i, \mathbf{x}_0^j) + \max_{1 \le j \le N, j \neq i} L_{\text{Triplet}}(\mathbf{x}_0^i, \mathbf{x}_t^i, \mathbf{x}_t^j), \tag{9}$$

$$L_{\text{Triplet}}(\mathbf{x}_t^i, \mathbf{x}_0^i, \mathbf{x}_0^j) = \max\left(0, \alpha + D(i, i) - D(j, i)\right), \tag{10}$$

$$L_{\text{Triplet}}(\mathbf{x}_0^i, \mathbf{x}_t^i, \mathbf{x}_t^j) = \max\left(0, \alpha + D(i, i) - D(i, j)\right). \tag{11}$$

α is the soft margin hyperparameter of the hinge-style triplet loss [5, 10] and we set $\alpha = 0.1$. All coordinate values are re-scaled into $[-1, 1]$. In this way, we reduce the computational cost from $O(N!)$ to $O(N^2 M)$, where $M = \max_{i=1}^{N} N_i$ is the maximum number of vertices of an element of $\mathbf{x}_0$. In practical implementation, the guidance training (Algorithm 1 of the main paper) uses $\hat{L}_{\text{perm}}(\phi)$ rather than $L_{\text{perm}}(\phi)$.

2.4 Diffusion models framework

Karras et al.[6] (EDM) presents a general diffusion model framework, where DDPM [4] and SDE-based DM formulations from Song et al.[12] can all be viewed as specializations of the proposed framework. We borrow its official codebase[4] to implement our GS-DM as it provides a general and clean base implementation suitable for all DM-based formulations. We then describe how we adapt the GS-DM into the EDM-based framework and list the relevant hyperparameter settings. This subsection follows the notations in Karras et al., where $\boldsymbol{y}$ is the data sample, $\boldsymbol{n}$ is the sampled Gaussian noise, $\boldsymbol{x}$ is the noisy sample, σ is the noise level (equivalent to the timestep), while $c_{\text{skip}}(\sigma)$, $c_{\text{in}}(\sigma)$, $c_{\text{out}}(\sigma)$, and $c_{\text{noise}}(\sigma)$ are the preconditioning factors [6, Section 5]. Similar to the notations of our main paper, we let $\boldsymbol{y}^i$ and $\boldsymbol{x}^i$ denote the i^{th} element of $\boldsymbol{y}$ and $\boldsymbol{x}$, respectively. The sensor condition is omitted for notation simplicity.

To adapt our GS-DM into the EDM framework, we first set $\sigma_{\text{data}} = 1.0$ for EDM [6, Table 1], and then adapt the preconditioning equation [6, Section 5, Eq.7] based on §3 of our main paper:

$$D_\theta(\boldsymbol{x}^i; \sigma, \boldsymbol{y}) = c_{\text{skip}}(\sigma)\, \boldsymbol{x}^i + c_{\text{out}}(\sigma)\, F_\theta^i\left(\left\{c_{\text{in}}(\sigma)\, \boldsymbol{x}^i + (1 - c_{\text{in}}(\sigma))\, \bar{\boldsymbol{\mu}}_\phi(\boldsymbol{y}, \sigma, i)\right\};\, c_{\text{noise}}(\sigma)\right), \tag{12}$$

where the per-element noise injection is defined as $\boldsymbol{x}^i = \boldsymbol{y}^i + \boldsymbol{n}\bar{\boldsymbol{\sigma}}_\phi(\boldsymbol{y}, \sigma, i)$, and noise $\boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$. $D_\theta(\boldsymbol{x}^i; \sigma, \boldsymbol{y})$ is the reconstructed $\boldsymbol{y}^i$. F_θ is the denoising network taking all elements of a noisy sample $\boldsymbol{x}$, and F_θ^i is the output of the i^{th} element. With the above modifications, we implement our GS-DM with the general EDM framework. We then describe the concrete hyperparameter settings [6, Table 1] for our two tasks. Please refer to Karras et al. for detailed explanations of each hyperparameter.

Floorplan reconstruction: For the guidance training, we set the $P_{\text{mean}} = 1.0$ and $P_{\text{std}} = 4.0$ to ensure sufficient coverage of the forward process. For the denoising training, we set the $P_{\text{mean}} = -0.5$ and $P_{\text{std}} = 1.5$. For inference (sampling), we set $\sigma_{\text{max}} = 5.0$ and $\sigma_{\text{min}} = 0.01$. Instead of the 2^{nd}

[4] `https://github.com/NVlabs/edm`

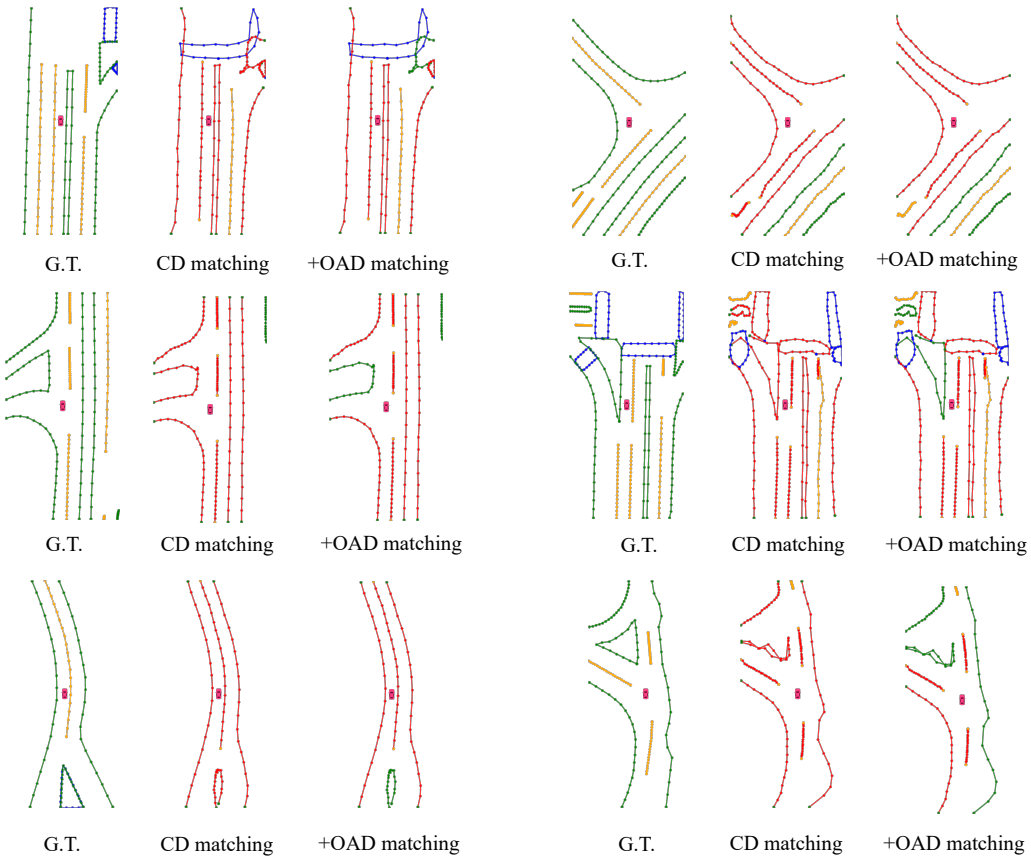

Figure 1: Illustration of the matching results with MapTR [7] predictions. **CD matching**: the original Chamfer distance (CD) matching criterion with a threshold of $1.0m$. **+OAD matching**: the original CD criterion with a threshold of $1.0m$, augmented with the order-aware angle distance (OAD) criterion with a threshold of $10°$. The matched instances (*i.e.*, true positives) are marked in red except for the two endpoints.

order Heun ODE solver adopted by the original EDM, we simply employ the 1^{st} order Euler solver for sampling, which is equivalent to DDIM [11]. All other hyperparameters are kept unchanged. All other dataset-specific settings are the same as previous works [2, 13, 15].

HD map reconstruction: The settings of P_{mean} and P_{std} for guidance and denoising training are the same as the floorplan reconstruction task. We set $c_{\text{skip}}(\sigma) = 0$ and $c_{\text{out}}(\sigma) = 1$ so that the denoising network directly estimates the sample $\boldsymbol{y}$, as we found this choice stabilizes the denoising training on the HD map construction task. For inference (sampling), we set $\sigma_{\text{max}} = 5.0$ and $\sigma_{\text{min}} = 0.1$, and use the Euler solver. All other hyperparameters are kept unchanged. All other dataset-specific settings are the same as previous works [7].

3 The augmented matching criterion for HD map reconstruction

In §5.2 of the main paper, we augment the Chamfer-distance (CD) matching criterion for the mean AP (mAP) metric used in previous works [7, 8] with an order-aware angle distance (OAD) to better evaluate the structural regularity and directional correctness of the results. This section first discusses the limitations of the original Chamfer-distance matching criterion and then provides complete implementation details of the order-aware angle distance.

3.1 The limitations of the Chamfer-distance matching criterion

Figure 1 shows examples of the Chamfer-distance (CD) matching (threshold is $1.0m$). Since CD considers each vertex separately, it mostly evaluates the global location of the instance and ignores the structural/directional information of the prediction. In the figure, some predictions with obviously wrong shapes are considered true positives under the CD matching criterion, while the augmented OAD matching criterion can effectively reject these bad shapes and align better with human judgment.

3.2 Implementation details

The computation of the order-aware angle distance is based on an optimal vertex-level matching between a predicted element (instance) and a ground truth (G.T.) element, similar to the "point-level matching" in the loss computation of MapTR [7]. Concretely, we enumerate through all equivalent representations of a G.T. map element and compute the average vertex-level $L1$ distance between the G.T. vertices and the predicted vertices. The variant with the smallest average vertex-level $L1$ distance forms the optimal matching with the predicted element. There are two equivalent variants for a polyline and $2(N_i - 1)$ equivalent variants for a polygon with N_i vertices. After obtaining the optimal one-to-one vertex-level matching, we trace along the vertices of the G.T. and predicted elements to compute their average angle distance.

With the augmented OAD matching criterion, we change the three-level thresholds of the original CD-based AP from $\{0.5m, 1.0m, 1.5m\}$ into $\{(0.5m, 5^\circ), (1.0m, 10^\circ), (1.5m, 15^\circ)\}$. However, compared to polylines (*i.e.*, road dividers and boundaries), we noticed that the angle direction of polygons (pedestrian crossings) is much more challenging to recover. With an OAD threshold of 5°, MapTR's average precision for the pedestrian crossing class (AP_p) is zero. Therefore, we loosen the OAD thresholds for the pedestrian crossing class by a factor of 2 (*i.e.*, $\{(0.5m, 10^\circ), (1.0m, 20^\circ), (1.5m, 30^\circ)\}$) while not changing the thresholds for the other two classes.

4 Additional experimental results

This section provides additional experimental results, including extra ablation studies and qualitative comparisons.

Table 1: Ablation study for the intra-element attention layer on the floorplan reconstruction task. Note that PolyDiffuse uses the RoomFormer from the first row to produce the initial reconstruction.

Evaluation Level →		Room			Corner			Angle		
Method	Intra-element-attn	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
RoomFormer	✗	96.3	96.2	96.2	89.7	86.7	88.2	85.4	82.5	83.9
RoomFormer	✓	96.9	96.4	96.6	90.3	87.0	88.6	84.9	81.9	83.4
PolyDiffuse(Ours)	✗	98.4	97.8	98.1	92.4	89.0	90.7	90.2	87.0	88.6
PolyDiffuse(Ours)	✓	98.7	98.1	98.4	92.8	89.3	91.0	90.8	87.4	89.1

Table 2: Ablation study for the intra-element attention layer on the HD map reconstruction task. Note that PolyDiffuse uses the MapTR from the first row to produce the initial reconstruction.

Matching Criterion →		Chamfer distance				+ Ordered angle distance			
Method	Intra-element-attn	AP_p	AP_d	AP_b	mAP	AP_p	AP_d	AP_b	mAP
MapTR	✗	55.8	60.9	61.1	59.3	46.1	43.4	41.9	43.8
MapTR	✓	56.5	59.9	60.2	58.8	48.6	44.8	41.6	45.0
PolyDiffuse(Ours)	✗	56.9	59.2	60.6	58.9	50.8	48.3	44.9	48.0
PolyDiffuse(Ours)	✓	58.2	59.7	61.3	59.7	52.0	49.5	45.4	49.0

4.1 Additional ablation studies

Table 1 presents the ablation study of the intra-element attention with the floorplan reconstruction task. Comparing the first and second rows, adding intra-element attention to RoomFormer slightly improves the room and corner results but deteriorates the angle-level performance. Table 2 provides a similar comparison for the HD map reconstruction task. Augmenting MapTR with intra-element attention worsens the mAP with the CD matching criterion while slightly improving the mAP with the OAD-augmented matching criterion. These comparisons provide a potential empirical explanation for the architecture choice of RoomFormer and MapTR – they only employ global attention layers without extra attention layers at the element or instance level.

On the contrary, as indicated by the third and fourth rows of both Table 1 and Table 2, applying intra-element attention consistently boosts PolyDiffuse across all metrics, although the improvements are not huge. A potential explanation here is that the denoising task of PolyDiffuse is more challenging than the regression/detection task of RoomFormer and MapTR, thus benefiting from extra modeling capacities.

4.2 A toy experiment with standard DM

In Figure 2, we provide a toy experiment to support what we motivated in §1 of the main paper, which demonstrates how standard DM easily fails even with a single data sample. Note that we have clarified the definition of standard DM in §5.3 of the main paper. In this experiment, the data contains a single toy sample with 6 rectangular shapes, so there are permutation-equivalent representations. After sufficient training, we draw four samples using the image-conditioned denoising process. The DDIM sampler is used with 10 sampling steps, so the randomness only comes from the initial noise. As the figure shows, only the third sample gets the correct reconstruction result. With the challenges of set ambiguity, a standard conditional DM has trouble overfitting a single data sample and easily gets wrong outputs when the initial noise is inappropriate.

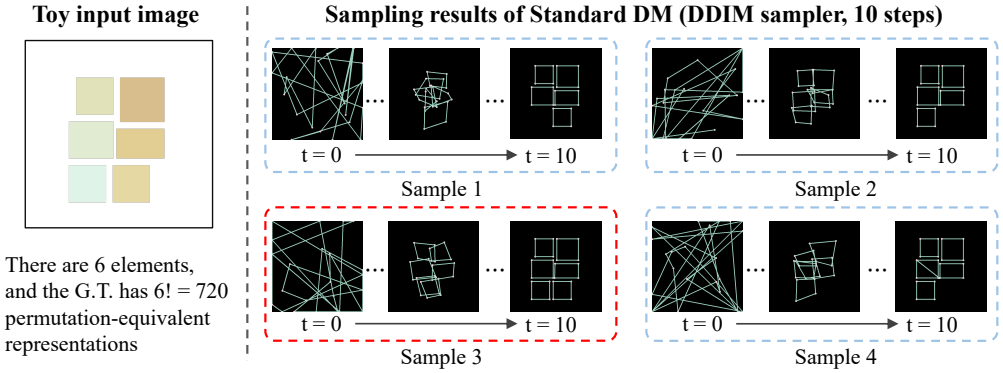

Figure 2: A simple toy experiment of using a standard DM to fit a single data sample with 6 elements. Four sampling results with *different initial noises* are shown. The DDIM sampler is used with 10 denoising steps. Only one of the four samples (*i.e.*, Sample 3) gets the correct final result due to the challenges induced by the set ambiguity, as explained in the main paper.

4.3 More qualitative examples

We provide additional qualitative results to compare PolyDiffuse against the state-of-the-art floorplan and HD map reconstruction methods, respectively. Note that the state-of-the-art method (*i.e.*, RoomFormer or MapTR) produces the initial reconstruction for PolyDiffuse. Figure 3 to Figure 5 are for the floorplan reconstruction task, while Figure 6 to Figure 8 are for the HD map reconstruction task.

Since the likelihood-based refinement is not employed in the qualitative examples of this subsection, PolyDiffuse cannot discover missing instances that are not covered by the proposal generator (*i.e.*, RoomFormer and MapTR in the qualitative results). However, the visual comparisons clearly demonstrate that PolyDiffuse significantly improves the structural regularity of the reconstructed polygonal shapes.

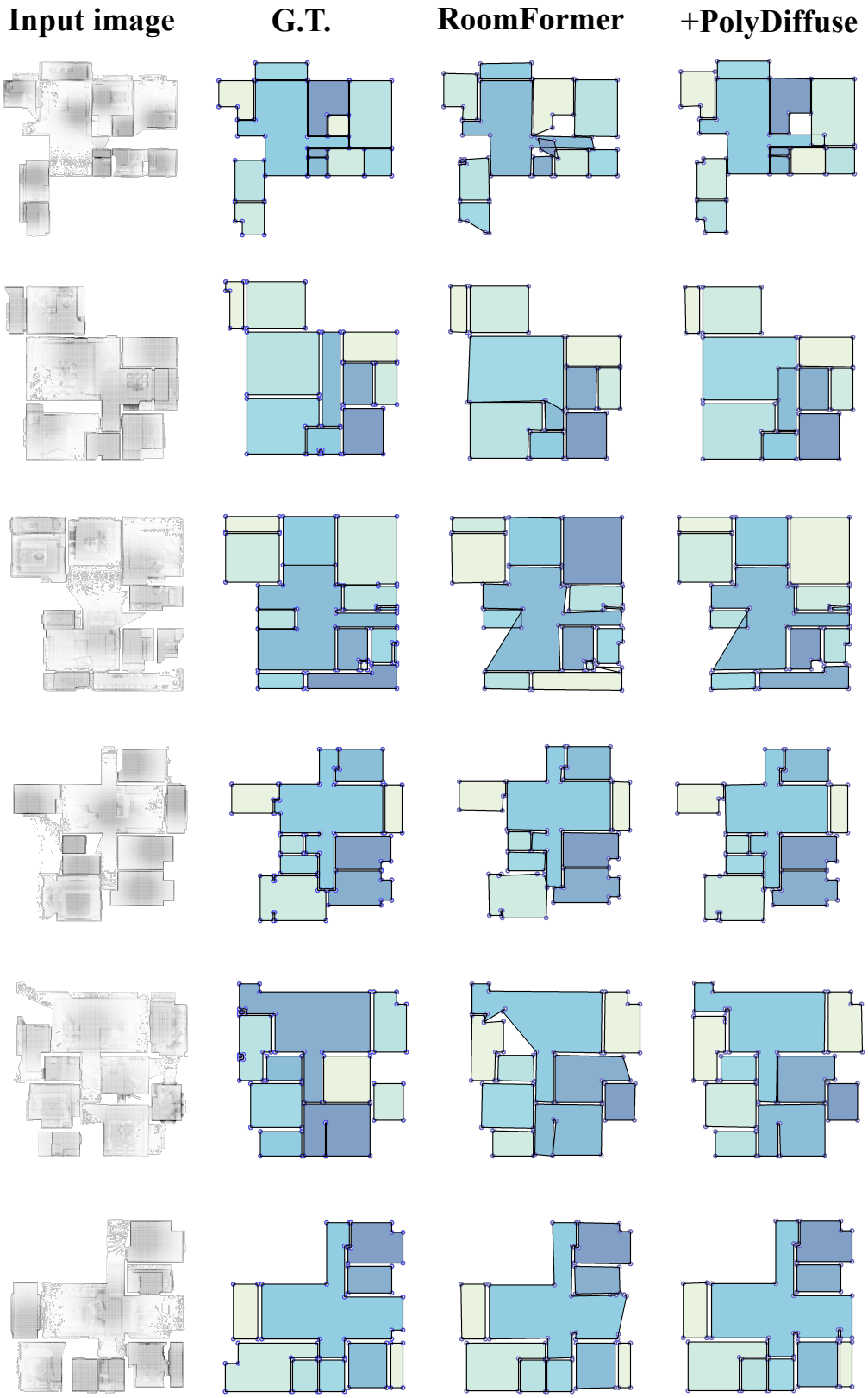

Figure 3: Additional qualitative results for the floorplan reconstruction task.

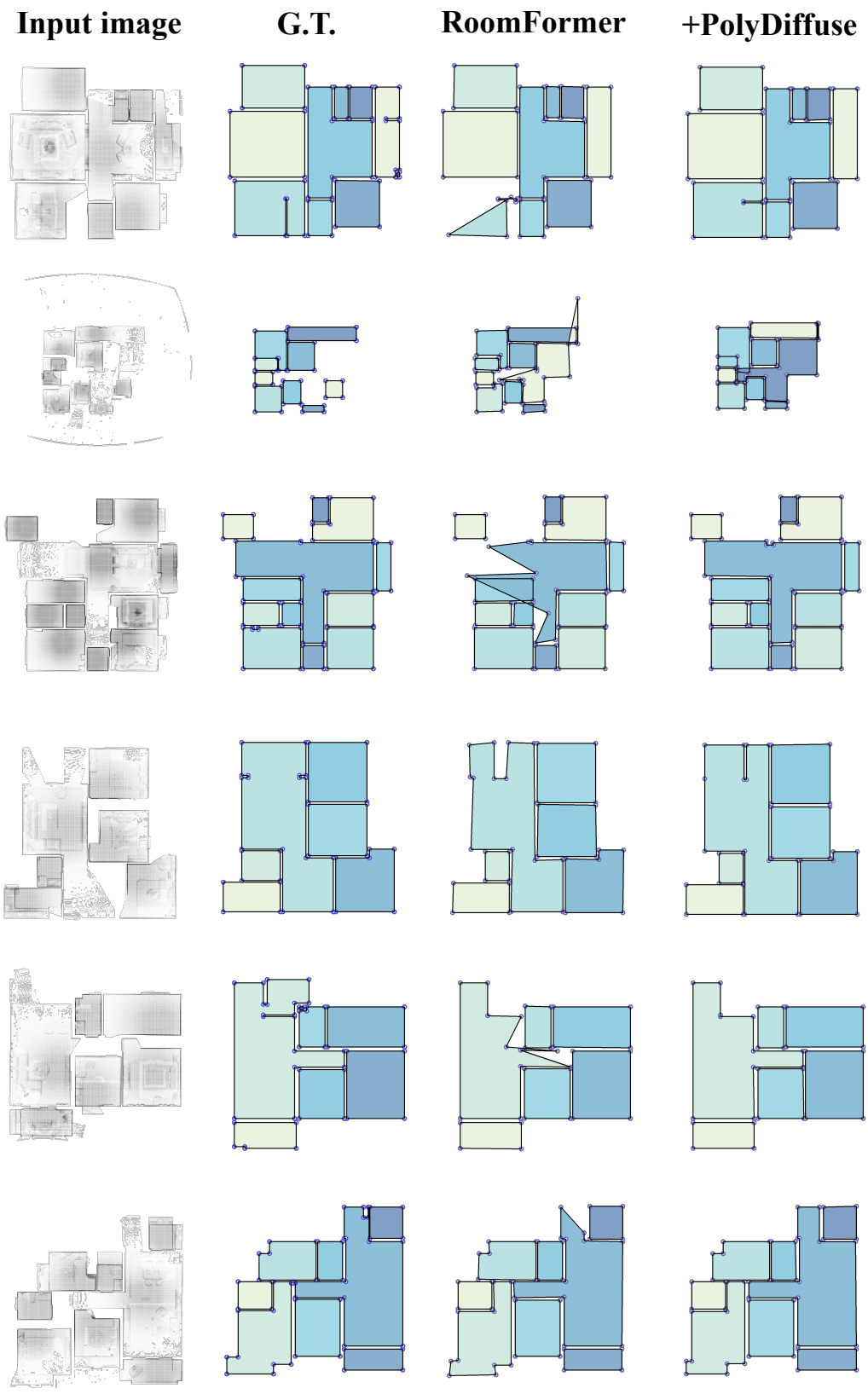

Figure 4: Additional qualitative results for the floorplan reconstruction task.

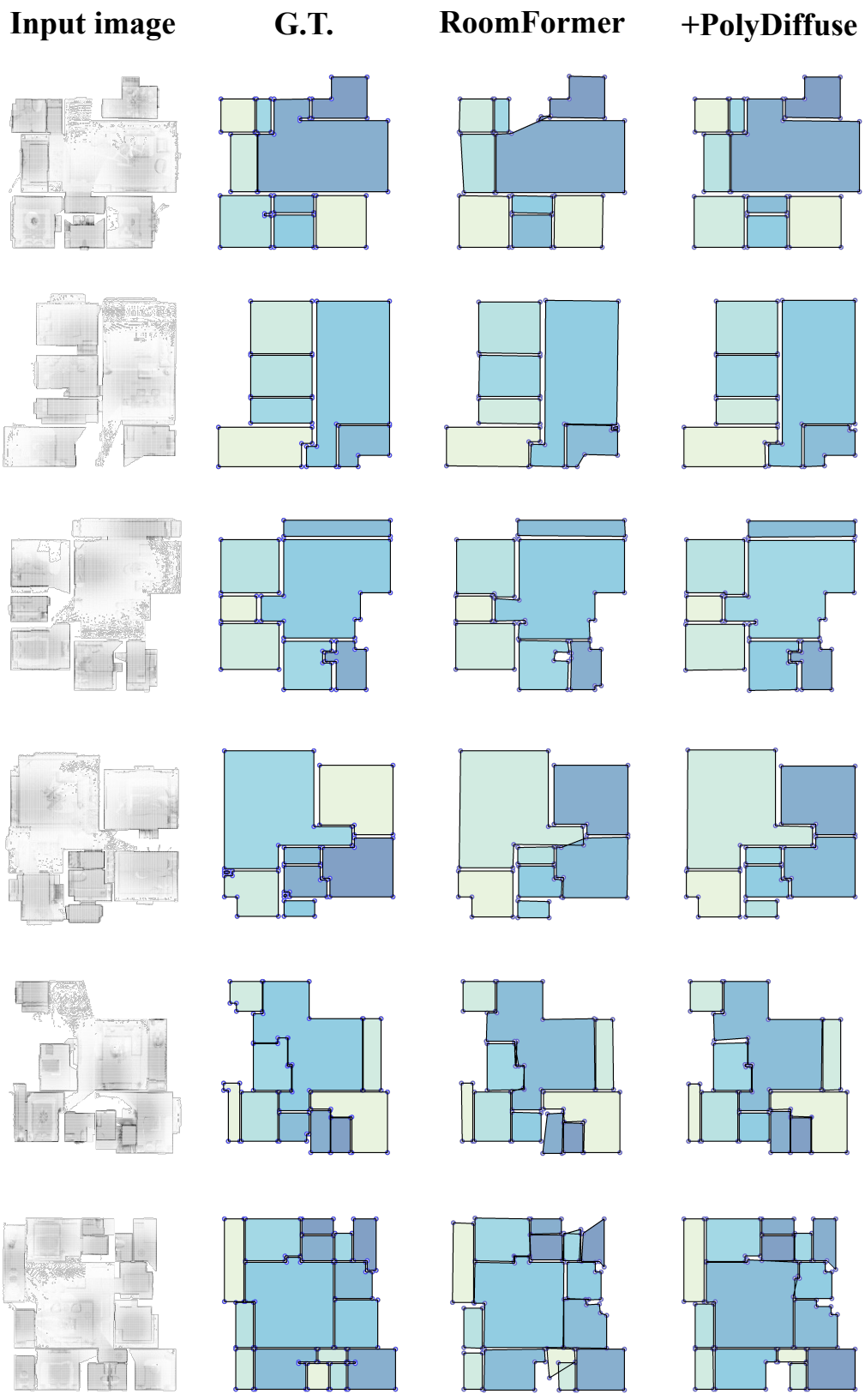

Figure 5: Additional qualitative results for the floorplan reconstruction task.

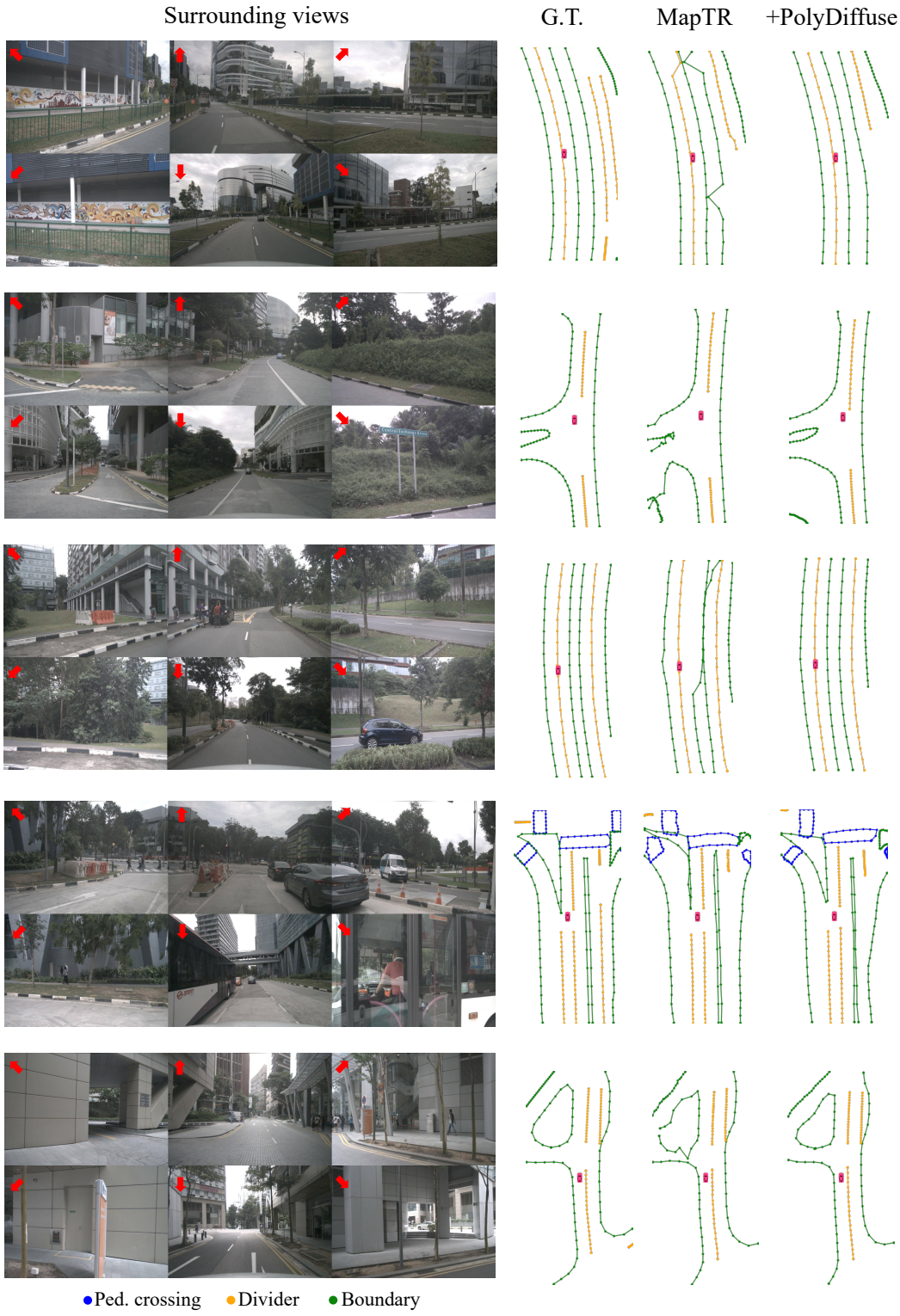

Figure 6: Additional qualitative results of the HD map reconstruction task.

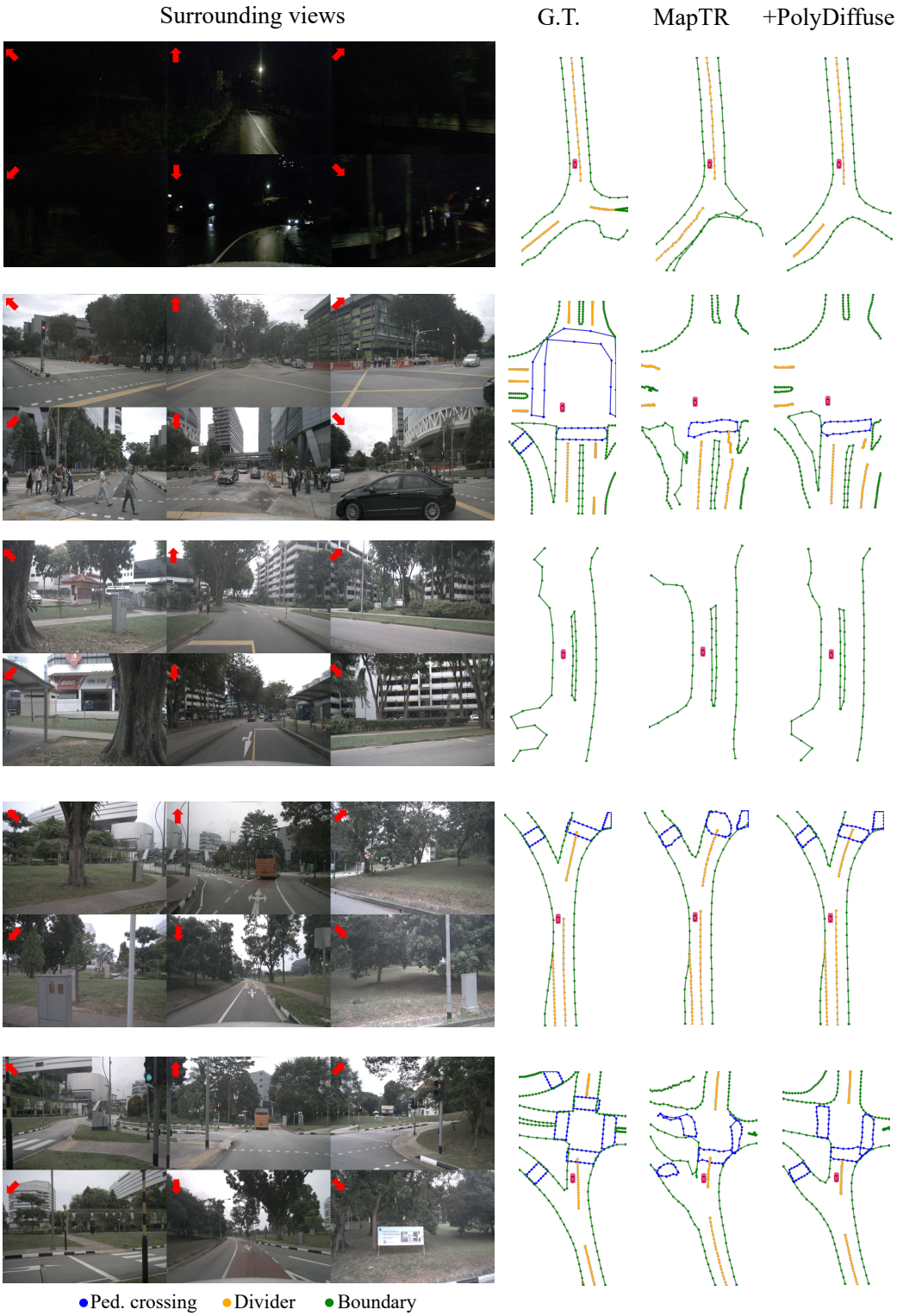

Figure 7: Additional qualitative results of the HD map reconstruction task.

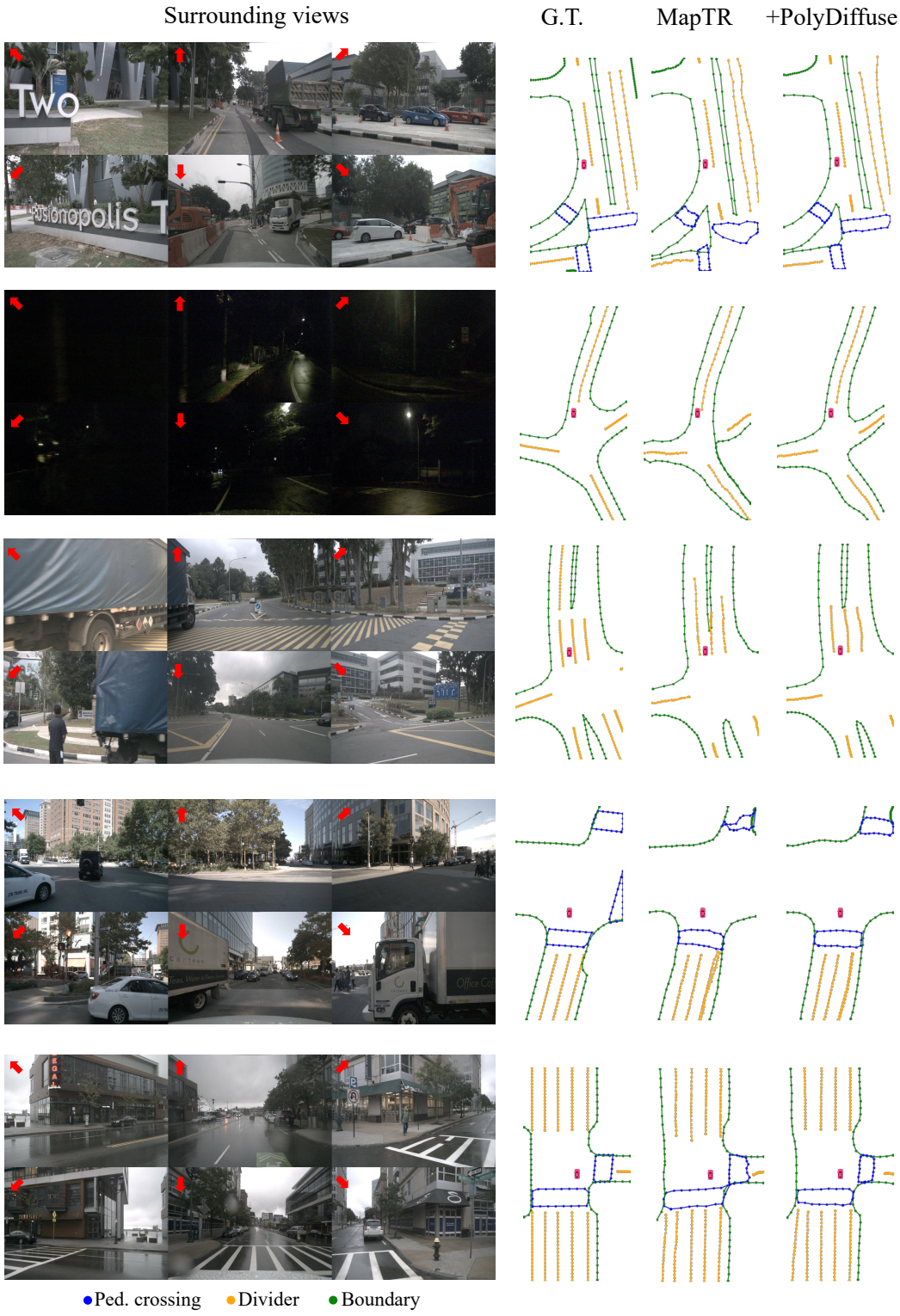

Figure 8: Additional qualitative results of the HD map reconstruction task.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *16th European Conference of Computer Vision (ECCV)*, 2020.

[2] Jiacheng Chen, Yiming Qian, and Yasutaka Furukawa. Heat: Holistic edge attention transformer for structured reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[5] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *Similarity-Based Pattern Recognition: Third International Workshop, SIMBAD 2015, Copenhagen, Denmark, October 12-14, 2015. Proceedings 3*. Springer, 2015.

[6] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[7] 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. *ICLR*, 2023.

[8] Yicheng Liu, Yue Wang, Yilun Wang, and Hang Zhao. Vectormapnet: End-to-end vectorized hd map learning. *ArXiv*, abs/2206.08920, 2022.

[9] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

[10] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. *Advances in neural information processing systems (NeurIPS)*, 2003.

[11] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *ArXiv*, abs/2010.02502, 2020.

[12] Yang Song, Jascha Narain Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *ICLR*, 2021.

[13] Sinisa Stekovic, Mahdi Rad, Friedrich Fraundorfer, and Vincent Lepetit. Montefloor: Extending mcts for reconstructing accurate large-scale floor plans. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems (NeurIPS)*, 2017.

[15] Yuanwen Yue, Theodora Kontogianni, Konrad Schindler, and Francis Engelmann. Connecting the dots: Floorplan reconstruction using two-level queries. *CVPR*, 2023.

[16] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.