DDIT: Semantic Scene Completion via Deformable Deep Implicit Templates (Supplementary Material)

Haoang Li[1,2,*] Jinhu Dong[3,*] Binghui Wen[1,*] Ming Gao[1,2,*]
Tianyu Huang[3] Yun-Hui Liu[3] Daniel Cremers[1,2,4]
[1]Technical University of Munich [2]Munich Center for Machine Learning (MCML)
[3]The Chinese University of Hong Kong [4]University of Oxford
{haoang.li, ming.gao, bh.wen, cremers}@tum.de {jhdong, tyhuang, yhliu}@mae.cuhk.edu.hk

Overview

In this supplementary material, we provide additional contents that are not included in the main paper due to the space limit:

- Establishment and use of our latent code-based hierarchical tree in Section 1.
- Additional information about our ScanARCW dataset in Section 2.
- Architecture of our neural network to predict latent codes in Section 3.
- Additional experimental results in Section 4.

1. Hierarchical Tree

Recall that in Section 4.2 of the main manuscript, we improve the efficiency of our transformation estimation based on a latent code-based hierarchical tree. In this section, we introduce how we establish and use such a tree.

Establishment of Tree. After pre-training deep implicit templates, CAD models from the same category are associated with respective pre-trained latent codes. We first cluster all these models based on K-Means algorithm, obtaining M clusters. We treat the centers of these clusters as M root nodes of the tree (see the first level of clustering in Fig. 1). Then we apply K-Means algorithm again to each cluster using a "tighter" threshold. For example, we divide the m-th cluster ($1 \leqslant m \leqslant M$) into N sub-clusters. We regard the centers of these sub-clusters as child nodes of the m-th root node, as shown in the second level in Fig. 1. We repeat the above procedures several times, i.e., a child node at the i-th level of clustering is a parent node at the $(i+1)$-th level. Intuitively, the cluster centers at the i-th level are more distinct than those at the $(i+1)$-th level. We terminate clustering when the latent codes from the same sub-cluster are similar enough.

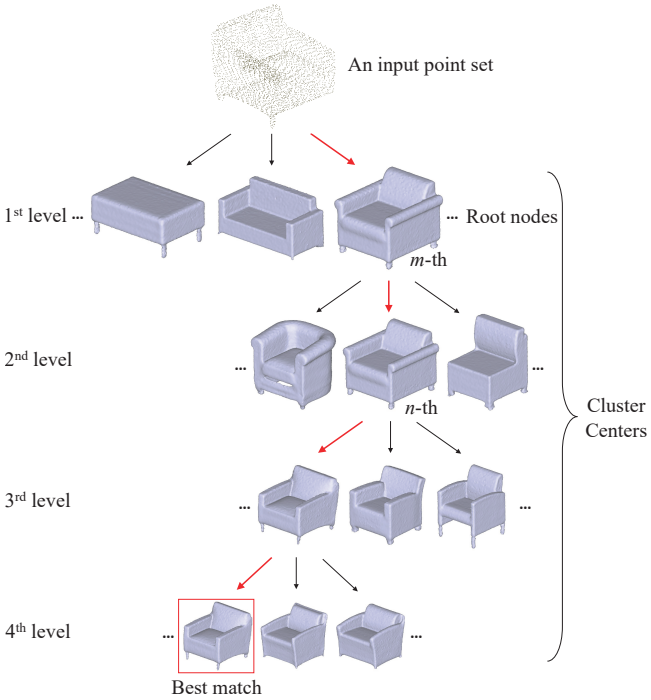


Figure 1. Illustration of our hierarchical tree based on the pre-trained latent codes. We take the category of sofa for example. Each tree node represents the center of a cluster. Red arrows represent our search path to find the best-matched node against the input point set.

Use of Tree. As mentioned in the main manuscript, given an incomplete point set in the world frame, we search for its best-matched CAD model along the above tree. We treat the transformation between this model and the input point set as the transformation of the input point set. Specifically, we first match the input point set against surface points of M root nodes at the first level. Without loss of generality, we

*Haoang Li, Jinhu Dong, Binghui Wen and Ming Gao contributed equally to this work.

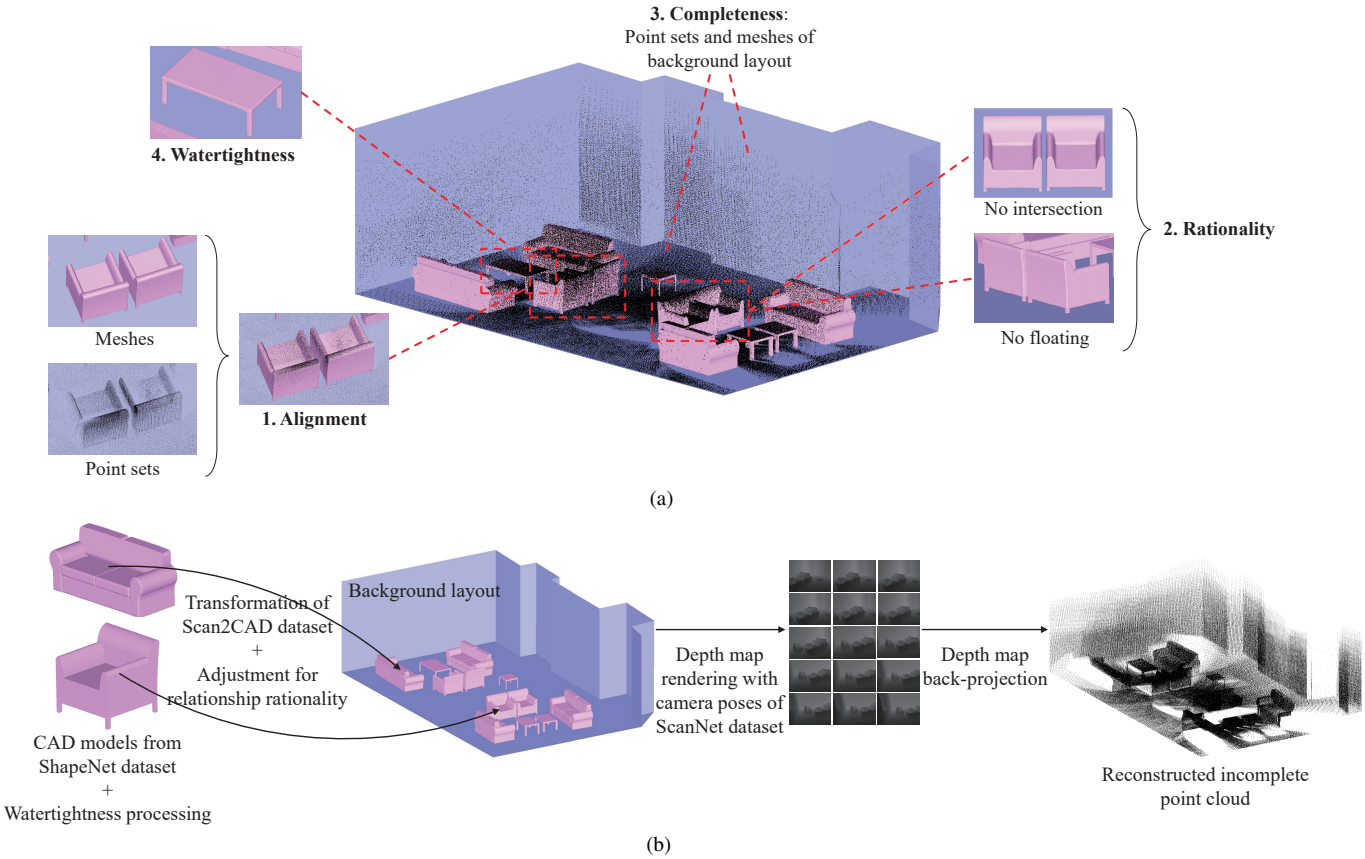


Figure 2. Illustration of our ScanARCW dataset. We take a representative scene for example (for visualization, we do not show ceiling and two walls). (a) Dataset characteristics. (b) Dataset establishment.

assume that the m-th root node achieves the highest inlier ratio. Then we match the input point set against N child nodes of the m-th root node, and neglect the child nodes of the other root nodes. Assume that the n-th child node achieves the highest inlier ratio at the second level. We thus continue our search from its child node. Intuitively, inlier ratio at a higher level (e.g., the fourth level) is generally higher than that at a lower level (e.g., the first level). The reason is that the similarity between the input point cloud and a candidate node becomes higher as the level increases. For example, in Fig. 1, the input point set and the m-th node at the first level only have similar overall shapes (both correspond to one-seat sofas). By contrast, the input point set and the best-matched node have not only close overall shapes but also similar details. The above coarse-to-fine search strategy avoids exhaustive search by the nodes at lower levels, and also leads to fine matches based on the nodes at higher levels.

2. Dataset

As introduced in Section 5 of the main manuscript, we establish ScanARCW dataset for semantic scene completion. As a complement, Fig. 2(a) illustrates the characteristics of our ScanARCW dataset, i.e., alignment, rationality, completeness, and watertightness. Fig. 2(b) shows the pipeline of our dataset establishment. We conduct watertightness processing based on ManifoldPlus [2], and render depth maps using Blender[1].

3. Network Architecture

As introduced in Section 4.1 of the main manuscript, our latent code prediction network consists of three sub-networks $\mathcal{N}_{\text{point}}$, $\mathcal{N}_{\text{intra}}$, and $\mathcal{N}_{\text{inter}}$. In the following, we introduce details.

Point Feature Extraction Network $\mathcal{N}_{\text{point}}$. Given a set of 3D points (cardinality is M), we first conduct 1D convolution to associate each point with an 8-dimensional feature. Then we apply EdgeConv to these point features, increasing their dimension from 8 to 16.

Intra-instance Feature Exaction Network $\mathcal{N}_{\text{intra}}$. We segment a point set into several patches at three different levels, i.e., 1024, 256, and 64 patches. A patch at the i-th level is composed of four sub-patches at the $(i-1)$-level. We call

[1] https://www.blender.org/

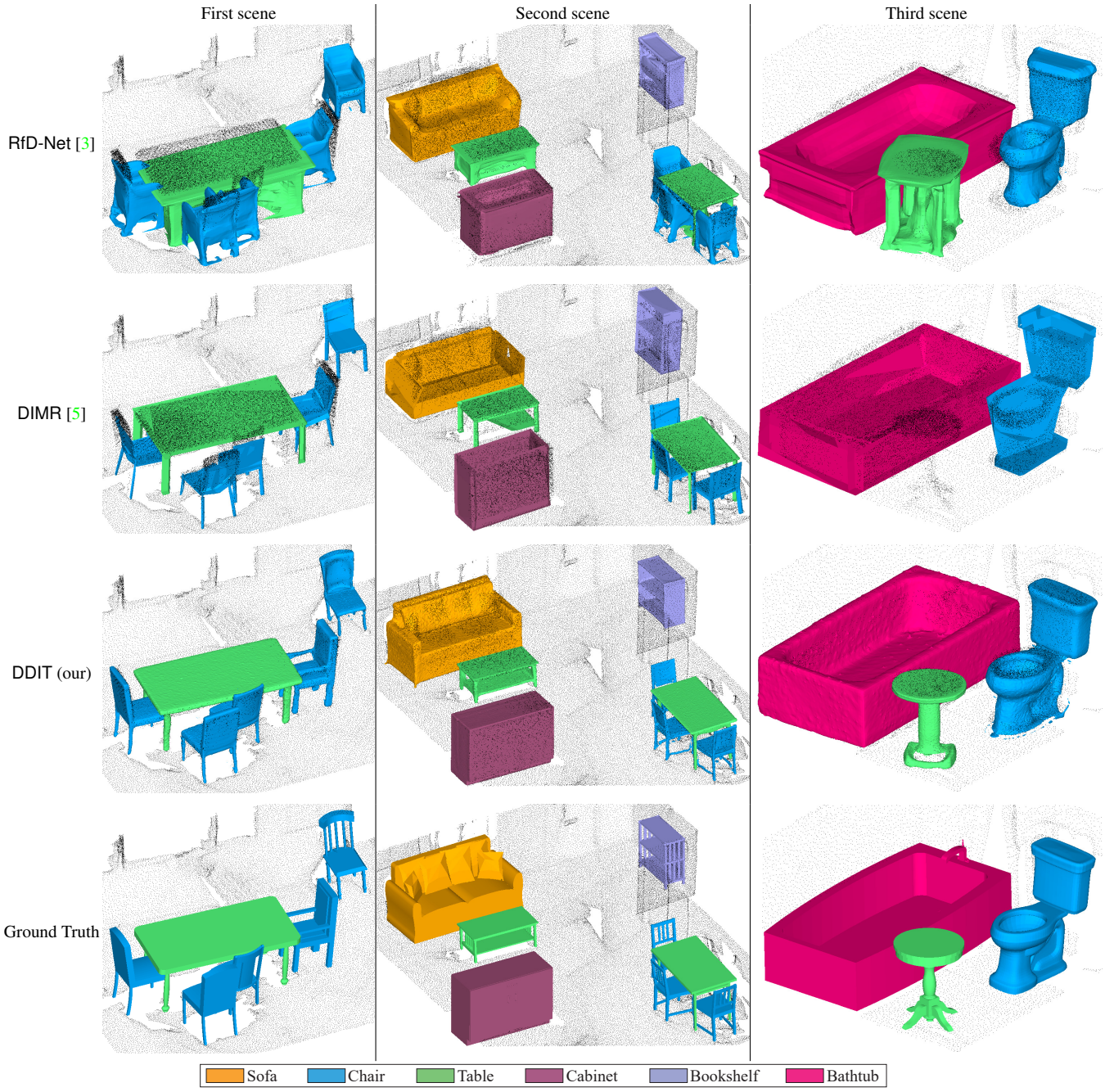
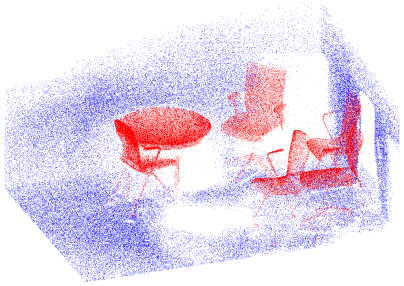


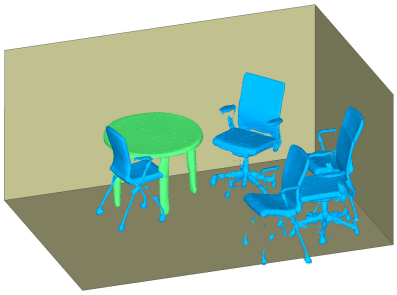
Figure 3. Additional qualitative comparisons with state-of-the-art methods in three representative scenes.

the center of a patch "control point". All the control points are from the above M points. We first consider 1024 control points at the first level. Each control point and its 8 nearest neighbors from the above M points define a graph. We apply edgeConv to this graph, and each control point is associated with a 32-dimensional feature.

Then we decrease the number of control points and also increase the dimension of point feature. Let us take the second level of segmentation (256 patches) for example. One patch $\mathcal{P}^{(2)}$ at the second level consists of four sub-patches $\{\mathcal{P}_i^{(1)}\}_{i=1}^4$ obtained in the first level of segmentation. Control points of four sub-patches define a three-edged graph, i.e., one control point is the root, and the remaining three are neighbors. We treat such a root as the control point of the patch $\mathcal{P}^{(2)}$. We apply EdgeConv to this graph, obtaining a 64-dimensional feature for the control point of the patch $\mathcal{P}^{(2)}$. We repeat the above procedures at the third level of segmentation.



Input point cloud



Our completion result

Figure 4. Qualitative result of our background and foreground completion in a representative scene (for visualization, we do not show ceiling and two walls). The segmented background and foreground point sets are shown in blue and red, respectively.

Table 1. Quantitative result of our background completion.

	PCR
Our method	94.83%

We update the above features of control points based on a set of transformer blocks (the number of blocks is 6 in our context). Then we employ 1D convolution to increase the dimension of each feature from 128 to 512. After that, we aggregate 64 features into a single feature by max pooling. Finally, we exploit MLP to map this 512-dimensional feature into a 256-dimensional code.

Inter-instance Feature Exaction Network $\mathcal{N}_{\text{inter}}$. We feed the above 256-dimensional latent codes of the i-th instance and its N neighbors to a multi-head attention module to obtain a new 256-dimensional feature (the number of heads is 10 in our context). As mentioned in the main manuscript, we do not directly treat this feature as the updated latent code. Instead, we concatenate this feature and the original code to obtain a 512-dimensional feature, followed by mapping it back to a 256-dimensional latent code based on MLP. The above data conversion can be summarized by $(1+N)\times 256 \rightarrow 1\times 256 \rightarrow 1\times 512 \rightarrow 1\times 256$.

4. Additional Experimental Results

We first provide additional comparisons with state-of-the-art methods in Section 4.1. Then we introduce our background completion in Section 4.2. After that, we show some unsatisfactory results of our method in Section 4.3, followed by presenting additional results of ablation study in Section 4.4.

4.1. Additional Comparisons

In Section 6.2 of the main manuscript, we compare our DDIT with state-of-the-art methods RfD-Net [3] and DIMR [5]. As shown in Fig. 3, we provide additional comparison results. For RfD-Net and DIMR, there is still room for accuracy improvement due to the lack of effective shape constraints. Our DDIT is more accurate than the above methods since deep implicit template provides constraints on the overall shapes and latent code guarantees fine details.

4.2. Background Completion

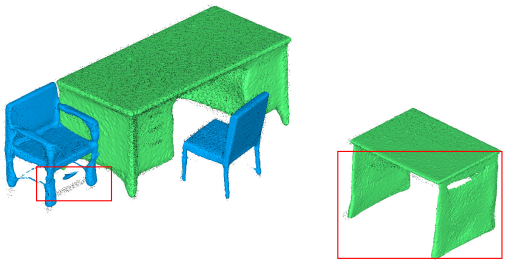
As mentioned in Section 4 of the manuscript, we also complete the background layout. Our completion consists of two main steps. First, given the segmented background point sets, we employ MLESAC [6], a variant of RANSAC to fit multiple planes. Second, we adjust some fitted planes to reduce the effects of noise. For one thing, we merge two over-fitted planes that have similar homogeneous coordinates. For another, we empirically find that some fitted walls may slightly incline due to insufficient observed points, while the fitted floor is more reliable. Accordingly, we enforce the Manhattan/Atlanta world assumption [1, 4] to make an inclined wall orthogonal to the floor. To quantitatively evaluate our background completion, we adopt the point coverage ratio (PCR) introduced in the main manuscript as the metric. Fig. 4 and Table 1 show that our completion is reliable.

4.3. Unsatisfactory Results of Our Method

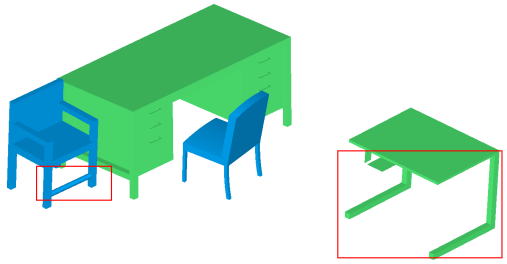
While our DDIT is generally accurate, it may still lead to unsatisfactory results in a small number of scenes. We empirically classify these results into two categories, i.e., inconsistent shapes and redundant parts.

Inconsistent Shapes. As shown in Fig. 5(a), the ground truth meshes show uncommon shapes, but our generated meshes exhibit regular shapes. The reason for such an inconsistency is that we pre-train a deep implicit template using the meshes with regular shapes. Our DDIT can hardly deform such a template into an uncommon shape. A potential solution is to incorporate CAD models with uncommon shapes for template training.

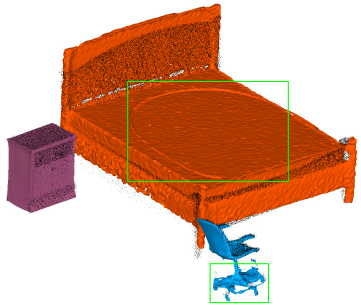
Redundant Parts. Fig. 5(b) shows that our generated meshes have some redundant parts. One reason is that our SDF value prediction inevitably results in some incorrect values. Another reason is that our estimated transformation may be affected by noise, and thus the input point cloud of our latent code prediction network is not strictly in the canonical frame. Accordingly, our estimated latent code



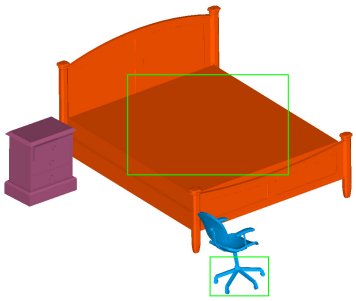
Generated meshes



Ground truth meshes
(a)



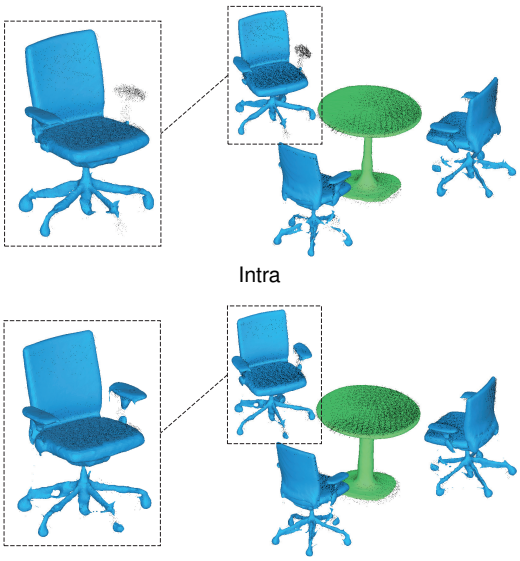
Generated meshes



Ground truth meshes
(b)

Figure 5. Representative unsatisfactory results of our DDIT. (a) Inconsistency between the ground truth and generated table legs and chair legs. (b) Redundant circular parts on the generated bed and planar parts on the generated chair.

is unreliable to some extent. We find that these redundant parts typically float around the main mesh and also have small volumes. Therefore, it is feasible to filter out these parts using a volume threshold.



Intra

Intra+Inter

Figure 6. Ablation study of our inter-instance information. We present a qualitative comparison in a representative scene.

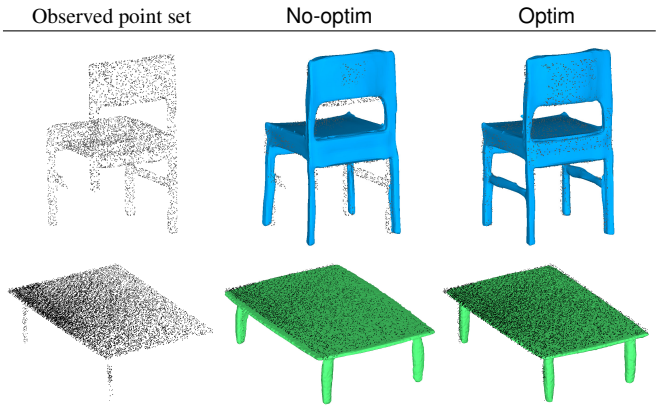


Figure 7. Ablation study of our optimization strategy. We present qualitative comparisons on two representative instances.

4.4. Ablation Study

Recall that in Section 6.3 of the main manuscript, we present ablation study of our inter-instance information and optimization strategy. Fig. 6 shows an additional test regarding inter-instance information. Intra+Inter using both intra- and inter-instance information provides more complete results than Intra using only intra-instance information. This result demonstrates the effectiveness of inter-instance information. As shown in Fig. 7, we present additional tests regarding optimization strategy. Compared with the original meshes denoted by No-optim, the optimized meshes denoted by Optim are better aligned to the observed point sets.

References

[1] James Coughlan and Alan Yuille. Manhattan world: Compass direction from a single image by Bayesian inference. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 941–947, 1999. 4

[2] Jingwei Huang, Yichao Zhou, and Leonidas Guibas. ManifoldPlus: A robust and scalable watertight manifold surface generation method for triangle soups. *arXiv preprint arXiv:2005.11621*, 2020. 2

[3] Yinyu Nie, Ji Hou, Xiaoguang Han, and Matthias Niessner. RfD-Net: Point scene understanding by semantic instance reconstruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4608–4618, 2021. 3, 4

[4] Grant Schindler and Frank Dellaert. Atlanta world: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 203–209, 2004. 4

[5] Jiaxiang Tang, Xiaokang Chen, Jingbo Wang, and Gang Zeng. Point scene understanding via disentangled instance mesh reconstruction. In *European Conference on Computer Vision (ECCV)*, pages 684–701, 2022. 3, 4

[6] Philip Torr and Andrew Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer vision and image understanding (CVIU)*, 78(1):138–156, 2000. 4