Parametric Point Cloud Completion for Polygonal Surface Reconstruction

Supplementary Material

Our supplementary materials provide visual demonstrations (Sec. A), instructions for reproducing our results (Sec. B), detailed implementation insights (Sec. C), and extended experimental analyses that augment the findings in the main paper (Sec. D).

A. Video

The accompanying video, available on our project page[1], highlights the motivation behind our work, illustrates its key ideas, and demonstrates the reconstruction results.

B. Reproducibility

The code and demo for our method are available in a public GitHub repository linked from the same project page[1]. Detailed instructions for setting up the environment and running the demo are provided in the `README.md` file.

C. Implementation Details

C.1. Dataset

We curated polygonal surface meshes from the ABC dataset [20] to establish a consistent benchmark for evaluating polygonal surface reconstruction. We include only objects composed entirely of planar surfaces, excluding those with multiple parts or non-watertight geometries. The resulting dataset comprises 15,339 CAD models. The distribution of face counts is shown in Fig. S1. Each input point cloud contains 2,048 points, while the ground truth point cloud comprises 8,192 points. For a fair comparison with voxel-based methods like BSP-Net [6] and SECAD-Net [23] (Sec. 4.3), the input point clouds are aligned with voxel representations. This alignment involves masking voxels based on their visibility from a selected viewpoint, excluding occluded voxels. Fig. S2 illustrates this alignment between point clouds and their corresponding voxel representation.

C.2. Handling Failure Cases

In cases of reconstruction failure (*e.g.*, no geometry generated), metrics are calculated against a bounding box with a diagonal length of 1. This ensures a fair evaluation by penalizing failures in a controlled manner, mitigating potential bias. Tab. S1 presents the results of a sanity check where all samples are treated as failures.

[1] https://parametric-completion.github.io

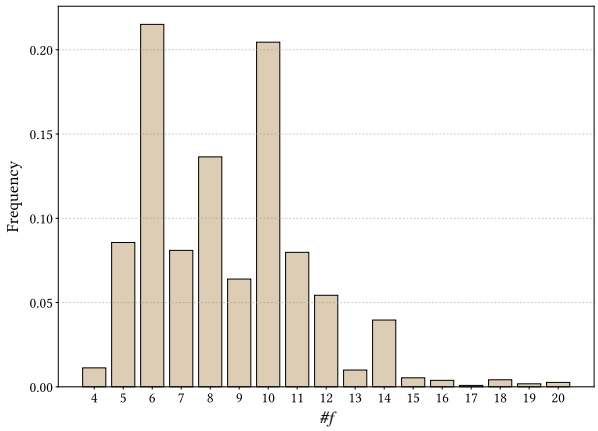

Figure S1. **Face count distribution in the dataset.**

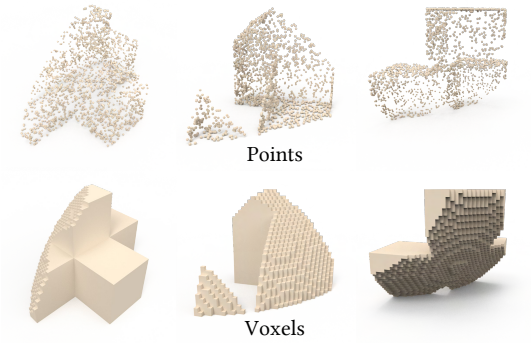

Figure S2. **Alignment between points and voxels.** This alignment ensures consistent representations of incomplete data.

Table S1. **Sanity check results.** "San." refers to cases where the normalized bounding box is used as the output geometry, ensuring unbiased metric calculation.

Method	CD ↓	HD ↓	NC ↑
San.	16.36	22.06	0.550
PaCo (ours)	1.87	4.09	0.943

C.3. Hyperparameters

PaCo is implemented in PyTorch and optimized using the AdamW optimizer with an initial learning rate of 10^{-4}, a weight decay of 5×10^{-4}, and a learning rate decay of 0.9 every 20 epochs. The encoder and decoder depths are set to 8 and 12, respectively. For ablation studies (Sec. 4.6), the encoder depth is set to 6 and the decoder depth to 8 to reduce computational complexity. The encoder produces 128 point proxies, and the number of plane proxies K after padding

is set to 20. A total of 40 queries (M) are used. During inference, we select primitives with confidence $\kappa_{\hat{\sigma}(i)} > 0.7$. For the loss terms in Eq. (10), we empirically set $\beta_2 = \beta_4 = 20$, and $\beta_3 = 2$. To address class imbalance, we set

$$\beta_1 = \begin{cases} 0.4, & \text{if } c_i = \emptyset, \\ 1, & \text{otherwise.} \end{cases} \tag{S1}$$

All the competing methods [4, 6, 8, 14, 23, 40, 42, 46–48] and reconstruction solvers [1, 26, 32] are used with their default settings.

D. Additional Analysis

D.1. Evaluation on Primitive Parameters

In the main paper, we use normal consistency (NC) as one of the metrics to evaluate the quality of the reconstruction results. The NC is calculated from surface-sampled points and is an indirect indicator of the accuracy of the primitive parameters recovered by parametric completion. To complement the indirect surface-based evaluation (see Sec. 4.1), we introduce a direct metric, NC_{prim}, defined as the average normal consistency between the predicted and ground truth primitives, to quantify the accuracy of the recovered primitive parameters. For the competing methods, GoCoPP [45] is used to extract primitives from the completed points. As shown in Tab. S2, our method achieves the highest primitive normal consistency ($\text{NC}_{\text{prim}} = 0.976$) among all methods.

Table S2. **Evaluation on primitive normal consistency.** PaCo produces planar primitives with the highest NC_{prim}.

Method	$\text{NC}_{\text{prim}} \uparrow$
PCN [48]	0.605
FoldingNet [42]	0.849
GRNet [40]	0.752
PoinTr [46]	0.863
AdaPoinTr [47]	0.930
ODGNet [4]	0.933
PaCo (ours)	**0.976**

D.2. Surface Complexity

Fig. S3 presents the performance of our method with respect to surface complexity. As the geometric complexity increases, CD tends to rise while NC declines. This trend arises from the inherent challenges of learning complex structures and the underrepresentation of highly complex samples in the dataset (see Fig. S1).

D.3. Point Density

Since competing methods often rely on point density for improved metrics, we normalize all outputs by down-sampling or up-sampling to a consistent total of 8,192 points. This ensures a fair comparison and prevents metric variations from being attributed to point count differences. Tab. S3 shows that this normalization has a negligible impact on performance for PaCo, demonstrating its robustness in recovering accurate parametric representations independent of point density.

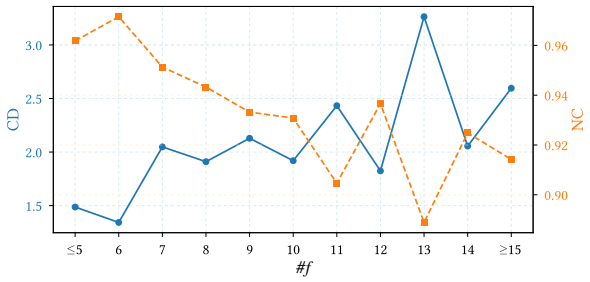

Figure S3. **Performance concerning surface complexity.** As the complexity increases, CD tends to rise while NC declines.

Table S3. **Impact of number of points for PaCo.** "S" indicates sampling to 8,192 points. The number of points has no significant impact on the reconstruction.

Solver	S	CD ↓	HD ↓	NC ↑	FR ↓
KSR [1]		1.91	4.14	0.940	0.25
	✓	1.92	4.19	0.945	0.28
COMPOD [32]		1.94	4.42	0.940	0.25
	✓	1.95	4.56	0.943	0.34
PolyFit [26]		1.87	4.09	0.943	0.48
	✓	1.93	4.48	0.941	0.63

D.4. Number of Proxies

We set the number of proxies to be greater than the maximum number of faces of the samples in the dataset. This surplus allows the primitive selector to distinguish between positive and negative primitives more effectively. Tab. S4 presents the results using different numbers of plane proxies, where 40 proxies achieve the best performance. The optimal number of proxies is likely correlated to the distribution of face counts (see Fig. S1).

Table S4. **Performance with different proxy numbers.** Our method performs the best with 40 proxies.

Num. Proxies	CD ↓	HD ↓	NC ↑
30	2.75	6.98	0.906
40	**2.11**	**5.26**	**0.941**
50	2.64	6.86	0.908

D.5. Evaluation on Real Data

We evaluate PaCo on real airborne LiDAR point clouds for building reconstruction to assess its transferability (Fig. S4). Although PaCo was trained on the ABC dataset [20] and fine-tuned with only 2,000 airborne LiDAR samples[2], it successfully reconstructs heavily occluded structures without relying on explicit rules. This demonstrates its robustness to noise and outliers and, notably, its ability to approximate freeform surfaces using planar primitives.

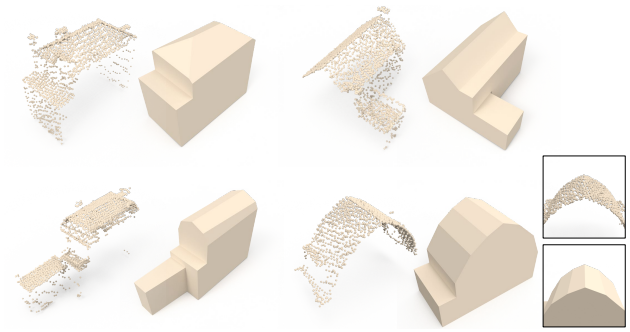

Figure S4. **Buildings reconstructed from real-world airborne LiDAR data with noise and outliers.** The fine-tuned PaCo model generalizes to unseen airborne LiDAR data.

D.6. Inference Speed

Fig. S5 presents the inference latency comparisons across different completion methods. PaCo achieves an average inference time of 29.8 ms, nearly twice as fast as the strongest competitor, ODGNet (53.6 ms), while reducing CD by 32%. Notably, bipartite matching applies only during training and does not affect inference.

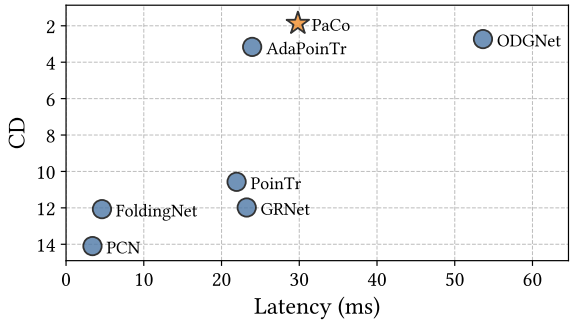

Figure S5. **Chamfer distance (CD) vs. inference latency.** CD is computed on meshes via the PolyFit solver, and latency is measured on an NVIDIA A40 GPU (excluding I/O).

[2] https://www.ahn.nl/