Unified Primitive Proxies for Structured Shape Completion

Supplementary Material

In the appendix, we provide instructions for reproducing our results (Sec. A), detailed implementation settings (Sec. B), extended experimental analyses (Sec. C), and further details on the datasets (Sec. D) and metrics (Sec. E).

A. Reproducibility

The code repository and demo are publicly accessible via the project page[1]. Detailed instructions for setup and running the code are described in the repository's `README.md` file.

B. Implementation Details

The point pathway follows the AdaPoinTr backbone [70] with its default depth and hyperparameters. Input points are grouped into local neighborhoods, encoded with self-attention, and decoded from learned point queries. The decoder produces $U = 512$ shape features, which a lightweight reconstruction head expands into 8,192 completed points. As in AdaPoinTr, we use denoising queries during training for an auxiliary denoising loss and drop them at inference. The primitive pathway consumes the same U shape features. We use $K = 40$ learnable primitive queries, contextualized by a 4-layer Transformer decoder with 8 attention heads and a hidden size of 128. For each query, a prediction head outputs a primitive type, a soft mask over the 512 shape features, and 10 coefficients of a homogeneous quadric.

UniCo is implemented in PyTorch and optimized using the AdamW optimizer with an initial learning rate of 2×10^{-3}, a weight decay of 5×10^{-4}, and a learning rate decay of 0.9 every 20 epochs. During inference, primitives with confidence $s_k > 0.5$ are retained. For the loss terms in Eq. (12), we empirically set $\alpha_2 = 0.125$, $\alpha_3 = 1$ and $\lambda = 0.05$. To mitigate class imbalance, we set

$$\alpha_1 = \begin{cases} 0.05, & \text{if } c_g \neq \emptyset, \\ 0.01, & \text{otherwise.} \end{cases} \tag{S1}$$

where c_g denotes the ground-truth primitive type. On *ABC-multi*, shapes contain on average about 8 primitives while we use $K = 40$ proxies, so the ratio for no-object *vs*. valid types roughly matches the expected proportion and keeps their aggregate loss contributions comparable. The impact of key hyperparameters is analyzed in Sec. C.

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

C. Additional Analyses

C.1. More Ablations

Number of primitive proxies. We vary the number of primitive proxies $K \in \{30, 40, 50\}$ and the no-object weight $\alpha_1(c_g = \emptyset)$ while keeping all other settings fixed. As reported in Tab. S1, $K = 40$ with a no-object weight of 0.01 achieves the best results.

Table S1. **Effect of proxy count and no-object weight.** Parameter counts (in millions) only include the primitive pathway $f_{\text{primitive}}$.

K	$\alpha_1(c_g = \emptyset)$	Params (M)	CD ↓	HD ↓	NC ↑	FR ↓
30	0.01	0.901	2.47	8.70	0.923	1.98
40	0.01	0.902	2.44	8.80	0.924	1.83
40	0.05	0.902	2.73	9.60	0.915	1.96
50	0.01	0.904	2.48	8.94	0.920	1.59

Confidence threshold. We assess the effect of the confidence threshold applied to the scores s_k in Eq. (14), varying the pruning value in $\{0.3, 0.5, 0.7\}$ at inference time. As summarized in Tab. S2, performance is fairly stable across thresholds, and $s_k > 0.5$ gives the best overall results.

Table S2. **Effect of confidence threshold.** Changing the pruning cutoff on s_k within a reasonable range has little effect on performance, while using $s_k > 0.5$ gives the best results.

$s_k >$	CD ↓	HD ↓	NC ↑	FR ↓
0.3	2.48	8.81	0.923	1.89
0.5	2.44	8.80	0.924	1.83
0.7	2.48	8.84	0.923	1.80

Analytic *vs*. fitted primitives. In Tab. S3, we compare primitives obtained directly from analytic quadric parameters with primitives obtained by fitting quadrics to the completed points. Since the assignments are identical, F1 and type accuracy remain unchanged. Analytic parameters achieve a lower axis error, whereas fitted primitives reduce the residual error and slightly improve coverage. For consistency with competing methods that rely on fitted primitives, the main paper reports the fitted variant.

Projection. Tab. S4 compares UniCo with and without a projection-based post-processing step that projects completed points onto their predicted primitives to enforce stricter planar geometry, with PaCo [10] as a reference. On both *ABC-plane* and *Building-PCC*, UniCo without projection already improves over PaCo, and projection brings only modest ad-

Source	F1 ↑	Type ↑	Axis ↓	Res ↓	Cov ↑
Analytic	0.712	94.85	2.71	0.70	92.16
Fitted	0.712	94.85	3.29	0.55	92.41

Table S3. **Primitive quality for analytic *vs*. fitted sources.** Fitting quadrics to completed points improves residual error and coverage at a cost in axis error.

ditional gains. For fairness, the main paper therefore reports UniCo without projection and treats the projected variant as an optional refinement for slightly sharper surfaces. Fig. S1 shows that PaCo tends to under-represent small primitives and fine details, whereas UniCo produces more uniform point distributions and cleaner, well-aligned primitive layouts across scales. Projection mainly sharpens surfaces and does not change this overall qualitative picture.

Table S4. **Effect of projection-based refinement.** UniCo already outperforms PaCo on *ABC-plane* and *Building-PCC*, and projection yields only marginal gains.

(a) *ABC-plane*

Method	CD ↓	HD ↓	NC ↑	FR ↓
PaCo [10]	1.87	4.09	0.943	0.48
UniCo	1.69	4.28	0.953	0.69
UniCo (proj.)	1.67	4.29	0.955	0.55

(b) *Building-PCC*

Method	CD ↓	HD ↓	NC ↑	FR ↓
PaCo [10]	4.89	10.74	0.932	0.54
UniCo	3.84	9.18	0.949	0.39
UniCo (proj.)	3.83	9.06	0.949	0.17

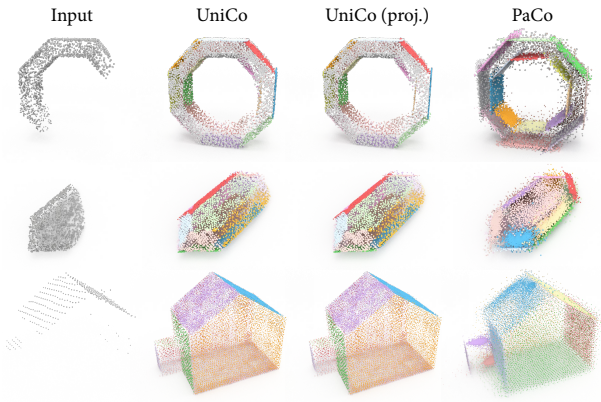

Figure S1. Qualitative primitive comparison on *ABC-plane* and *Building-PCC*. UniCo recovers more uniform point distributions and cleaner, well-aligned primitive structures.

Robustness across backbones. We additionally swap the point completion backbone f_{point} to PoinTr [69] or ODGNet [3]. UniCo retains *consistent* gains over the corresponding completion+extractor baselines. This shows that the improvements are driven by UniCo's primitive pathway and persist across backbones.

Table S5. **Robustness across backbones.** UniCo retains consistent gains over the corresponding completion+extractor baselines.

Method	f_{point} = PoinTr [69]			f_{point} = ODGNet [3]		
	CD↓	NC↑	FR↓	CD↓	NC↑	FR↓
Compl.+Extr.	6.58	0.791	11.27	4.33	0.873	7.41
UniCo (ours)	2.64	0.915	2.49	2.47	0.922	0.31

C.2. Transferability

We train UniCo separately on the *ABC-multi* and *ABC-plane* and evaluate both on the plane-only split. To avoid data leakage, we exclude test shapes that also appear in the mixed-type training set. As shown in Tab. S6, the model trained on the mixed-type split exhibits only moderate degradation across all metrics and still achieves strong performance, indicating that UniCo transfers well from mixed-type to plane-only data.

Table S6. **Transferability.** UniCo trained on different datasets and evaluated on *ABC-plane*, excluding overlapping samples.

Training → Evaluation	CD ↓	HD ↓	NC ↑	FR ↓
ABC-plane → ABC-plane	1.70	4.35	0.953	1.16
ABC-multi → ABC-plane	2.06	5.36	0.942	2.13

Moreover, UniCo *generalizes* to unseen freeform geometries in *real* OmniObject3D scans [58] *without* training on them, by compactly approximating them with the available primitives, as shown in Fig. S2.

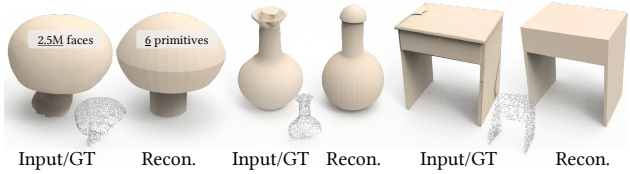

Figure S2. **OOD generalization via approximation.** UniCo trained on synthetic data with a fixed primitive set generalizes to real-world point clouds with unseen geometry by approximation.

C.3. Computational Efficiency

Tab. S7 reports per-scan runtimes on *ABC-plane*. UniCo processes each partial input in 27.6 ms end-to-end, roughly twice as fast as the strongest pointwise competitor, ODGNet [3], and faster than the structured competitor PaCo [10]. All other completion methods, except PaCo and UniCo, require an additional primitive extraction stage. To quantify this overhead, we evaluate RANSAC [42], HPNet [64], and PTv3 [59] on 100 randomly selected ABC-multi samples completed by ODGNet. UniCo is the fastest end-to-end pipeline.

Table S7. **Runtime and complexity.** All completion methods are measured on a single A40 GPU, excluding the first iteration to ensure steady-state measurements. For methods requiring primitive extraction, we additionally report RANSAC, HPNet, and PTv3 post-processing costs, evaluated on 100 random *ABC-multi* samples. "Total Params" and "Total Latency" refer to the parameter count and latency of the full pipeline. Params are reported in millions and all times are in milliseconds.

Method	Params	Latency	RANSAC [42]		HPNet [64]		PTv3 [59]	
			Total Params	Total Latency	Total Params	Total Latency	Total Params	Total Latency
AdaPoinTr [70]	32.5	23.9	32.5	124.9	33.8	1280.9	208.5	169.3
ODGNet [3]	**11.5**	53.6	**11.5**	154.6	**12.8**	1310.6	187.5	199.0
SymmComplete [63]	13.3	**20.0**	13.3	121.0	14.6	1277.0	189.4	165.4
PaCo [10]	41.4	29.8	41.4	29.8	41.4	29.8	41.4	29.8
UniCo (ours)	33.4	27.6	33.4	**27.6**	33.4	**27.6**	**33.4**	**27.6**

C.4. Failure cases

Typical failure modes involve missing small primitives and misclassified semantics, with examples shown in Fig. S3.

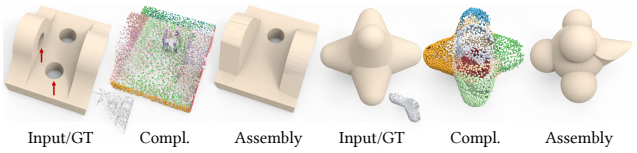

Figure S3. **Failure cases.** Typical failure cases involve missing small primitives and misclassified semantics.

D. Datasets

For *ABC-multi*, we randomly select 30,000 single-piece watertight CAD models from the ABC dataset [24], covering a broad spectrum of primitive configurations, from simple shapes with only a few primitives to complex assemblies with several tens of parts. Planes and cylinders dominate the primitive inventory, while cones and spheres provide additional geometric variety. This yields a diverse, structured benchmark for primitive-based completion. A quantitative summary is provided in Tab. S8.

In addition, we use *ABC-plane* [10], a plane-only subset of ABC, and *Building-PCC* [15], an airborne LiDAR dataset of roughly 50,000 urban buildings with noise and occlusions. These datasets complement *ABC-multi* by providing plane-only CAD assemblies and real-world scans for evaluation.

E. Metrics

For failed reconstructions, we follow the established protocol [10] and evaluate against the unit cube. Below we detail the additional primitive-level metrics used in Tab. 3, following established practice [16, 30, 36]. We denote a matched primitive pair by $(k^*, g^*) \in \mathcal{M}$:

Table S8. **Primitive statistics on *ABC-multi*.** Primitive counts, per-sample statistics, and type compositions for 30,000 CAD models.

Statistic	**Value**
Samples	30,000
Primitive instances	219,477
Primitive instances by type	
Plane	156,558 (71.3%)
Cylinder	55,349 (25.2%)
Cone	6,541 (3.0%)
Sphere	1,029 (0.5%)
Primitives per sample	
Min / median / max	2 / 7 / 38
25th / 75th percentile	4 / 10
Mean	7.32
Type composition	
Plane	5,800 (19.3%)
Cylinder	19 (0.1%)
Cone	87 (0.3%)
Sphere	15 (0.1%)
Plane+Cylinder	20,404 (68.0%)
Plane+Cone	448 (1.5%)
Plane+Sphere	139 (0.5%)
Cone+Cylinder	22 (0.1%)
Cylinder+Sphere	29 (0.1%)
Plane+Cone+Cylinder	2,538 (8.5%)
Plane+Cylinder+Sphere	339 (1.1%)
Plane+Cone+Cylinder+Sphere	121 (0.4%)
Other mixed combinations	39 (0.1%)

F1: $\frac{1}{|\mathcal{M}|} \sum_{(k^*,g^*)\in\mathcal{M}} \mathrm{F}_1(k^*, g^*),$

Type: $\frac{1}{|\mathcal{M}|} \sum_{(k^*,g^*)\in\mathcal{M}} \mathbb{1}\{\hat{c}_{k^*} = c_{g^*}\},$

Axis: $\frac{\sum_{(k^*,g^*)\in\mathcal{M}} \mathbb{1}\{\hat{c}_{k^*} = c_{g^*}\} \arccos\langle n_{k^*}, n_{g^*}\rangle}{\sum_{(k^*,g^*)\in\mathcal{M}} \mathbb{1}\{\hat{c}_{k^*} = c_{g^*}\}},$

Res: $\frac{\sum_{(k^*,g^*)\in\mathcal{M}} \mathbb{1}\{\hat{c}_{k^*} \neq \emptyset\}\, \mathbb{E}_{\mathbf{x}\sim U(\theta_{g^*})}\, \mathrm{D}(\mathbf{x}, \theta_{k^*})}{\sum_{(k^*,g^*)\in\mathcal{M}} \mathbb{1}\{\hat{c}_{k^*} \neq \emptyset\}},$

Cov: $\frac{\sum_{(k^*,g^*)\in\mathcal{M}} \mathbb{1}\{\hat{c}_{k^*} \neq \emptyset\}\, \mathbb{E}_{\mathbf{x}\sim U(\theta_{g^*})} \mathbb{1}\{\mathrm{D}(\mathbf{x}, \theta_{k^*}) < \epsilon\}}{\sum_{(k^*,g^*)\in\mathcal{M}} \mathbb{1}\{\hat{c}_{k^*} \neq \emptyset\}},$

where $\mathrm{D}(\mathbf{x}, \theta_{k^*})$ denotes the point-to-primitive distance, and $\mathbf{x} \sim U(\theta_{g^*})$ indicates uniform sampling over the bounded surface of the ground-truth primitive.