LADA: LEARNED DECOMPOSITION AND ASSEMBLY FOR NON-MANIFOLD SURFACE RECONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Explicit reconstruction of non-manifold surfaces remains challenging because, although most geometry is locally manifold-like, the global topology is governed by a small number of intricate non-manifold curves. Existing methods typically resolve this topology locally using marching-cubes-style algorithms, making them prone to missing, broken, or fragmented non-manifold edges. We view a non-manifold surface as a union of manifold patches meeting along non-manifold curves and propose a decomposition-and-assembly framework for explicit reconstruction from point clouds. First, a learned decomposition partitions the point cloud into overlapping blocks and predicts whether neighboring point pairs belong to the same manifold patch. Thresholding these affinities and applying union-find splits the points into a set of manifold patches as well as points on non-manifold curves. Second, the assembly step reconstructs each manifold patch with an off-the-shelf manifold surface reconstructor and stitches the patches with explicitly non-manifold edges. Because non-manifold structures are local, the network is trained only on shapes created by intersecting manifold meshes and generalizes zero-shot to unseen data. Across three benchmarks, including CAD models and point clouds sampled from neural implicit fields, our method matches strong baselines in Chamfer distance while achieving the lowest non-manifold Chamfer distance and substantially fewer topological errors on both clean and noisy inputs.

1 INTRODUCTION

Many surfaces of practical interest are not manifolds. Assemblies of CAD parts, architectural models, and thin-walled structures are often unions of manifold patches. Wherever several patches meet along a curve, the resulting surface is no longer a manifold. Almost all of the surface may still be locally simple: away from the non-manifold curves, it is a regular 2-manifold that mature reconstruction methods can handle well. The topology, however, is decided entirely along the non-manifold curves, which occupy only a small one-dimensional subset of the shape. Reconstructing an explicit non-manifold mesh from an unoriented point cloud, therefore, hinges on locating these curves and connecting the patches correctly along them.

Unsigned distance fields (UDFs) (Chibane et al., 2020; Zhou et al., 2022) are able to represent non-manifold surfaces, yet extracting meshes from UDFs remains challenging. Approaches based on local pseudo-signs (Guillard et al., 2022) and shrinking double-cover surfaces (Hou et al., 2023) do not produce topological non-manifold curves, whereas dual-contouring variants (Zhang et al., 2023; Bao et al., 2026), multi-material extraction (Chen et al., 2026), and direct point-cloud meshing (Ren & Hou, 2025) infer connectivity on a per-cell basis using field values or crossing indicators. Nonetheless, these methods still cannot robustly recover non-manifold curves, often yielding broken, missing, or fragmented non-manifold edges (Fig. 3).

We take an alternative strategy and make the decomposition of the shape into manifold patches the central object. Cutting a non-manifold surface along its non-manifold curves yields a collection of independent manifold patches, each of which can be handled by an off-the-shelf manifold reconstructor. The problem therefore splits into two parts: deciding where to cut the point cloud and assembling the reconstructed patches back into one mesh. The first part contains most of the ambiguity. Hand-designed signals such as normals or point density are unreliable near non-manifold curves, where the adjacent patches may be equally smooth and symmetric. We instead learn this

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

decision from data. This is feasible because non-manifold structure is fundamentally local: within a small neighborhood, only a limited set of configurations occurs, most commonly three patches meeting along a curve. Such local patterns are easy to learn, and training shapes for them are easy to construct by intersecting manifold meshes, so ground-truth patch membership is known by construction. Once a correct decomposition is available, the second part requires no learning and can be carried out deterministically.

Our contributions are as follows. (1) We formulate the non-manifold surface reconstruction as a decomposition followed by assembly. We decouple non-manifold surfaces into manifold patches that could reuse off-the-shelf methods for manifold reconstruction. Then, these manifold patches are assembled into the desired non-manifold surface mesh. (2) We introduce a learned decomposition that estimates pairwise affinities over overlapping blocks, because non-manifoldness is a local phenomenon. This is more straightforward than training on full models. The method is trained only on simple shapes. The assembly step then stitches the recovered patches together along explicitly generated non-manifold surfaces. (3) On three zero-shot test sets, our method matches the strongest baselines in Chamfer distance while delivering the lowest non-manifold Chamfer distance and dramatically fewer topological errors; these benefits persist even in the presence of noise.

2 Related Work

Manifold surface reconstruction. Classical methods (Kazhdan et al., 2006; Kazhdan & Hoppe, 2013; Amenta & Bern, 1998) and learned implicit fields (Park et al., 2019; Gropp et al., 2020; Ma et al., 2021; Wang et al., 2021; Mescheder et al., 2019; Peng et al., 2020), typically extracted with Marching Cubes (MC) (Lorensen & Cline, 1987), reconstruct a single closed manifold. Extensions using additional signs or masks (Chen et al., 2022a; Meng et al., 2023; Liu et al., 2024) can represent open surfaces, but still output subsets of one manifold. These methods assume that the input lies on a single 2-manifold, so the normal orientation and inside–outside distinction they rely on are undefined near non-manifold curves. Our method leaves such manifold reconstructors unchanged and applies the iterative Poisson surface reconstruction (iPSR) (Hou et al., 2022) with Neumann boundary conditions (Li et al., 2025) to each patch point set after decomposition (Sec. 3.3).

Non-manifold surface reconstruction. Existing work typically follows two routes. The first meshes the point cloud directly: Wang et al. (2005) voxelize the points, thin the voxels into a surface, and polygonize it; SALS (Ren & Hou, 2025) predicts whether and where each line segment in space crosses the surface and connects the crossings by an edge-based dual contouring. The second uses an unsigned distance field (UDF) (Chibane et al., 2020; Zhou et al., 2022; Ye et al., 2022; Zhou et al., 2023; Long et al., 2023; Kong et al., 2026), which represents non-manifold surfaces once the sign is dropped, so the difficulty moves to mesh extraction. Recovering local pseudo-signs for MC (Guillard et al., 2022; Zhou et al., 2022; Stella et al., 2024) or shrinking an offset isosurface onto the zero level set (Hou et al., 2023) never yields an edge shared by three or more faces; dual contouring (DC) (Ju et al., 2002; Chen et al., 2022b; Zhang et al., 2023) produces non-manifold connections only as a by-product, whereas DCx (Bao et al., 2026) corrects DC connectivity with a lookup table over expanded cubes, MIND (Chen et al., 2026) derives material-interface labels for multi-material MC (Wu & Sullivan Jr, 2003), and Kohlbrenner & Alexa (2026) resample the UDF adaptively with a regular triangulation, all outputting non-manifold edges by design. In every case the topology is decided locally from field values or crossing flags, so field errors near a non-manifold curve become connectivity errors, and none exposes the manifold patches or non-manifold curves. Our method also starts from a point cloud, raw or sampled from any neural implicit field, but decides where to cut with a learned decomposition (Sec. 3.2) and generates every chain of non-manifold edges deterministically during assembly (Sec. 3.3).

Primitive and patch segmentation. Primitive and patch segmentation is the closest task in form. Some methods predict a membership matrix with a fixed maximum number of primitives and use Hungarian matching (Li et al., 2019), others cluster per-point embeddings by mean shift (Sharma et al., 2020; Yan et al., 2021; Li et al., 2023), and PrimitiveNet (Huang et al., 2021) classifies locally adjacent point pairs and grows regions from them; all fit a parametric surface to every instance, most rely on normals, and their boundaries are changes of primitive type on one manifold. Our decomposition differs in that its boundaries are non-manifold curves, across which both sides may be smooth and symmetric, it makes only local binary decisions on a mutual K-nearest-neighbor (KNN)

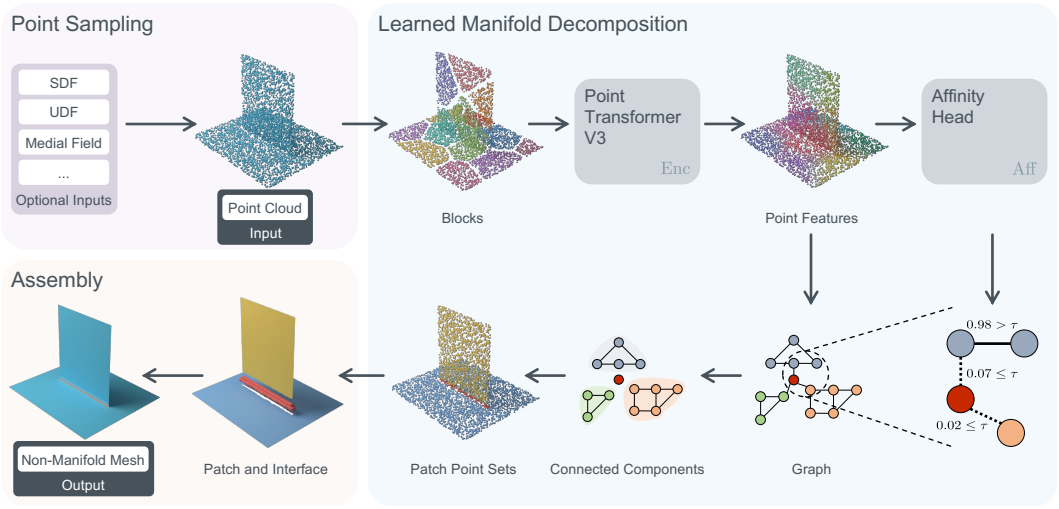


Figure 1: Pipeline overview. A learned decomposition decides where to cut, followed by a deterministic assembly that reconstructs every patch, recovers its interface, and stitches the patches back together along generated non-manifold edges.

graph and takes connected components by union-find, and it is supervised by simple composite shapes built by intersecting manifold meshes (Sec. 4.1).

3 Methodology

3.1 Overview

We aim to reconstruct a non-manifold surface $\mathcal{S}$ from a point cloud $\mathcal{P} = \{\boldsymbol{p}_i\}_{i=1}^{N}$ with $\boldsymbol{p}_i \in \mathbb{R}^3$. A non-manifold surface $\mathcal{S} = \bigcup_k \mathcal{S}_k$ is the union of a set of 2-manifolds intersecting at non-manifold curves. We call each $\mathcal{S}_k$ a *manifold patch*. Three or more patches meet along one-dimensional curves, which we call *non-manifold curves*. An edge of a triangle mesh is a *non-manifold edge* if it is shared by at least three triangular faces.

By splitting a non-manifold surface along its non-manifold curves, it can be decomposed into a collection of independent manifold patches. Motivated by this observation, our approach proceeds in two stages (Fig. 1). The learned decomposition partitions $\mathcal{P}$ into one point set for each manifold patch (Sec. 3.2). The assembly then reconstructs the patches individually and stitches them together into a single mesh (Sec. 3.3).

For decomposing manifold patches, hand-designed signals like normals and local point density are often unreliable, so we instead learn this decision with a network. We begin by partitioning $\mathcal{P}$ into overlapping blocks. A backbone network (Point Transformer V3 (Wu et al., 2024)) computes a feature for each point within each block. Next, an affinity head estimates an affinity score for every pair of neighboring points. We retain edges with an affinity no smaller than τ to form a graph G_τ. Any connected component in G_τ containing at least $N_{\min}$ points is treated as a patch point set $\mathcal{P}_k$. Points that are not assigned to such components lie on non-manifold curves (shown in red in Fig. 1).

In the assembly step, each patch point set is sampled from a single manifold patch. We can therefore reconstruct it with an off-the-shelf manifold reconstructor, which yields a patch mesh $\mathcal{M}_k$. We then stitch them into a non-manifold mesh.

3.2 Learned Manifold Decomposition

Our key observation is that non-manifold structure is fundamentally local. Within a small neighborhood, the non-manifold structure configurations are quite limited–most often involving three manifold patches that meet along a curve. These local patterns are easy to learn and, importantly,

easy to generate training data. By comparison, learning a model over entire shapes seems significantly harder, both in producing training data and in the learning process itself. We therefore train on local blocks instead of full shapes. Learned manifold decomposition has three stages: we first partition the point cloud into overlapping blocks, then estimate pairwise affinities inside each block, and finally merge the local predictions into a globally consistent decomposition using union-find.

Block-wise feature extraction. We split $\mathcal{P}$ into overlapping blocks $\{\mathcal{B}_b\}_{b=1}^{N_b}$: farthest point sampling selects N_b centers, assigning every point to its nearest center yields disjoint *cores* that cover $\mathcal{P}$ exactly once, and each core is grown outward by a breadth-first search of h hops on a KNN adjacency graph of $\mathcal{P}$, which adds an *overlap region* shared with the neighboring blocks. Every block is processed independently by a shared backbone Enc (Point Transformer V3, PTv3 (Wu et al., 2024)), which takes only the block-normalized coordinates $\tilde{\boldsymbol{x}}_i$ as input and returns a feature $\boldsymbol{f}_i = \mathrm{Enc}(\mathcal{B}_b)_i \in \mathbb{R}^d$ for each point. The normalization, together with a random rotation of every block during training, removes global scale and pose so that the backbone sees only local geometry.

Because of the overlapping regions, a point may appear in several blocks, receiving a different feature from each forward pass (strictly $\boldsymbol{f}_i^b$; we state this once and drop the superscript b hereafter). Different forward passes use different normalization and context, so these features cannot be compared across blocks. We therefore compute the KNN graph and the affinities below only among the core and its one-hop ring within each block; the remaining points of the overlap region serve solely as context for the backbone. This ensures that every pairwise comparison uses features from the same forward pass. Meanwhile, the overlap regions connect the connected components that the same patch forms in neighboring blocks, which keeps the final decomposition globally consistent, as we explain shortly. During training we use blocks at several granularities (different combinations of core size and h); inference uses a single granularity, with values given in Sec. 4.1.

Pairwise affinity. Within each block, we build a mutual K-nearest-neighbor (KNN) graph: (i, j) is an edge only if $j \in \mathcal{N}_K^b(i)$ and $i \in \mathcal{N}_K^b(j)$. The mutuality requirement discards one-sided edges that reach from a denser patch into a sparser one or across a gap. These are exactly the edges most likely to cross a non-manifold curve. For every edge, we predict an *affinity* with an *affinity head* Aff,

$$a_{ij} = \sigma\big(\mathrm{Aff}(\boldsymbol{g}_{ij})\big) \in [0,1], \qquad \boldsymbol{g}_{ij} = \big[\boldsymbol{f}_i + \boldsymbol{f}_j,\ |\boldsymbol{f}_i - \boldsymbol{f}_j|,\ \boldsymbol{f}_i \odot \boldsymbol{f}_j,\ |\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j|,\ \|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{x}}_j\|_2\big], \quad (1)$$

which estimates whether $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ lie on the same manifold patch. Here Aff is a two-layer MLP that outputs a logit, and σ is the sigmoid function. The descriptor $\boldsymbol{g}_{ij}$ consists of the sum, the elementwise absolute difference, and the elementwise product of the two features $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$, together with the elementwise absolute coordinate difference and the distance between the two points $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_j$, so $\boldsymbol{g}_{ij} = \boldsymbol{g}_{ji}$. Hence, the outputs of Aff are symmetric $a_{ij} = a_{ji}$. Fig. 1 illustrates the intended behavior with magnified KNN pairs: two neighbors on the same patch should receive an affinity close to 1 and their edge is kept, whereas two neighbors that straddle a non-manifold curve—although equally close or even closer in space—should receive an affinity close to 0 and their edge is discarded. Sec. 4.4 ablates two design choices: the input to Aff and the pairwise formulation.

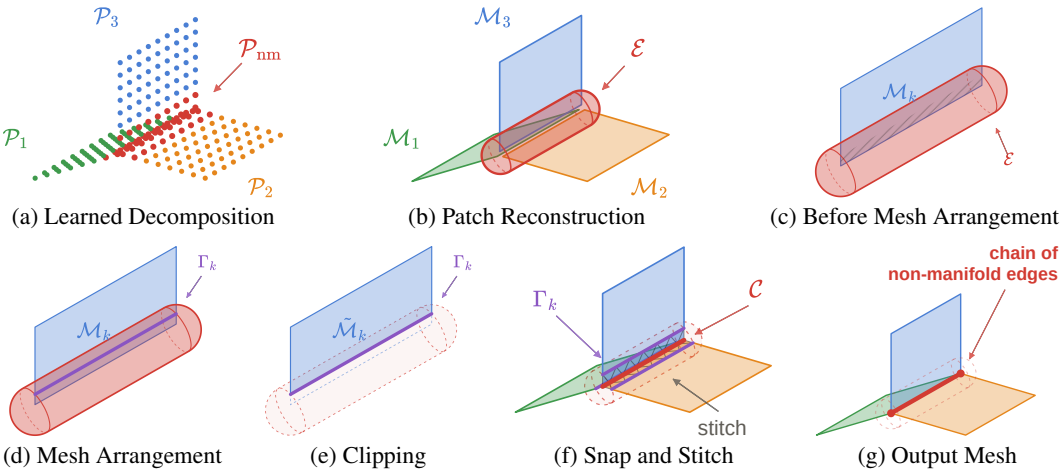
Supervision. We train on simple composite shapes (Sec. 4.1), whose patch membership $\mathcal{P}_k^{\mathrm{gt}}$ and non-manifold curves $\mathcal{C}^{\mathrm{gt}}$ are known by construction. A pair is labeled $a_{ij}^{\mathrm{gt}} = 1$ if both points belong to the same $\mathcal{P}_k^{\mathrm{gt}}$, and 0 otherwise.

Enc and Aff are trained jointly with a group-balanced binary cross-entropy (BCE) loss,

$$\mathcal{L}_{\mathrm{BCE}} = \frac{1}{N_b} \sum_{b=1}^{N_b} \frac{\mathcal{L}_{\mathrm{pos}}^b + 2\,\mathcal{L}_{\mathrm{neg}}^b}{3}, \quad (2)$$

where $\mathcal{L}_{\mathrm{pos}}^b$ and $\mathcal{L}_{\mathrm{neg}}^b$ are the mean BCE over the positive and negative pairs of block b. If a block contains only one label, the missing term is dropped and the weights are renormalized. Negative pairs receive twice the weight of positive pairs, because a single false-positive edge across a non-manifold curve merges two patches into one connected component.

Graph construction and union-find. At inference, the edges of each block are mapped back to global point indices, and affinities predicted by multiple blocks for the same pair are averaged into



(a) Learned Decomposition (b) Patch Reconstruction (c) Before Mesh Arrangement

(d) Mesh Arrangement (e) Clipping (f) Snap and Stitch (g) Output Mesh

Figure 2: Assembly on three manifold patches meeting along one non-manifold curve. Panel (b) is patch reconstruction, (c)–(e) interface recovery, and (f)–(g) non-manifold edge generation.

$\bar{a}_{ij}$. Thresholding with a fixed τ yields a global graph $G_\tau = (\mathcal{P}, E_\tau)$ with $E_\tau = \{(i,j) : \bar{a}_{ij} \geq \tau\}$, whose connected components are obtained by union-find. Points in the overlap regions carry edges in neighboring blocks, so components of the same patch across blocks are joined through these shared points without any cross-block comparison.

Connected components with fewer than $N_{\min}$ points—typically singletons—form the *predicted points on non-manifold curves* $\mathcal{P}_{\text{nm}}$. The remaining components are the *patch point sets* $\{\mathcal{P}_k\}$, giving the decomposition $\mathcal{P} = \mathcal{P}_{\text{nm}} \sqcup \mathcal{P}_1 \sqcup \cdots \sqcup \mathcal{P}_{N_p}$ (Fig. 1). Both τ and $N_{\min}$ are fixed hyperparameters (Sec. 4.1); Sec. A gives the implementation details of both stages.

3.3 Assembly

Manifold patch reconstruction. Since each patch point set $\mathcal{P}_k$ contains the points of a single manifold patch, we reconstruct all patch point sets independently and in parallel using a manifold reconstructor, yielding the corresponding *manifold patch meshes* $\mathcal{M}_k$ (Fig. 2a–b). We use iterative Poisson surface reconstruction (iPSR) with Neumann boundary conditions (Hou et al., 2022; Li et al., 2025). Since a Poisson surface may extend beyond the support of $\mathcal{P}_k$, we trim $\mathcal{M}_k$ by discarding faces whose distance to $\mathcal{P}_k$ exceeds a threshold δ_{trim}.

Manifold patch stitching for non-manifold surface reconstruction. We dilate $\mathcal{P}_{\text{nm}}$ into a tube-like watertight manifold, the *non-manifold envelope* $\mathcal{E}$, which serves as an auxiliary. Each patch mesh $\mathcal{M}_k$ generally intersects $\mathcal{E}$ near the non-manifold curves. As shown in Fig. 2c–d, mesh arrangement (Cherchi et al., 2020) computes their explicit intersection curve, the *interface* Γ_k, along which we clip the portion of $\mathcal{M}_k$ inside $\mathcal{E}$; this yields the clipped patch mesh $\tilde{\mathcal{M}}_k$ with $\Gamma_k \subset \partial\tilde{\mathcal{M}}_k$ (Fig. 2e). We then subdivide $\mathcal{E}$ according to the recovered interfaces, establishing the correspondence $\Gamma_k \leftrightarrow \mathcal{E}$ on the refined envelope. To explicitly extract the non-manifold topology, we further contract the refined $\mathcal{E}$ into the estimated non-manifold curves $\mathcal{C}$ using the mean-curvature-flow (MCF) skeletonization (Tagliasacchi et al., 2012) of CGAL, which provides the correspondence $\mathcal{E} \leftrightarrow \mathcal{C}$. Composing the two correspondences yields the *interface-to-curve correspondence* $\phi_k : \Gamma_k \to \mathcal{C}$ for each patch. Figure 2f–g illustrates the resulting organization and final stitching: guided by ϕ_k, all interfaces meeting along the same curve of $\mathcal{C}$ are snapped onto it and stitched consistently along it, so that every non-manifold curve is realized as one chain of non-manifold edges, recovering the non-manifold connectivity of the final non-manifold mesh $\mathcal{M}$.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Data. We train on 1,800 *simple composite shapes*, built by combining and intersecting a few planar meshes or Thingi10K meshes (Zhou & Jacobson, 2016), for which $\mathcal{P}_k^{\mathrm{gt}}$ and $\mathcal{C}^{\mathrm{gt}}$ are known by construction. We use 40% as a validation set, used only for hyperparameter selection, e.g., $\tau = 0.5$, and for the ablation study (Sec. 4.4), and set $N_{\min} = 16$. All evaluation is zero-shot on three unseen datasets: 1) **Constructed** (31 shapes), created because no public non-manifold dataset provides $\mathcal{C}^{\mathrm{gt}}$; 2) **Non-manifold ABC** (50 shapes), each formed by intersecting two CAD models from ABC (Koch et al., 2019); and 3) **QMDF** (14 medial fields), which we learn with Q-MDF (Kong et al., 2026) from 14 point clouds; it has no $\mathcal{C}^{\mathrm{gt}}$. Sec. B and Sec. C describe the training data and the test sets and show examples.

Baselines and metrics. We compare five non-manifold reconstruction methods using their official code and default parameters: DCUDF (Hou et al., 2023), DMUDF (Zhang et al., 2023), MIND (Chen et al., 2026), DCx (Bao et al., 2026), and SALS (Ren & Hou, 2025). We omit Kohlbrenner & Alexa (2026), whose method cannot faithfully reconstruct non-manifold curves where more than three manifold patches meet. For Constructed and Non-manifold ABC, we follow DEUDF (Xu et al., 2025) to learn a UDF from 300K sampled points, which is used by DCUDF, DMUDF, MIND, and DCx. For QMDF, these four methods directly use the learned fields. SALS and our method operate on point clouds, obtained either from the sampled points or by sampling points from the fields and projecting them onto the zero level set. SALS uses 40K points following its official setting. For our method, the point cloud is normalized to a longest bounding-box extent of 2 and placed in a padded 96^3 voxel grid spanning $[-1.05, 1.05]^3$, giving a voxel size of 0.021875. We retain at most one observed point per occupied voxel, chosen as the point closest to the voxel center. We report CD; non-manifold CD; and F1@0.01, which compare the non-manifold edges of $\mathcal{M}$ with $\mathcal{C}^{\mathrm{gt}}$; and the Betti-number errors $|\Delta b_0|$, $|\Delta b_1|$, $|\Delta b_2|$, and the Euler-characteristic error $|\Delta\chi|$. Sec. D and Sec. E provide the settings of the baselines and the details of the evaluation.

Implementation. All experiments were run on an NVIDIA Tesla V100 PCIe GPU with 32 GB of memory and an Intel Xeon Platinum 8168 CPU. Training takes 3 h 23 min on a single GPU. At test time, the learned decomposition takes about 1.64 s per input point cloud on average, and the assembly takes about 349.5 s per test shape on average. Sec. A gives the hyperparameters and further implementation details of both stages.

4.2 RESULTS ON CLEAN DATA

Our method achieves the lowest non-manifold CD and by far the best topology on all three test sets (Table 1), although it has seen none of them during training. It also attains a competitive CD, second only to DCx. As Fig. 3 shows, it produces one continuous chain of non-manifold edges along each non-manifold curve and none elsewhere, whereas the baselines miss, break, or scatter them.

4.3 RESULTS ON NOISY DATA

We evaluate on Constructed, the only test set with exact $\mathcal{C}^{\mathrm{gt}}$, and add isotropic Gaussian noise with a standard deviation $\epsilon \in \{0.001, 0.002, 0.004\}$ in the input coordinates $[-1, 1]$ to its inputs. For DCUDF, DMUDF, MIND, and DCx, we add the noise to the 300K sampled points and re-learn the UDF from them with DEUDF. Our method keeps the lowest non-manifold CD at every noise level, and its $|\Delta\chi|$ stays low as ϵ grows, whereas those of the baselines rise sharply (Table 2, Fig. 4).

4.4 ABLATION STUDY

All experiments in this section are evaluated on the validation set of simple composite shapes. We ablate the two design choices of the learned decomposition and report decomposition quality by ARI and matched mIoU against $\mathcal{P}_k^{\mathrm{gt}}$; Sec. F gives the setup and details.

Table 1: Quantitative comparison on clean inputs. The number in parentheses is the size of each test set, and all metrics are averaged over the shapes for which a method outputs a mesh (#Success). Non-manifold CD and F1@0.01 compare the non-manifold edges of $\mathcal{M}$ with $\mathcal{C}^{\mathrm{gt}}$; $|\Delta b_i|$ and $|\Delta\chi|$ are the absolute errors of the Betti numbers and the Euler characteristic. NA: no non-manifold edges in the output; –: no $\mathcal{C}^{\mathrm{gt}}$ available. Best results are in **bold**.

Dataset	Method	#Success↑	CD↓	Non-manifold CD↓	Non-manifold F1@0.01↑	$\lvert\Delta b_0\rvert\downarrow$	$\lvert\Delta b_1\rvert\downarrow$	$\lvert\Delta b_2\rvert\downarrow$	$\lvert\Delta\chi\rvert\downarrow$
Constructed (31)	DCUDF	31	0.00292	NA	NA	4.58	34.19	1.87	35.42
	DMUDF	31	0.00334	0.0965	0.145	13.13	576.39	4685.26	4139.03
	MIND	31	0.00417	0.0526	0.661	133.58	43.26	133.00	273.77
	DCx	31	**0.00289**	0.0188	**0.907**	2.65	8.03	34.13	28.94
	SALS	31	0.00295	0.0933	0.213	15.45	851.45	639.10	482.90
	Ours	31	0.00305	**0.0071**	0.837	**0.06**	**1.68**	**1.35**	**2.77**
Non-manifold ABC (50)	DCUDF	50	0.00278	NA	NA	3.84	8.04	0.96	8.44
	DMUDF	50	0.00311	0.0935	0.109	3.20	669.10	4748.10	4082.20
	MIND	49	0.00306	0.0131	0.922	38.20	8.37	36.55	67.86
	DCx	50	**0.00276**	0.0070	**0.960**	1.48	2.28	15.16	14.36
	SALS	50	0.00281	0.0796	0.293	6.44	618.04	518.88	520.76
	Ours	50	0.00286	**0.0068**	0.921	**0.04**	**0.58**	**0.84**	**1.38**
QMDF (14)	DCUDF	14	0.00331	–	–	18.64	19.14	19.29	20.36
	DMUDF	8	0.04114	–	–	3.38	617.50	1083.00	1092.13
	MIND	13	**0.00240**	–	–	1.31	58.46	**1.08**	56.69
	DCx	14	0.00257	–	–	4.07	12.43	955.86	950.07
	SALS	14	0.00249	–	–	2.29	695.29	411.29	390.43
	Ours	14	0.00258	–	–	**1.21**	**2.43**	1.29	**2.21**

Table 2: Quantitative comparison on Constructed under Gaussian noise of standard deviation ϵ. Each cell reports non-manifold CD / $|\Delta\chi|$ (lower is better), averaged over the shapes for which a method outputs a mesh: all 31 shapes, except for MIND, which does not finish within 3 hours on 1, 1, and 2 shapes at the three noise levels, respectively. DCUDF, DMUDF, MIND, and DCx extract meshes from a UDF re-learned with DEUDF from the 300K noisy points; SALS and our method take the noisy point clouds directly. NA: no non-manifold edges in the output. Best results are in **bold**.

Noise level	DCUDF	DMUDF	MIND	DCx	SALS	Ours
$\epsilon = 0.001$	NA / 8.55	0.0979 / 5379.97	0.0478 / 211.33	0.0177 / 25.58	0.0933 / 345.00	**0.0111** / **7.23**
$\epsilon = 0.002$	NA / **8.06**	0.1004 / 7843.87	0.0550 / 271.30	0.0186 / 31.00	0.0988 / 636.71	**0.0102** / 9.10
$\epsilon = 0.004$	NA / 76.35	0.1026 / 16383.71	0.0705 / 464.54	0.0563 / 291.65	0.1038 / 2927.77	**0.0225** / **11.26**

Input to the affinity head. We replace $\boldsymbol{g}_{ij}$ with either the scaled inner product $\boldsymbol{f}_i^\top \boldsymbol{f}_j/\sqrt{d}$ or the concatenation of $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$. Both variants fall behind $\boldsymbol{g}_{ij}$ (Table 3). Points on opposite sides of a non-manifold curve see nearly symmetric local geometry and thus have similar features, which a similarity-based score cannot separate; $\boldsymbol{g}_{ij}$ additionally encodes their feature difference and relative position.

Pairwise versus per-point formulation. We replace the pairwise affinity with a per-point classifier or a per-point distance regression, which predicts the points on non-manifold curves and takes the connected components of the remaining KNN graph as the patch point sets. Both yield far lower decomposition quality than the pairwise formulation (Table 4). A per-point prediction must decide inside a narrow band around $\mathcal{C}^{\mathrm{gt}}$, where a single missed point bridges two patches into one connected component, whereas a wrong pairwise decision affects only one edge.

Sensitivity to the threshold τ. We evaluate E_τ as a binary classification of the edges of the KNN graph for τ from 0.1 to 0.9. Edge accuracy and edge F1 remain nearly constant over this range (Table 5), so τ has little effect on the result and needs no careful tuning.

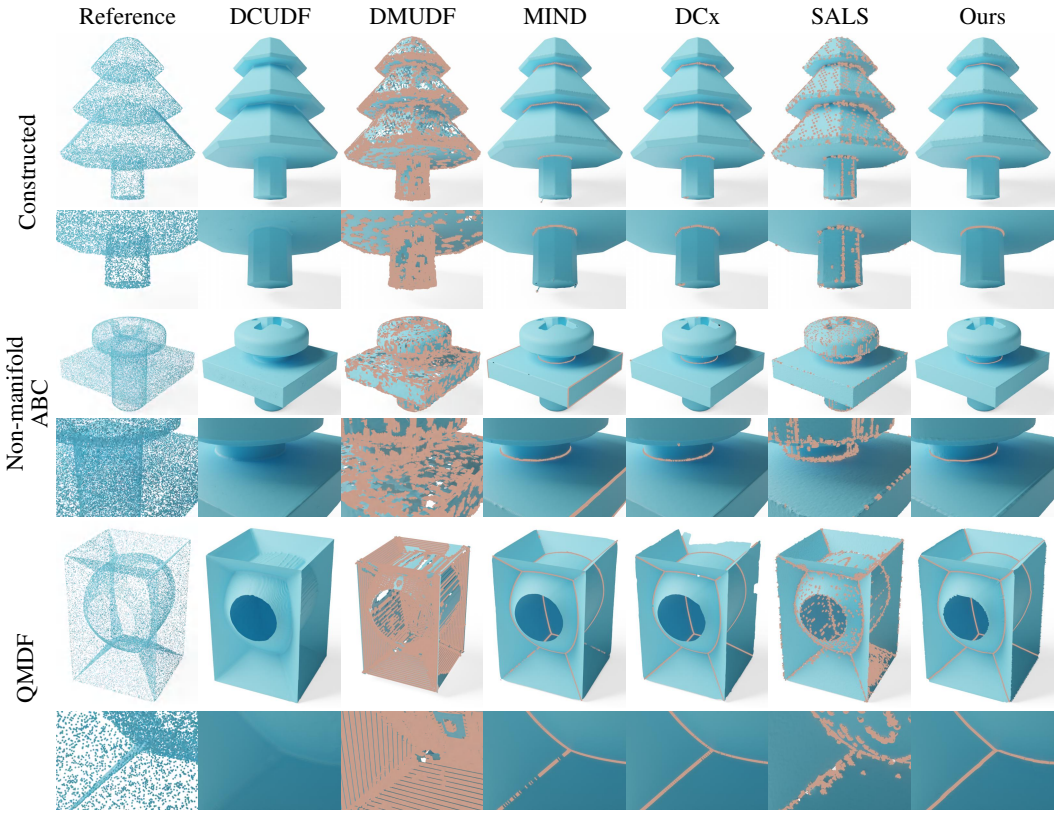


Figure 3: Qualitative comparison on clean inputs, one shape per test set. Bottom images zoom in near the non-manifold curves, and non-manifold edges are shown in orange. Our method produces one continuous chain of non-manifold edges along each non-manifold curve and none elsewhere.

Table 3: Affinity head input.

Affinity head input	ARI↑	mIoU↑
$\sigma\left(\boldsymbol{f}_i^\top \boldsymbol{f}_j/\sqrt{d}\right)$	0.630	0.706
MLP $(\boldsymbol{f}_i, \boldsymbol{f}_j)$	0.785	0.862
MLP $(\boldsymbol{g}_{ij})$ (Ours)	**0.836**	**0.881**

Table 4: Pairwise vs. per-point.

Formulation	ARI↑	mIoU↑
Per-point classification	0.264	0.392
Per-point distance regression	0.260	0.385
Pairwise (Ours)	**0.836**	**0.881**

Table 5: Sensitivity to τ.

τ	Accuracy↑	F1↑
0.1	0.988	0.994
0.3	0.986	0.993
0.5	0.984	0.992
0.7	0.982	0.990
0.9	0.977	0.987

4.5 Limitations

We test the limit of our method on a thin structure made of two parallel squares of side length 1 connected by a rectangle, whose width we call the thickness. The rectangle meets each square along a non-manifold curve, so the shape contains two non-manifold curves. We decrease the thickness from 0.07 to 0.04 and run only our method (Fig. 5). When the thickness is at least 0.05, the output mesh contains two separate chains of non-manifold edges. At a thickness of 0.04, the two chains merge into one and the rectangle disappears. The thickness limit of our method is therefore about 0.05.

The points at the two ends of the rectangle are very close to each other and also close to the points on the two squares. Such points are difficult to separate from noise, outliers, or an adjacent surface, so the network cannot determine whether they originate from noise or from an actual thin structure. In the KNN graph, a point near one non-manifold curve also connects to neighbors near the other curve, which makes both the backbone features and the affinities unreliable. Consequently, in Fig. 5 (right), the predicted points on the two non-manifold curves collapse into a single region. As a result, only one non-manifold curve is recovered.

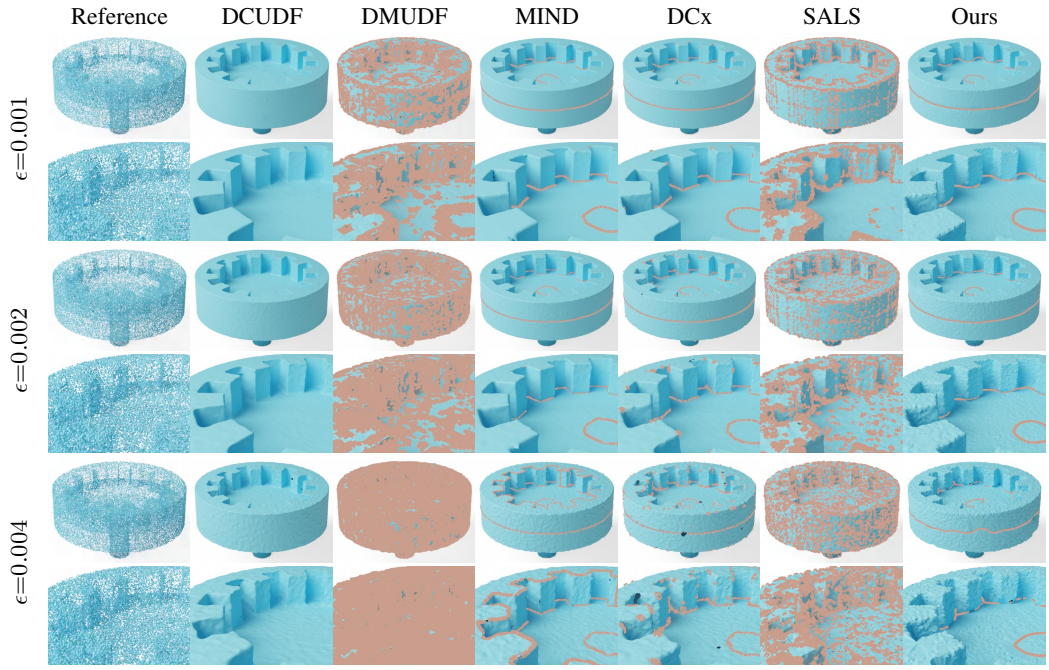


Figure 4: Qualitative comparison on Constructed under Gaussian noise of standard deviation ϵ, one row per noise level. Bottom images zoom in near the non-manifold curves, and non-manifold edges are shown in orange.

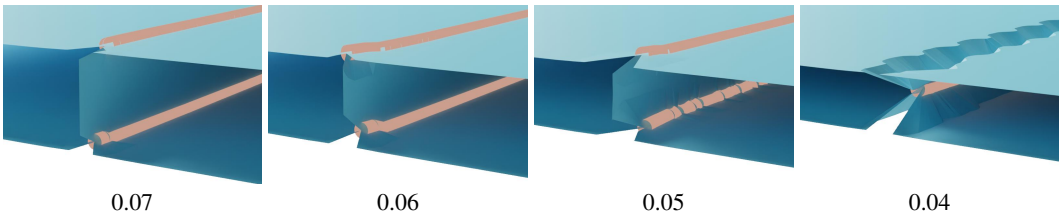


Figure 5: Our method on thin structures of decreasing thickness (left to right). Each shape consists of two parallel squares of side length 1 connected by a rectangle whose width is the thickness; orange marks the chains of non-manifold edges. Two non-manifold curves are recovered for thicknesses of at least 0.05; at 0.04 they merge into one.

5 Conclusion

We have presented a two-stage method for reconstructing explicit non-manifold meshes from raw point clouds as input. A learned decomposition decides where to cut the point cloud, using only local pairwise affinities that a network trained on simple composite shapes predicts within overlapping blocks, and a deterministic assembly reconstructs each manifold patch independently and stitches all patches along explicitly generated non-manifold edges. Decoupling patch geometry from non-manifold topology lets our method reuse manifold reconstruction unchanged while producing non-manifold edges exactly along the non-manifold curves, which gives the lowest non-manifold Chamfer distance and by far the smallest topological errors on three zero-shot test sets, on clean and noisy inputs. Because the decomposition explicitly reveals the manifold patches and non-manifold curves, it may be valuable for more than just reconstruction—for example, enabling editing and simulation on non-manifold shapes. These directions can be explored further in the future.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing and grammar correction of the paper text, for assistance with code implementation, and for searching related work. We have not used generative AI tools for research ideation, drafting the paper text, or data generation and annotation, and no other tasks requiring disclosure apply to this work. We have reviewed all AI-assisted work. In particular, every sentence edited with AI assistance was checked word by word by the authors to ensure that it preserves the intended meaning and is consistent with our methods and experimental results; all references suggested by AI tools were checked against the original papers by the authors; and all AI-assisted code was reviewed by the authors and validated through experiments. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects. The training shapes are built from procedural planar meshes and meshes from Thingi10K (Zhou & Jacobson, 2016), and one test set is built from CAD models in ABC (Koch et al., 2019); both are public datasets, and we use them in accordance with their licenses. The 14 point clouds underlying QMDF were collected by us and capture only objects, without people, private spaces, or personally identifiable information. We are not aware of any direct harmful application of our method beyond the general risks of 3D reconstruction.

REPRODUCIBILITY STATEMENT

Sec. 3 describes both stages of our method. Sec. 4.1 summarizes the training data, test sets, baselines, metrics, and hardware. Sec. A lists the network architecture, training settings, and all hyperparameters of the learned decomposition and the assembly. Sec. B and Sec. C give the construction of the training data and the three test sets, and Sec. D and Sec. E give the baseline settings and the evaluation protocol. We will release our code, trained model, training data, and test sets upon acceptance.

REFERENCES

Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, SCG '98, pp. 39–48, New York, NY, USA, 1998. Association for Computing Machinery.

Qingchao Bao, Xuhui Chen, Jingpeng Yin, Fei Hou, Wencheng Wang, Hong Qin, and Ying He. Dual contouring over expanded cubes (DCx) for zero-level set extraction from neural unsigned distance functions. *ACM Transactions on Graphics (TOG)*, 45(4):1–20, 2026.

Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):349–359, oct 1999.

Weikai Chen, Cheng Lin, Weiyang Li, and Bo Yang. 3psdf: Three-pole signed distance function for learning surfaces with arbitrary topologies. In *Proc. of CVPR*, pp. 18501–18510, 2022a.

Xuhui Chen, Fei Hou, Wencheng Wang, Hong Qin, and Ying He. Mind: Material interface generation from udfs for non-manifold surface reconstruction. *Advances in Neural Information Processing Systems*, 38:120283–120306, 2026.

Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. Neural dual contouring. *ACM Transactions on Graphics (Special Issue of SIGGRAPH)*, 41(4), 2022b.

Gianmarco Cherchi, Marco Livesu, Riccardo Scateni, and Marco Attene. Fast and robust mesh arrangements using floating-point arithmetic. *ACM Transactions on Graphics (TOG)*, 39(6):1–16, 2020.

Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, December 2020.

Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *International Conference on Machine Learning*, 2020.

Benoit Guillard, Federico Stella, and Pascal Fua. Meshudf: Fast and differentiable meshing of unsigned distance field networks. In *European Conference on Computer Vision*, 2022.

Fei Hou, Chiyu Wang, Wencheng Wang, Hong Qin, Chen Qian, and Ying He. Iterative poisson surface reconstruction (ipsr) for unoriented points. *ACM Trans. Graph.*, 41(4), 2022.

Fei Hou, Xuhui Chen, Wencheng Wang, Hong Qin, and Ying He. Robust zero level-set extraction from unsigned distance fields based on double covering. *ACM Trans. Graph.*, 42(6), dec 2023.

Jingwei Huang, Yanfeng Zhang, and Mingwei Sun. Primitivenet: Primitive instance segmentation with local primitive embedding under adversarial metric. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15323–15333. IEEE, 2021.

Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. *ACM Trans. Graph.*, 21(3):339–346, 2002.

Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3), 2013.

Michael M. Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Eurographics Symposium on Geometry Processing*, 2006.

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9601–9611, 2019.

M Kohlbrenner and M Alexa. Meshing unsigned distance fields with regular triangulations. In *Computer Graphics Forum*, pp. e70524. Wiley Online Library, 2026.

Jiayi Kong, Chen Zong, Jun Luo, Shiqing Xin, Fei Hou, Hanqing Jiang, Chen Qian, and Ying He. Quasi-medial distance field (q-mdf): A robust method for approximating and discretizing neural medial axes. *ACM Transactions on Graphics*, 45(3):1–16, 2026.

Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J Guibas. Supervised fitting of geometric primitives to 3d point clouds. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 2647–2655. IEEE, 2019.

Yuanqi Li, Shun Liu, Xinran Yang, Jianwei Guo, Jie Guo, and Yanwen Guo. Surface and edge detection for primitive fitting of point clouds. In *ACM SIGGRAPH 2023 conference proceedings*, pp. 1–10, 2023.

Zhuodong Li, Fei Hou, Wencheng Wang, Xuequan Lu, and Ying He. A divide-and-conquer approach for global orientation of non-watertight scene-level point clouds with 0-1 integer optimization. *ACM Transactions on Graphics (TOG)*, 44(4):1–15, 2025.

Zhen Liu, Yao Feng, Yuliang Xiu, Weiyang Liu, Liam Paull, Michael J. Black, and Bernhard Schölkopf. Ghost on the shell: An expressive representation of general 3d shapes. In *The Twelfth International Conference on Learning Representations*, 2024.

Xiaoxiao Long, Cheng Lin, Lingjie Liu, Yuan Liu, Peng Wang, Christian Theobalt, Taku Komura, and Wenping Wang. Neuraludf: Learning unsigned distance fields for multi-view reconstruction of surfaces with arbitrary topologies. In *Proc. of CVPR*, pp. 20834–20843, 2023.

William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proc. of ACM SIGGRAPH*, pp. 163–169, 1987.

Baorui Ma, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. Neural-pull: Learning signed distance function from point clouds by learning to pull space onto surface. In *ICML*, volume 139, pp. 7246–7257, 2021.

Xiaoxu Meng, Weikai Chen, and Bo Yang. Neat: Learning neural implicit surfaces with arbitrary topologies from multi-view images. In *Proc. of CVPR*, pp. 248–258, 2023.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. of CVPR*, pp. 4455–4465, 2019.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. of CVPR*, pp. 165–174, 2019.

Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Proc.of ECCV*, pp. 523–540, 2020.

Siyu Ren and Junhui Hou. Shape as line segments: Accurate and flexible implicit surface representation. In *International Conference on Learning Representations*, volume 2025, pp. 82842–82862, 2025.

Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomír Měch. Parsenet: A parametric surface fitting network for 3d point clouds. In *European conference on computer vision*, pp. 261–276. Springer, 2020.

Federico Stella, Nicolas Talabot, Hieu Le, and Pascal Fua. Neural surface detection for unsigned distance fields. In *European Conference on Computer Vision*, pp. 394–409. Springer, 2024. ISBN 978-3-031-73636-0.

Andrea Tagliasacchi, Ibraheem Alhashim, Matt Olson, and Hao Zhang. Mean curvature skeletons. In *Computer Graphics Forum*, volume 31, pp. 1735–1744. Wiley Online Library, 2012.

Jianning Wang, Manuel M. Oliveira, and Arie E. Kaufman. Reconstructing manifold and non-manifold surfaces from point clouds. In *Proc. of IEEE Visualization*, pp. 415–422, 2005.

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021.

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4840–4851. IEEE, 2024.

Ziji Wu and John M. Sullivan Jr. Multiple material marching cubes algorithm. *International Journal for Numerical Methods in Engineering*, 58(2):189–207, 2003.

Cheng Xu, Fei Hou, Wencheng Wang, Hong Qin, Zhebin Zhang, and Ying He. Details enhancement in unsigned distance field learning for high-fidelity 3D surface reconstruction. In *Proc. of AAAI*, pp. 8806–8814, 2025.

Siming Yan, Zhenpei Yang, Chongyang Ma, Haibin Huang, Etienne Vouga, and Qixing Huang. Hpnet: Deep primitive segmentation using hybrid representations. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 2733–2742. IEEE, 2021.

Jianglong Ye, Yuntao Chen, Naiyan Wang, and Xiaolong Wang. Gifs: Neural implicit function for general shape representation. In *Proc. of CVPR*, pp. 12819–12829, 2022.

Congyi Zhang, Guying Lin, Lei Yang, Xin Li, Taku Komura, Scott Schaefer, John Keyser, and Wenping Wang. Surface extraction from neural unsigned distance fields. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 22474–22483, 2023.

Junsheng Zhou, Baorui Ma, Yu-Shen Liu, Yi Fang, and Zhizhong Han. Learning consistency-aware unsigned distance functions progressively from raw point clouds. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Junsheng Zhou, Baorui Ma, Shujuan Li, Yu-Shen Liu, and Zhizhong Han. Learning a more continuous zero level set in unsigned distance fields through level set projection. In *Proc. of ICCV*, pp. 3158–3169, 2023.

Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016.

A Implementation Details

This section provides the implementation details and hyperparameters of both stages of our method. The same values are used for all test sets and for both clean and noisy inputs. The distance thresholds of the assembly scale with the sampling density of the input: they are given in units of the mean sampling spacing $\bar{s}$, the average distance from each point of $\mathcal{P}$ to its nearest neighbor, except for the trimming thresholds, which are given in units of s_k, the median nearest-neighbor distance within $\mathcal{P}_k$.

A.1 Learned Decomposition

Blocks. We split $\mathcal{P}$ into overlapping blocks as described in Sec. 3.2. The KNN adjacency graph on which the breadth-first search grows from each core is a mutual 24-nearest-neighbor graph of $\mathcal{P}$, whose edges longer than $3\bar{s}$ are removed, and whose remaining edges are filtered with normals estimated from 12 nearest neighbors (tangent threshold 0.5). Cores contain at least 64 points. Each block is centered at the centroid of its core and scaled by the largest distance from its points to that center, which gives the block-normalized coordinates $\tilde{\boldsymbol{x}}_i$. Table 6 lists the three block granularities. The maximum number of points per block is a capacity rather than a fixed size; unused slots are seen by neither the backbone, the KNN graph, nor the loss. Training uses all three granularities, and every mini-batch contains blocks of a single granularity.

Table 6: Block granularities. Training uses all three; inference uses the medium one.

Granularity	Max. points per block	Target core size	Hops h
Fine	1,024	384	2
Medium	2,048	768	3
Coarse	3,072	1,152	5

Backbone and affinity head. The backbone is PTv3 (Wu et al., 2024) with the configuration in Table 7. It takes the block-normalized coordinates $\tilde{\boldsymbol{x}}_i$ both as point coordinates and as the only input feature and returns features of dimension $d = 256$. The descriptor $\boldsymbol{g}_{ij}$ in equation 1 therefore has dimension $3d + 3 + 1 = 772$. The affinity head Aff is $\mathrm{Linear}(772, 256) \to \mathrm{LayerNorm} \to \mathrm{GELU} \to \mathrm{Dropout}(0.1) \to \mathrm{Linear}(256, 1)$. The KNN graph on which the affinities are predicted uses $K = 24$ and is built only among the core and its one-hop ring of each block, as described in Sec. 3.2; point pairs outside this graph are not treated as negative pairs.

Training. Enc and Aff are trained with the loss in equation 2 only, with no auxiliary loss, on blocks drawn from the training point clouds (Sec. B.2). A point pair that appears in several overlapping blocks contributes one loss term per block, each computed with the context of that block. The remaining settings are listed in Table 8. One epoch consists of 1,024 blocks, so the budget of 256,000 blocks corresponds to 250 epochs and 16,000 optimizer updates. We train with two random seeds and select the seed, the checkpoint, and τ jointly on the validation set; the test sets are never used for selection. The selected checkpoint is the one at epoch 250.

Inference. At inference, $\mathcal{P}$ is split into blocks of medium granularity only. The affinities predicted for the same pair by several blocks are fused by the arithmetic mean $\bar{a}_{ij}$, and τ and $N_{\min}$ take the values in Table 8.

Table 7: Configuration of the PTv3 backbone.

Parameter	Setting
Serialization orders	z-order, transposed z-order (shuffled during training)
Grid size (normalized coordinates)	0.02
Encoder strides	2, 2, 2, 2
Encoder channels	32, 64, 128, 256, 512
Encoder depths	2, 2, 2, 6, 2
Encoder attention heads	2, 4, 8, 16, 32
Decoder channels	256, 256, 512, 512
Decoder depths	2, 2, 2, 2
Decoder attention heads	4, 4, 8, 16
Patch size (all encoder and decoder stages)	128
Output feature dimension d	256
Drop path rate	0.1
Batch normalization	Frozen
Flash attention	Disabled
Attention and softmax precision	Upcast to FP32

Table 8: Training and inference settings of the learned decomposition.

Parameter	Setting
Training	
Loss	equation 2, positive : negative $= 1 : 2$
Optimizer	AdamW
Learning rate	2.12×10^{-4}
Weight decay	10^{-4}
Learning rate schedule	Warmup over the first 5% of updates, then cosine decay
Gradient clipping (norm)	1.0
Batch size	16 blocks
Training budget	256,000 blocks
Numerical precision	FP32 (no mixed precision)
Data augmentation	One random 3D rotation per block
Validation interval	Every 25 epochs
Early stopping	From epoch 100, patience 3 validations, min. improvement 0.001
Random seeds	2
Inference	
Block granularity	Medium
Fusion of a_{ij} across blocks	Arithmetic mean $\bar{a}_{ij}$
Affinity threshold τ	0.5
Minimum patch size $N_{\min}$	16 points

A.2 Assembly

Table 9 lists the hyperparameters of the assembly, grouped by step.

Decomposition refinement. Before patch reconstruction, we refine the decomposition on a *coarse mesh* of $\mathcal{P}$, a local Delaunay triangulation over the nearest neighbors of every point with long edges removed. Points at which two patch point sets touch on the coarse mesh form a contact; we add every contact, expanded by one hop, to $\mathcal{P}_{\mathrm{nm}}$, dilate $\mathcal{P}_{\mathrm{nm}}$ by one hop so that the following flood fill cannot leak through narrow gaps, and recompute the patch point sets $\mathcal{P}_k$ by a flood fill on the coarse mesh blocked by $\mathcal{P}_{\mathrm{nm}}$. Finally, $\mathcal{P}_{\mathrm{nm}}$ is thinned again for interface recovery and non-manifold edge generation.

Patch reconstruction. iPSR (Hou et al., 2022) starts from normals estimated and oriented over the nearest neighbors of every point; if it fails on a small patch point set, we fall back to local Delaunay triangulation and then to ball pivoting (Bernardini et al., 1999). We then trim each patch mesh in two passes by the distance from each face centroid to $\mathcal{P}_k$: the first pass discards all faces farther than

δ_{trim}, and the second repeatedly discards boundary faces farther than the smaller boundary trimming threshold.

Interface recovery. The unsigned distance field to $\mathcal{P}_{\mathrm{nm}}$ is evaluated on a voxel grid covering the enlarged bounding box of $\mathcal{P}_{\mathrm{nm}}$, and the envelope is its isosurface at δ_{env}. After the mesh arrangement, small disconnected pieces of each clipped patch mesh are discarded.

Table 9: Hyperparameters of the assembly. $\bar{s}$: mean sampling spacing (average nearest-neighbor distance in $\mathcal{P}$); s_k: median nearest-neighbor distance within $\mathcal{P}_k$.

Parameter	Setting
Decomposition refinement	
Coarse mesh neighbors	25
Coarse mesh max. edge length	$2.5\bar{s}$
Contact expansion	1 hop
Dilation of $\mathcal{P}_{\mathrm{nm}}$	1 hop
Patch reconstruction	
iPSR iterations	30
iPSR point weight	100
iPSR octree depth	10
iPSR neighbors	10
Trim threshold δ_{trim}	$4.0\, s_k$
Boundary trimming threshold	$1.5\, s_k$
Interface recovery	
Voxel grid resolution	128^3
Envelope offset δ_{env}	$1.5\bar{s}$
Bounding box enlargement (per side)	$\max(4\bar{s},\ 1.5\,\delta_{\mathrm{env}})$

B Training and Validation Data

B.1 Simple Composite Shapes

All simple composite shapes are built by us and are disjoint from the three test sets. There are 1,800 of them from two sources: 1,200 procedural patch arrangements and 600 Thingi10K-derived shapes. The manifold patches and non-manifold curves of every shape are fixed at construction, so $\mathcal{P}_k^{\mathrm{gt}}$ and $\mathcal{C}^{\mathrm{gt}}$ are available directly. We randomly split the shapes into a training set of 1,080 shapes (60%) and a validation set of 720 shapes (40%). The validation set is used only for hyperparameter selection and the ablation study (Sec. F).

Procedural patch arrangements are built in three steps. First, we pick a topology template. The template fixes the relation between two to four planar meshes: each pair either intersects, shares one edge, or stays apart. There are nine template families: transverse crossing, near-parallel pair, sharp manifold hinge, acute crossing, radial triple, branching, layered cutting, three-plane crossing, and layered branching. Second, we randomly set the pose, size, gap, and extent of each planar mesh within the range the template allows. Third, we intersect the planar meshes and split their triangles along the intersection lines, so that these lines become mesh edges. Thingi10K-derived shapes are built as follows. We select 150 manifold meshes from Thingi10K (Zhou & Jacobson, 2016), then transform, combine, and intersect them. This gives 450 non-manifold shapes. The 150 original meshes are also kept as manifold examples. Fig. 6 and Fig. 7 show examples of the two sources. Both sources contain manifold shapes, for which $\mathcal{C}^{\mathrm{gt}}$ is empty.

B.2 Training Point Clouds

We translate and scale each shape by its bounding box so that its longest side is 2. We then pick triangles in proportion to their area and sample points uniformly inside them, which gives a *training point cloud*. The sampling does not look at any label. The reference spacing is $2.1/96 \approx 0.022$. The number of points grows with the surface area and is not fixed. Each point records the triangle it comes from, and its labels are assigned from that triangle. Table 10 lists the five *sampling settings*: clean, dense, sparse, noisy, and incomplete. Each shape has only one noisy and one incomplete training point cloud, whose strength is one of two levels.

All blocks are split offline before training, with the granularities in Table 6, and are not split again during training. Training draws blocks hierarchically: first a source, then a shape, a granularity, and a sampling setting, and finally a group defined by the number of patches in the block. The source weights are $2/3$ for procedural patch arrangements, $1/4$ for non-manifold Thingi10K-derived shapes, and $1/12$ for manifold Thingi10K meshes. The groups with 1, 2, 3, and at least 4 patches have weights 0.20, 0.25, 0.30, and 0.25. The three granularities have equal weights, and so do the sampling settings of a shape. Shapes are drawn in turn, so that a large shape does not take up too many training steps.

Table 10: Sampling settings of the training point clouds. The reference spacing is $2.1/96$ in the normalized coordinates $[-1, 1]$.

Sampling setting	Generation	Granularity
		1st: fine, medium, coarse
Clean ($\times 3$)	Three independent samplings at the reference spacing	2nd, 3rd: medium
Dense	Resampled at $0.8\times$ the reference spacing	Medium
Sparse	Resampled at $1.5\times$ the reference spacing	Medium
Noisy	Gaussian noise on the 1st clean one, std. 0.05 or $0.15\times$ the reference spacing	Medium
Incomplete	90% or 95% of the points of the 1st clean one kept	Medium

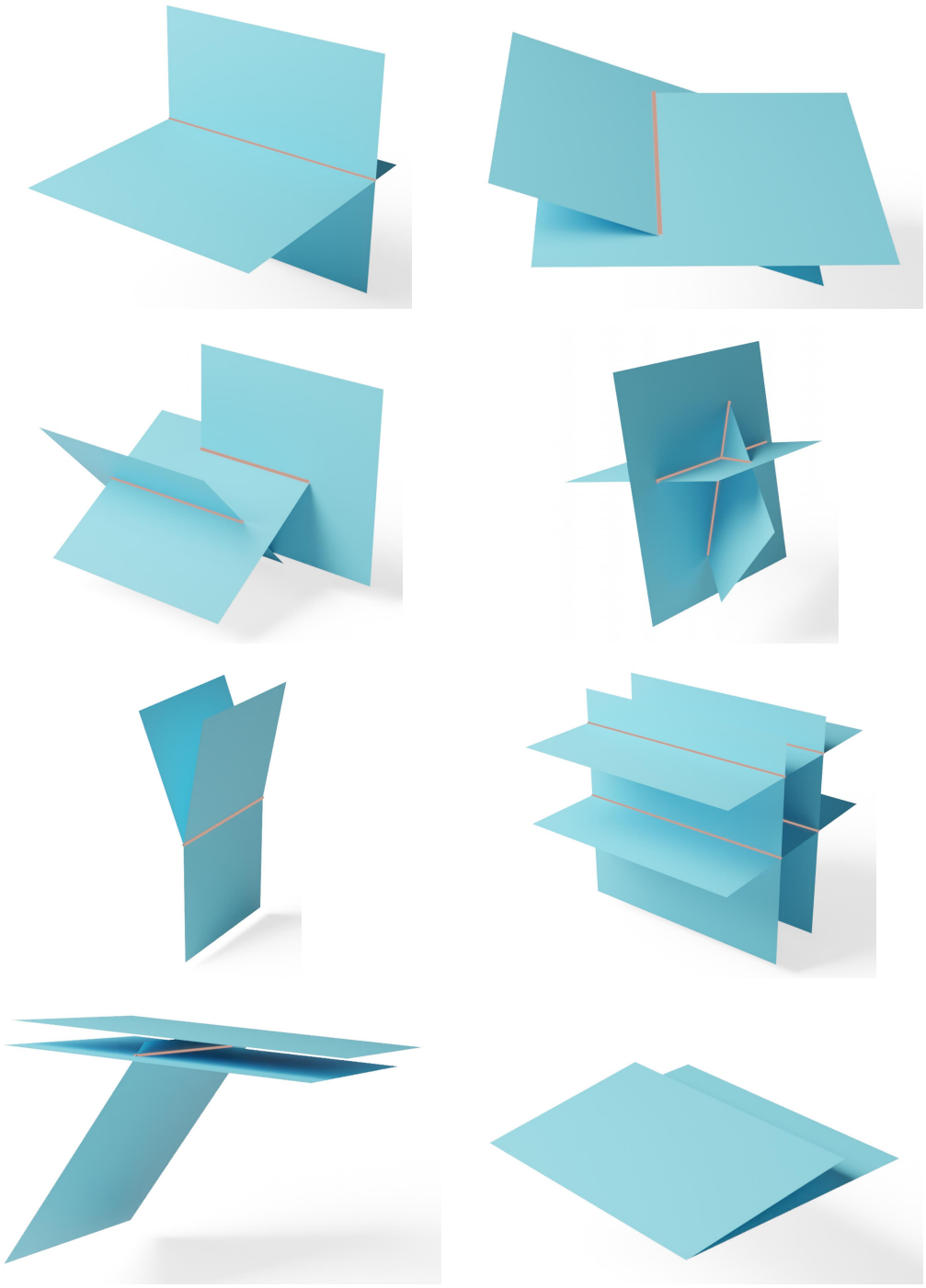


Figure 6: Examples of procedural patch arrangements from our training and validation sets. Orange curves mark the ground truth non-manifold curves $\mathcal{C}^{\mathrm{gt}}$. The bottom-right shape is a sharp manifold hinge: its two planar meshes share one edge, so it is manifold and $\mathcal{C}^{\mathrm{gt}}$ is empty.

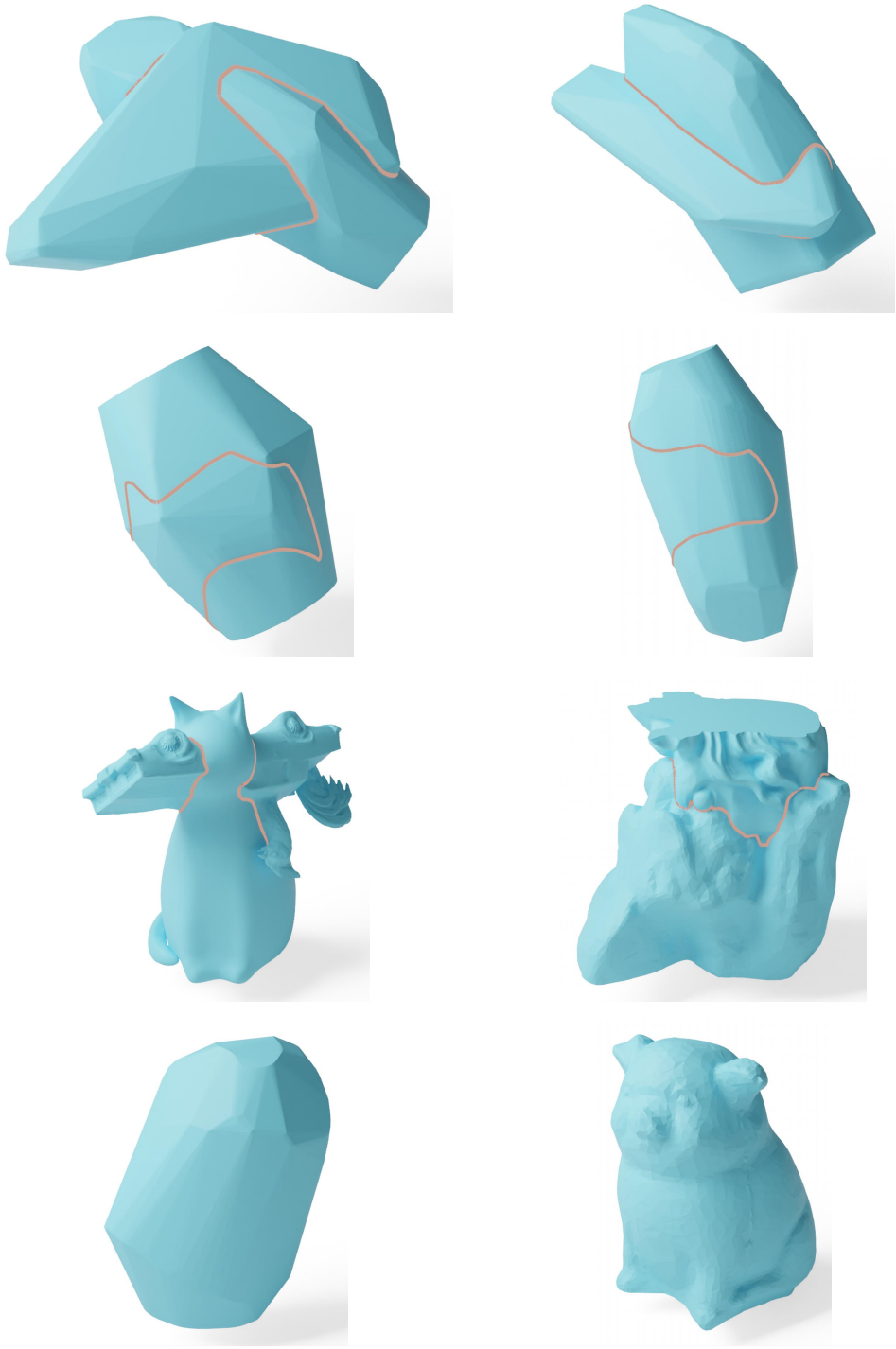


Figure 7: Examples of Thingi10K-derived shapes from our training and validation sets. The first three rows are non-manifold shapes obtained by transforming, combining, and intersecting manifold meshes from Thingi10K (Zhou & Jacobson, 2016); orange curves mark the ground truth non-manifold curves $\mathcal{C}^{\mathrm{gt}}$. The bottom row shows original manifold meshes, for which $\mathcal{C}^{\mathrm{gt}}$ is empty.

C TEST DATA

None of the three test sets overlaps the simple composite shapes or is used for training or hyperparameter selection. For Constructed and Non-manifold ABC, the input point cloud is sampled from the ground truth mesh surface in proportion to face area; for QMDF, it is obtained by sampling points from each learned medial field and projecting them onto its zero level set.

C.1 CONSTRUCTED

No public non-manifold dataset provides $\mathcal{C}^{\mathrm{gt}}$, so we construct 31 shapes, denoted Constructed, separately from the training and validation sets. Fig. 8 shows examples.

C.2 NON-MANIFOLD ABC

Non-manifold ABC contains 50 shapes built from watertight CAD models randomly selected from ABC (Koch et al., 2019): we first scale two CAD models so that they overlap well, then rotate one by 90° about a coordinate axis, and intersect them. Fig. 9 shows examples.

C.3 QMDF

QMDF is built from 14 point clouds that we collected ourselves; for each of them, we learn a medial field with Q-MDF (Kong et al., 2026). QMDF provides no $\mathcal{C}^{\mathrm{gt}}$, so we do not report non-manifold metrics on it; its ground truth topology is annotated manually. The densities of the QMDF point clouds vary widely, so we first apply *voxel downsampling* to them. We normalize the point cloud to $[-1, 1]$ and build a 96^3 voxel grid over $[-1.05, 1.05]^3$. In each non-empty voxel, we keep only the original point closest to the voxel center. This step neither interpolates nor adds points, so the number of points is not fixed. After downsampling, the QMDF point clouds have about 1.5K to 50K points. Fig. 10 shows examples.

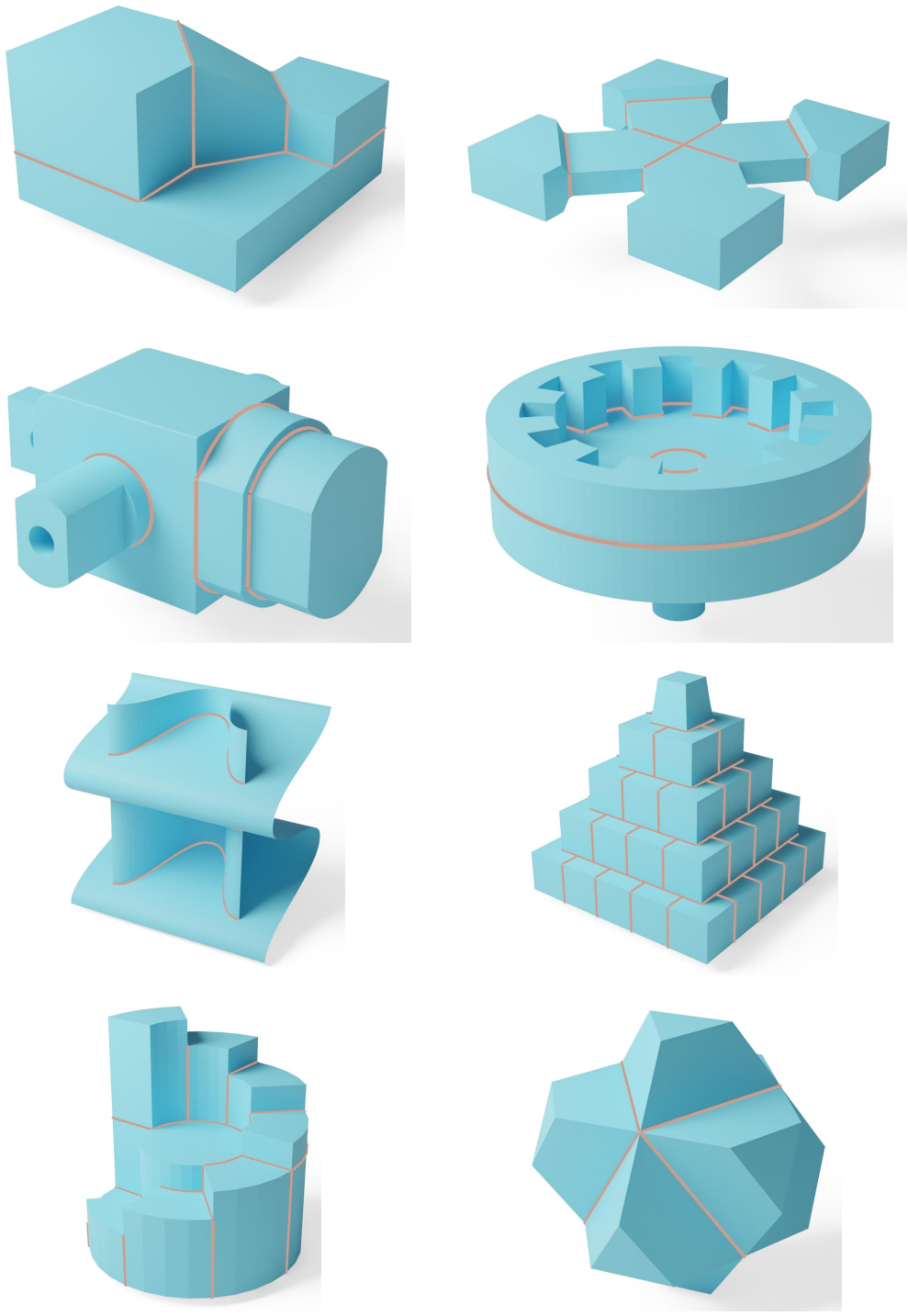


Figure 8: Examples from our constructed non-manifold test set. Orange curves mark the ground truth non-manifold curves $\mathcal{C}^{\mathrm{gt}}$.

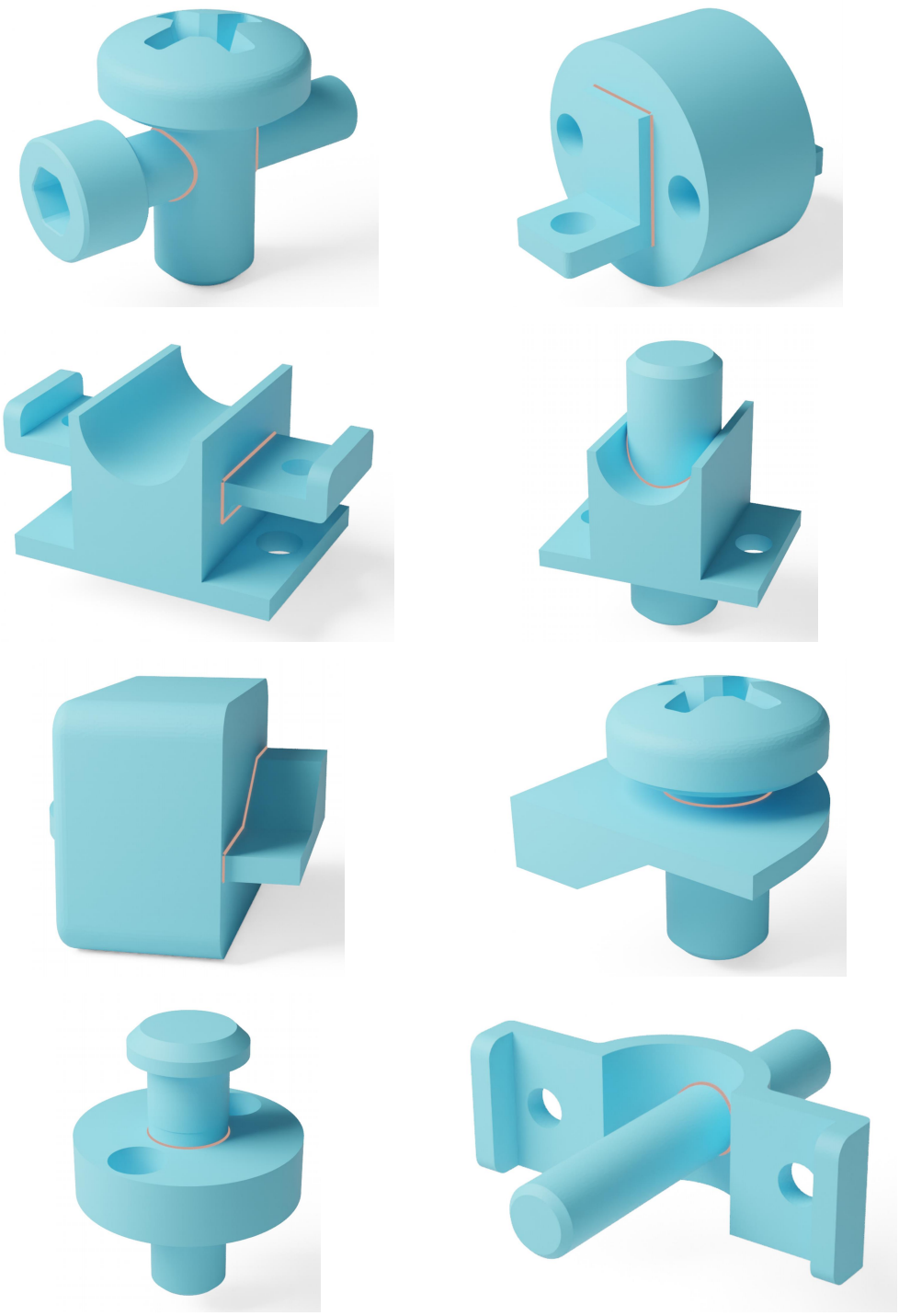


Figure 9: Examples from the Non-manifold ABC test set. Each shape is obtained by intersecting two CAD models from ABC (Koch et al., 2019). Orange curves mark the ground truth non-manifold curves $\mathcal{C}^{\text{gt}}$.

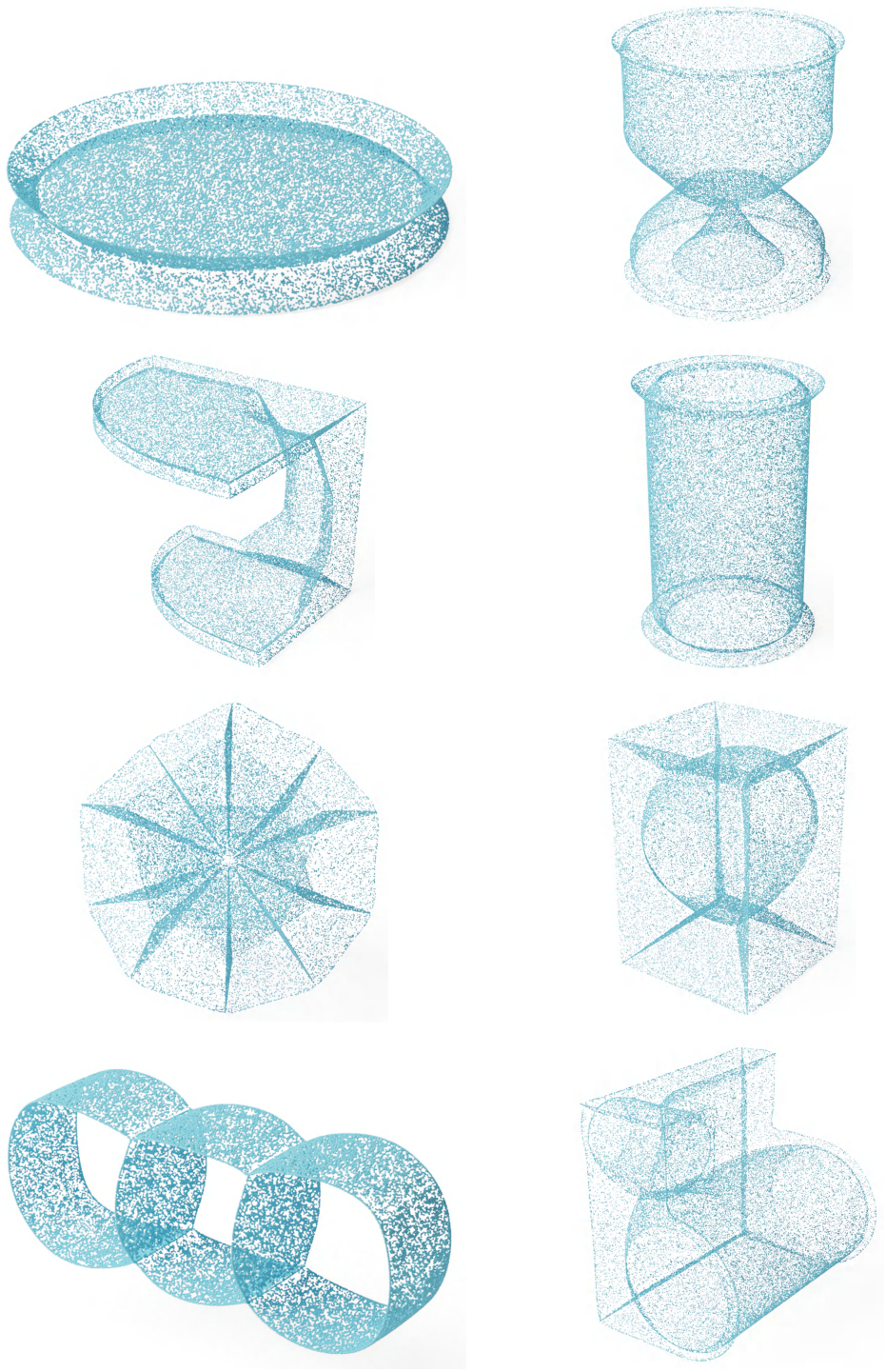


Figure 10: Examples of input point clouds from the QMDF test set after voxel downsampling. QMDF provides no ground truth non-manifold curves $\mathcal{C}^{\mathrm{gt}}$.

D BASELINES

DCUDF (Hou et al., 2023), DMUDF (Zhang et al., 2023), MIND (Chen et al., 2026), and DCx (Bao et al., 2026) extract a mesh from a UDF. DCUDF shrinks a double-covering surface onto the zero level set, so its output never contains non-manifold edges. DMUDF applies DC to the UDF, and the non-manifold edges it outputs arise only as a by-product of local connectivity decisions. MIND derives material-interface labels from the UDF and runs multi-material MC, and DCx extends DC by a lookup table over $2 \times 2 \times 2$ cubes; both output non-manifold edges by design. SALS (Ren & Hou, 2025) replaces distance values by intersection flags on line segments and extracts a mesh with an edge-based DC, which can also output non-manifold edges. For the four UDF extractors, we fit one UDF per shape with DEUDF (Xu et al., 2025), following its official implementation with 300K points per shape, and all four extract a mesh from this same UDF. SALS takes a point cloud of 40K points as input, as recommended by its authors. All baselines use their official code and default parameters.

E EVALUATION DETAILS

We scale the ground truth and the outputs of all methods by the ground truth bounding box so that its longest side is 1, i.e., into $[-0.5, 0.5]$, and compute all metrics in these evaluation coordinates. We do not align any method or shape separately.

For CD, we sample 100K points uniformly on $\mathcal{M}$ and on the ground truth surface, find nearest neighbors by the unsquared L2 distance, and average the two directions. On QMDF, the reference is its ground truth point cloud; if it has more than 100K points, we draw 100K points without replacement with a fixed random seed. For non-manifold CD and F1@0.01, we split every non-manifold edge of $\mathcal{M}$ into segments no longer than 0.0025, take the midpoint of each segment, and compare these points with points sampled on $\mathcal{C}^{\mathrm{gt}}$. Non-manifold CD is computed in the same way as CD. F1@0.01 computes precision and recall with a distance threshold of 0.01, i.e., 1% of the longest side. DCUDF outputs no non-manifold edges, so its non-manifold CD and F1@0.01 are undefined and marked NA in Table 1; its topology errors are computed as for the other methods.

For topology, we compute the Betti numbers b_0, b_1, b_2 of $\mathcal{M}$ and of the ground truth with $\mathbb{Z}_2$ coefficients, compute the Euler characteristic as $\chi = V - E + F$, and report the absolute errors of both.

On noisy inputs, all outputs are compared with the noise-free ground truth. If a method outputs no mesh for a shape, the shape counts as a failure in #Success and is excluded from the averages of all other metrics; failed shapes are not filled with zeros.

F ABLATION DETAILS

All variants in Sec. 4.4 share the backbone, loss, graph construction, and union-find of our method (Sec. 3.2) and differ only in the part named below. All variants are evaluated on the validation set of simple composite shapes. ARI is the adjusted Rand index of the predicted patch point sets against $\mathcal{P}_k^{\mathrm{gt}}$; matched mIoU is computed after matching predicted and ground truth patches.

Input to the affinity head. We replace only the input to Aff (Table 3). The first variant passes the scaled inner product $\boldsymbol{f}_i^\top \boldsymbol{f}_j / \sqrt{d}$ through a sigmoid and uses it directly as the affinity. The second variant concatenates $\boldsymbol{f}_i$ and $\boldsymbol{f}_j$ and feeds them to an MLP.

Pairwise versus per-point formulation. We keep the backbone fixed and replace the pairwise affinity with two per-point formulations (Table 4). A per-point classifier predicts whether each point lies on a non-manifold curve. A per-point distance regression predicts the distance from each point to $\mathcal{C}^{\mathrm{gt}}$, and points predicted to lie close to $\mathcal{C}^{\mathrm{gt}}$ are taken as points on non-manifold curves. Both variants remove the predicted points on non-manifold curves and take the connected components of the remaining KNN graph as the patch point sets.

Sensitivity to the threshold τ. We evaluate E_τ as a binary classification of the edges of the KNN graph, where an edge is positive if both endpoints belong to the same $\mathcal{P}_k^{\mathrm{gt}}$, and vary τ from 0.1 to 0.9 (Table 5).